Symmetry Reduction for Reo and Constraint Automata

Author: Amin Timany
Supervisors: Dr. S. Klüppelholz
Dr. J. Klein
Professor: Prof. C. Baier

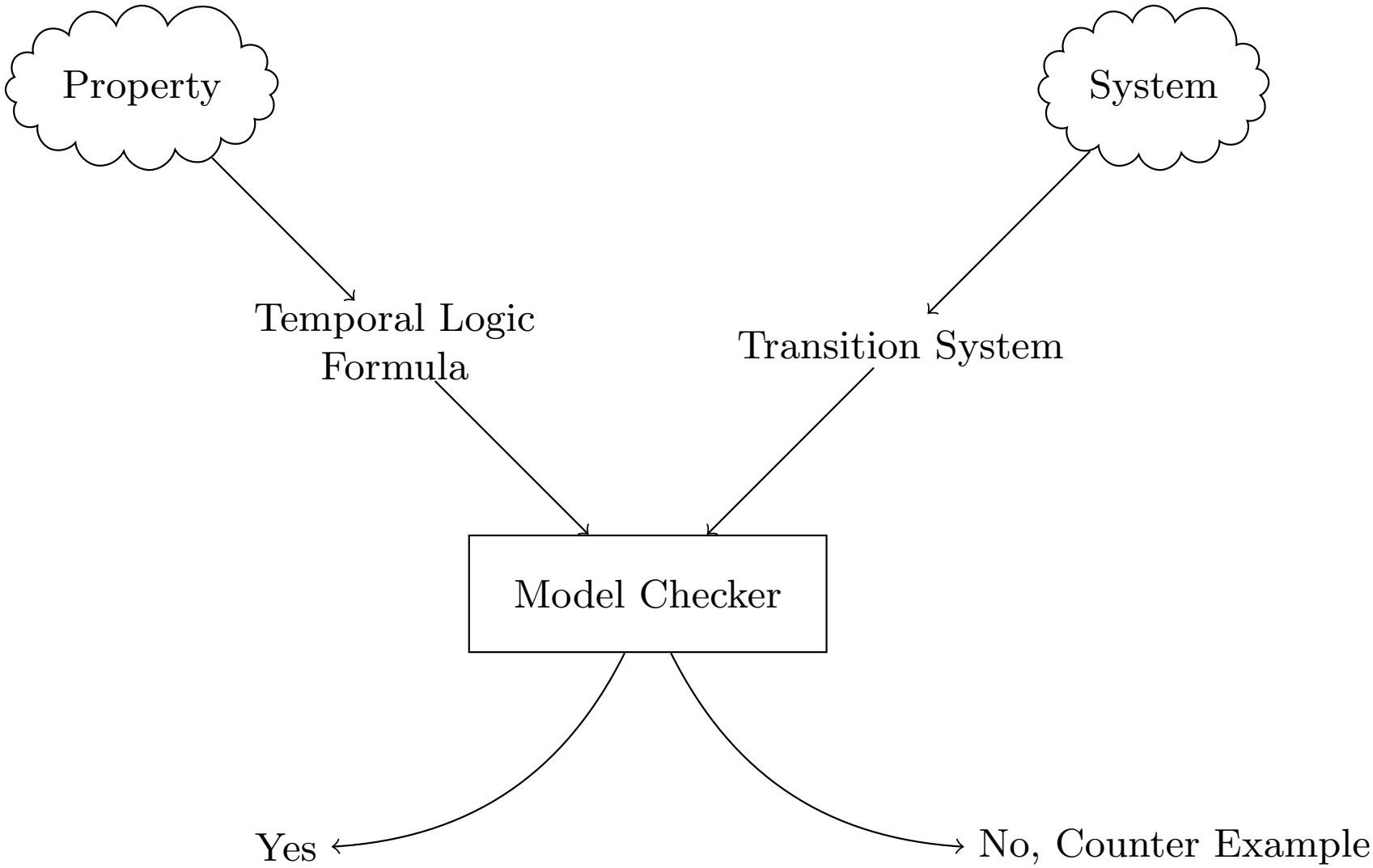
International Center for Computational Logic

Technical University of Dresden

June 20, 2013

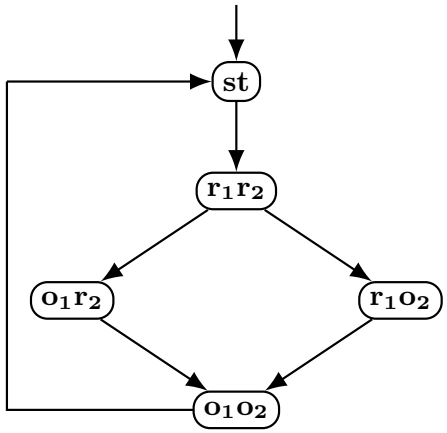
Outline

- Model Checking: an automated verification approach



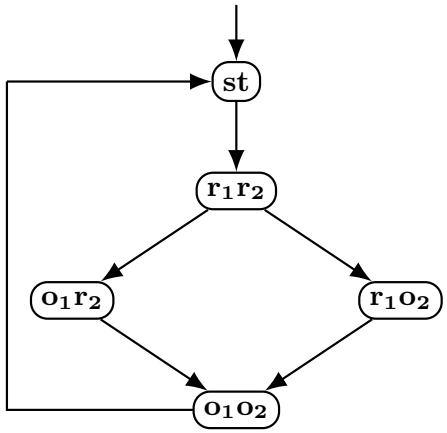
- Model Checking

- Model Checking
  - Model as Transition System



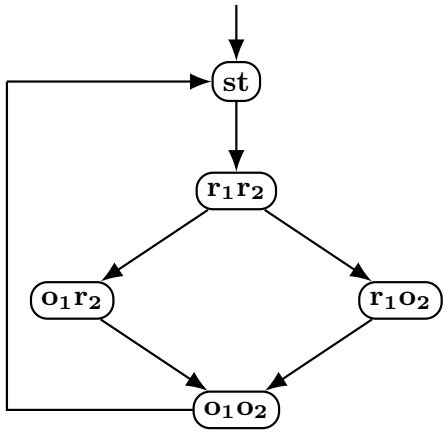
A process requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties



A process requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

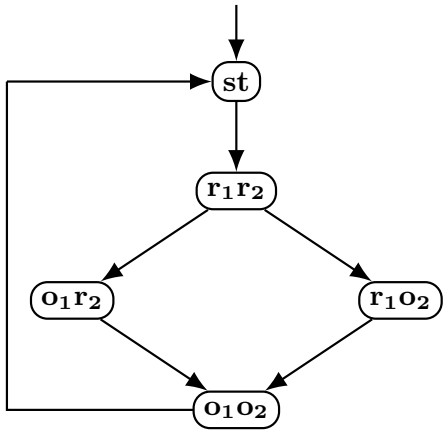
- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby



A process requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

Model Checking and State Space Explosion

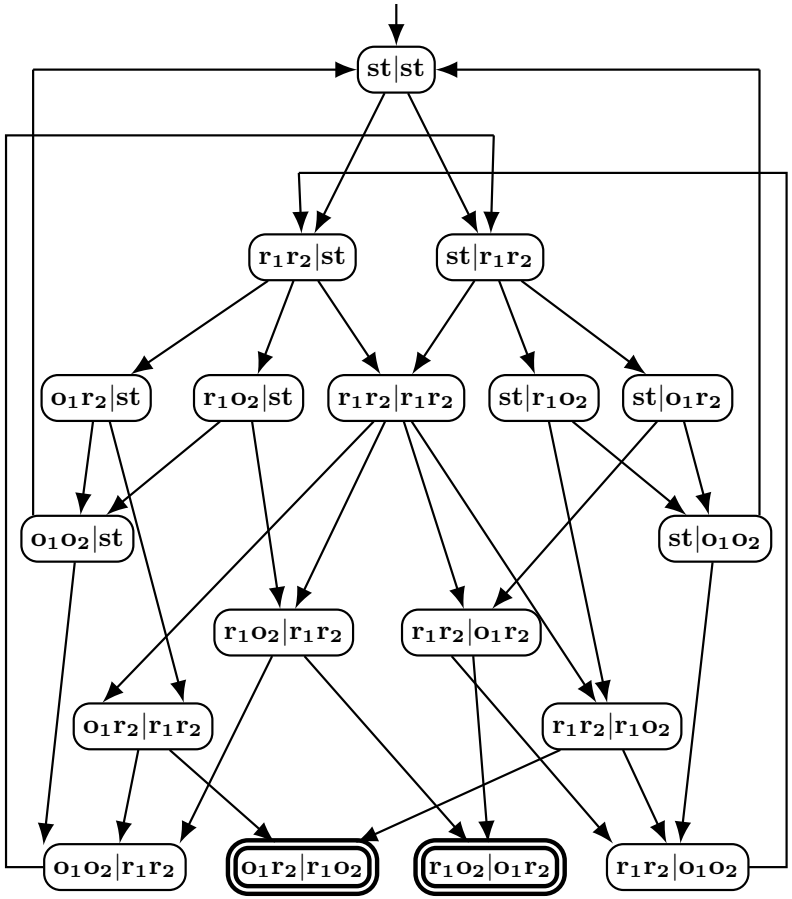
- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby
    - Mutual exclusion



A process requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

Model Checking and State Space Explosion

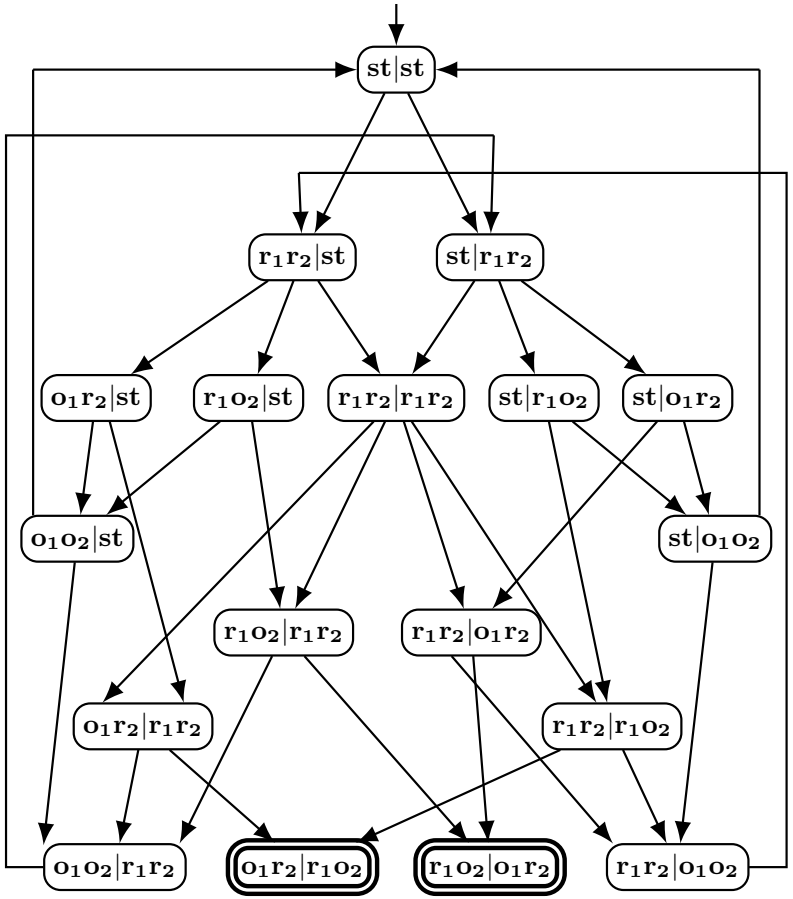
- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby
    - Mutual exclusion
- State Space Explosion



Two processes each requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

Model Checking and State Space Explosion

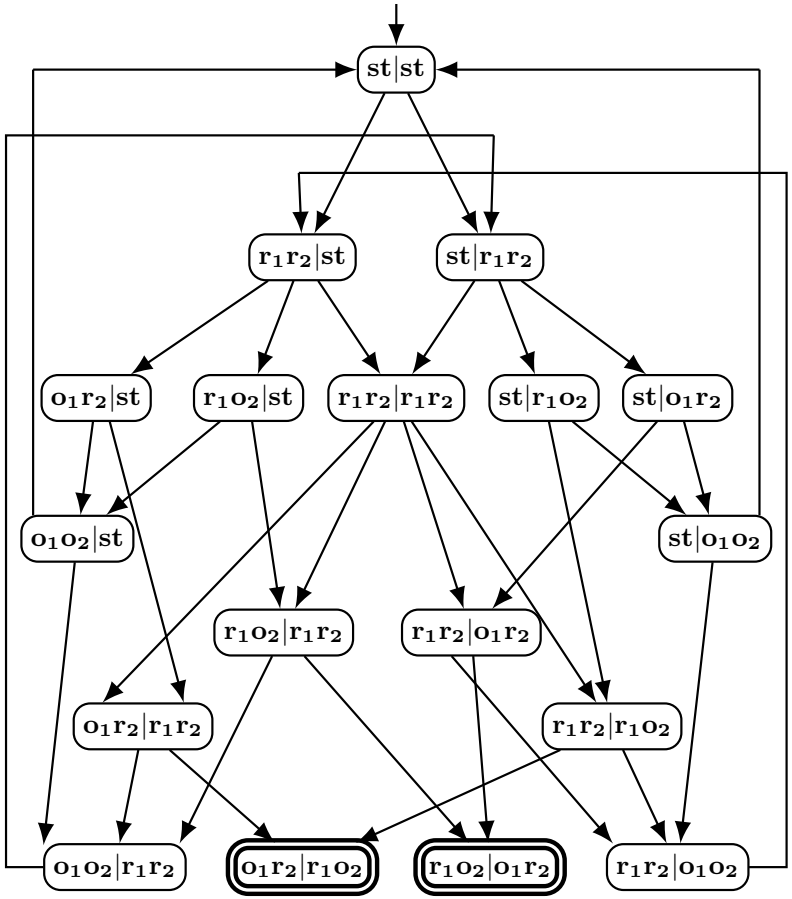
- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby
    - Mutual exclusion
- State Space Explosion
  - States are tuples of States of the Subsystems



Two processes each requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

Model Checking and State Space Explosion

- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby
    - Mutual exclusion
- State Space Explosion
  - States are tuples of States of the Subsystems
  - States of the composite system: $\mathcal{O}(k^n)$ states



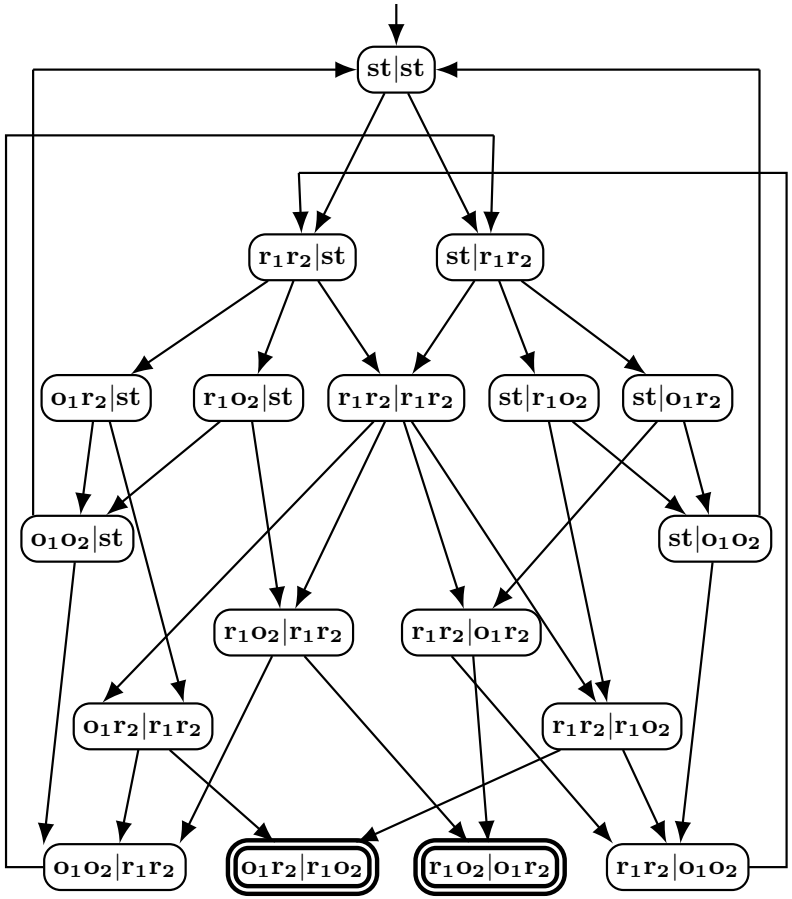
Two processes each requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

k = number of states (here 5)

n = number of copies (here 2)

Model Checking and State Space Explosion

- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby
    - Mutual exclusion
- State Space Explosion
  - States are tuples of States of the Subsystems
  - States of the composite system: $\mathcal{O}(k^n)$ states
  - A well known problem vastly investigated



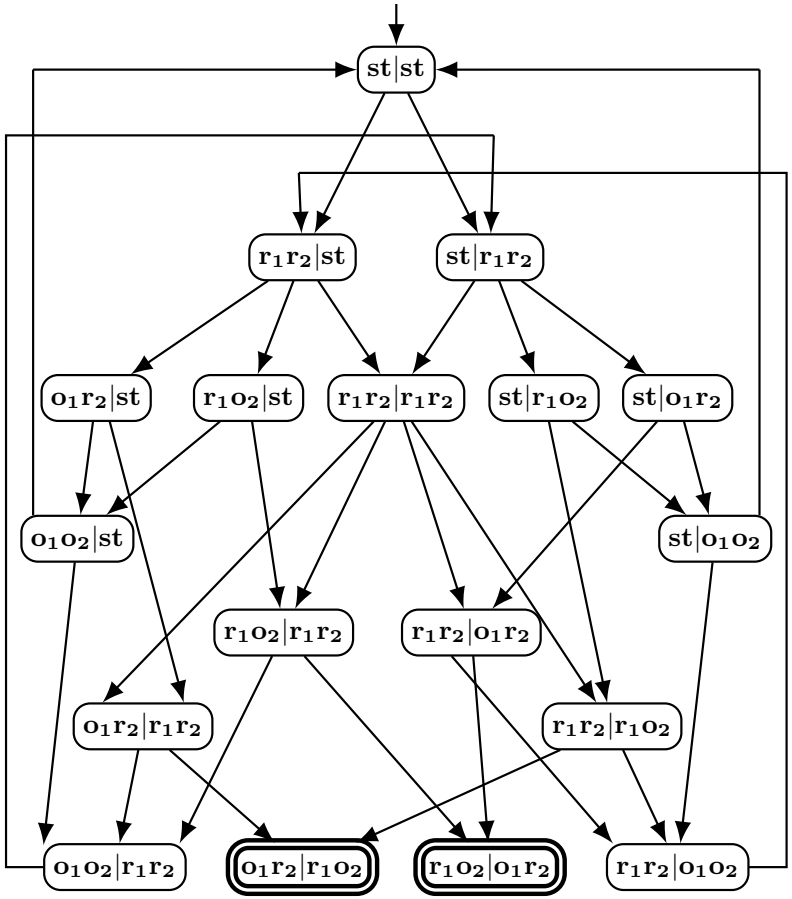
Two processes each requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

k = number of states (here 5)

n = number of copies (here 2)

Model Checking and State Space Explosion

- Model Checking
  - Model as Transition System
  - Verification of Temporal Properties
    - After requesting, always obtains both resources before going back to standby
    - Mutual exclusion
- State Space Explosion
  - States are tuples of States of the Subsystems
  - States of the composite system: $\mathcal{O}(k^n)$ states
  - A well known problem vastly investigated
  - Symmetry to tackle this problem

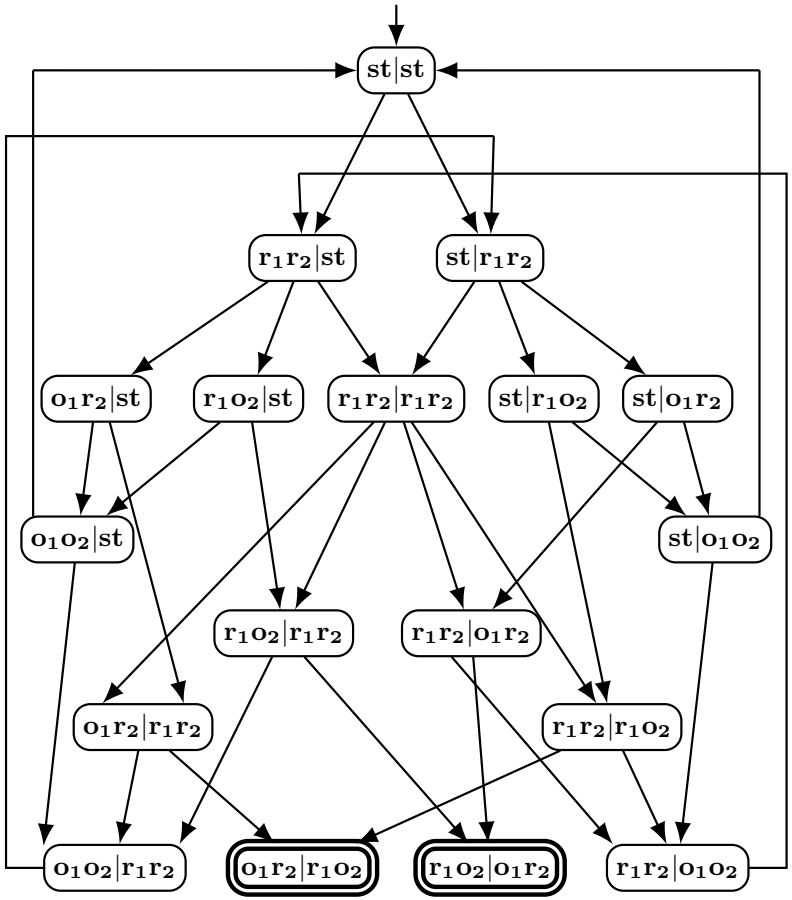


Two processes each requesting two resources, $\mathbf{r_1}$ and $\mathbf{r_2}$.

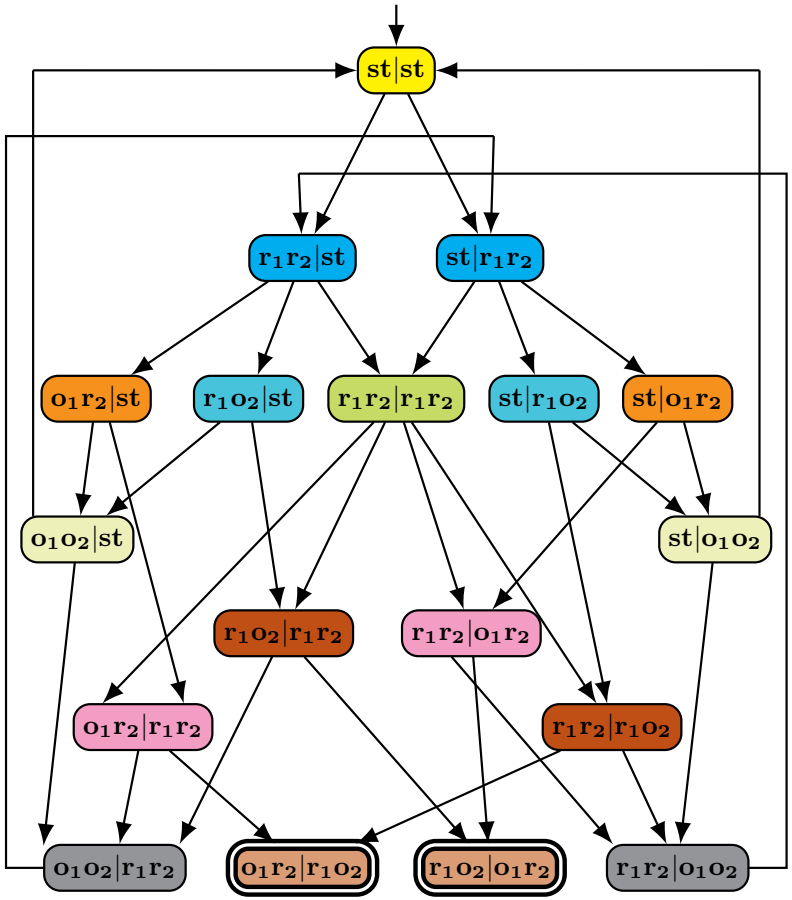
k = number of states (here 5)

n = number of copies (here 2)

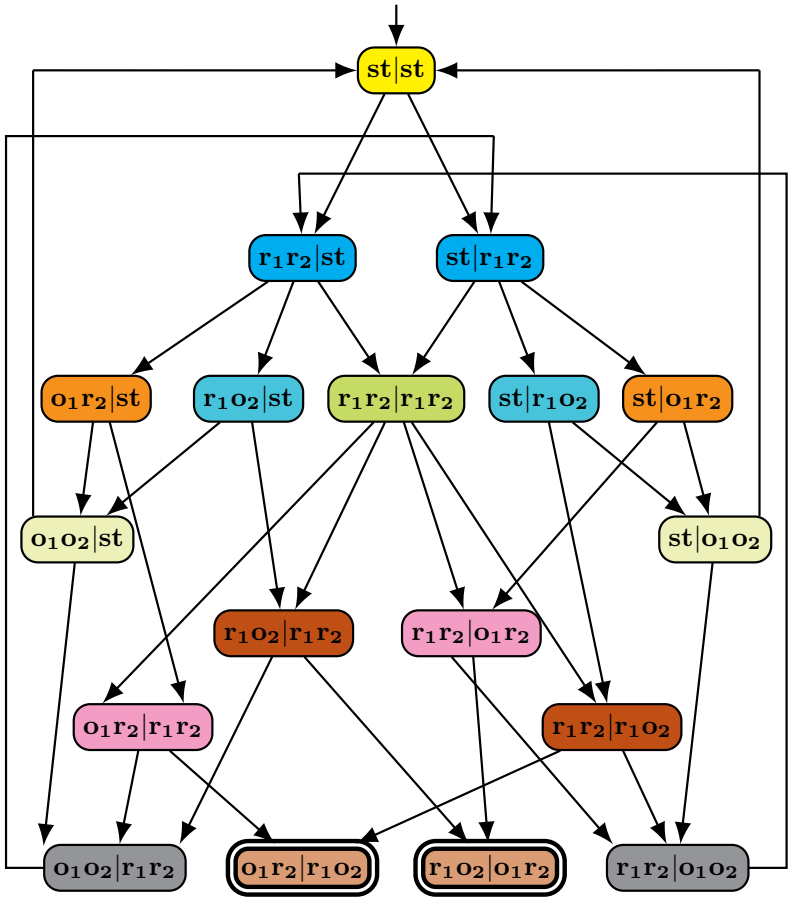
- Symmetry


st|st
r1r2|st
st|r1r2
o1r2|st
r1o2|st
r1r2|r1r2
st|r1o2
st|o1r2
o1o2|st
st|o1o2
r1o2|r1r2
r1r2|o1r2
o1r2|r1r2
r1r2|r1o2
o1o2|r1r2
o1r2|r1o2
r1o2|o1r2
r1r2|o1o2

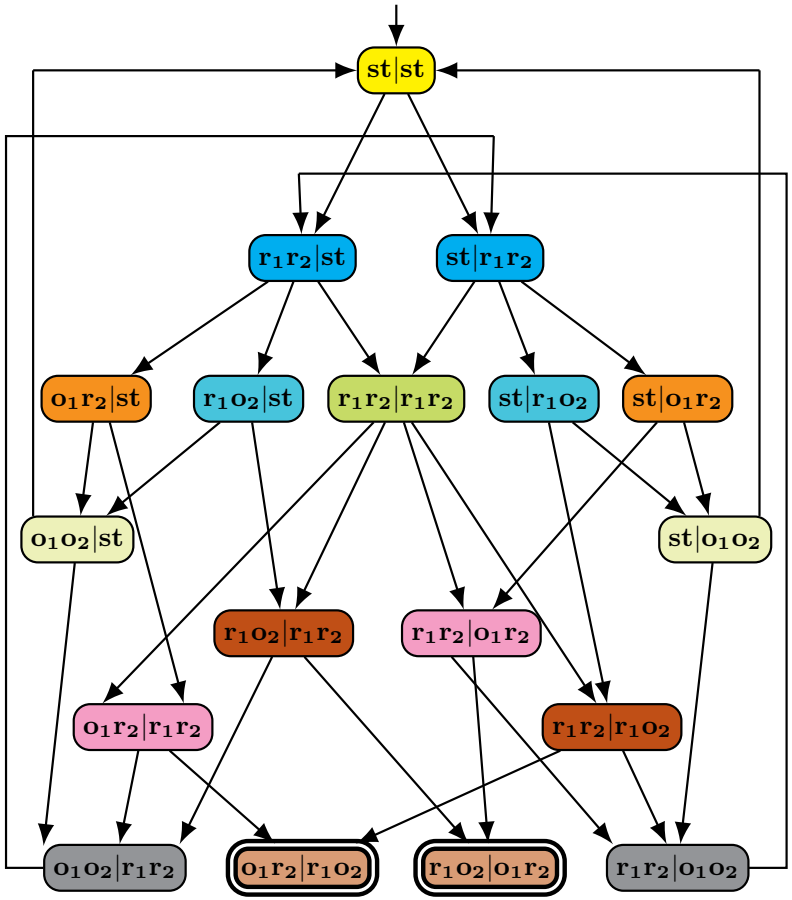
- Symmetry
- Swapping of states of symmetric sub-systems preserves transitions
- e.g., $(\mathbf{st}|\mathbf{o_1 r_2}) \rightarrow (\mathbf{st}|\mathbf{o_1 o_2}) \Leftrightarrow$
  $(\mathbf{o_1 r_2}|\mathbf{st}) \rightarrow (\mathbf{o_1 o_2}|\mathbf{st})$



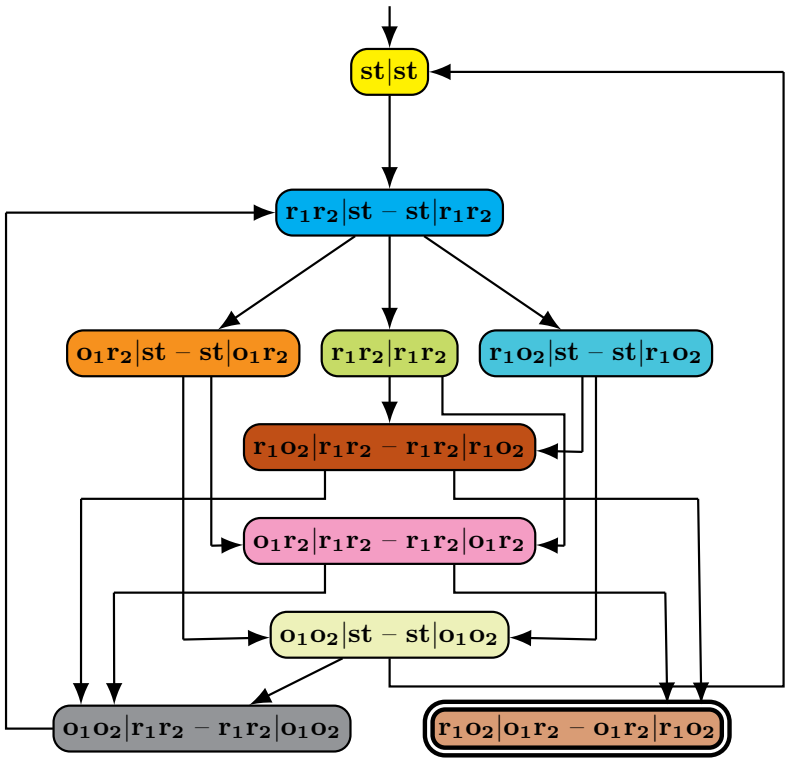
- Symmetry
- Swapping of states of symmetric sub-systems preserves transitions
- e.g., $(\mathbf{st}|\mathbf{o_1 r_2}) \rightarrow (\mathbf{st}|\mathbf{o_1 o_2}) \Leftrightarrow (\mathbf{o_1 r_2}|\mathbf{st}) \rightarrow (\mathbf{o_1 o_2}|\mathbf{st})$
- Well known approach used in various formalisms

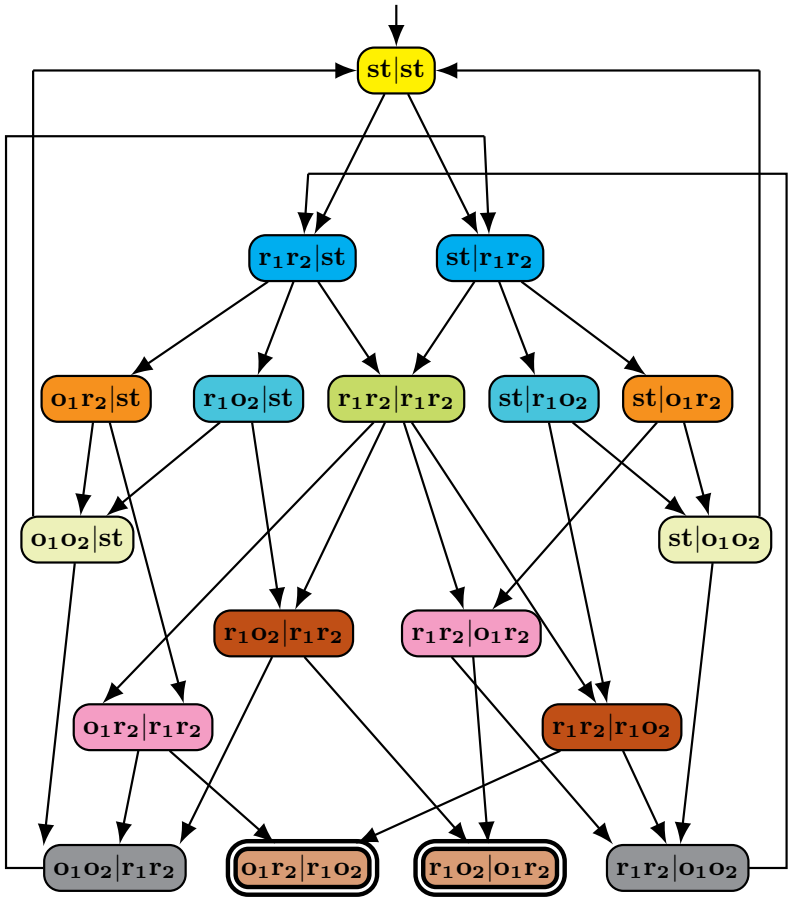


- Symmetry
- Swapping of states of symmetric sub-systems preserves transitions
- e.g., $(\mathbf{st}|\mathbf{o_1 r_2}) \rightarrow (\mathbf{st}|\mathbf{o_1 o_2}) \Leftrightarrow$
  $(\mathbf{o_1 r_2}|\mathbf{st}) \rightarrow (\mathbf{o_1 o_2}|\mathbf{st})$
- Well known approach used in various formalisms
- Represent symmetric states with a representative

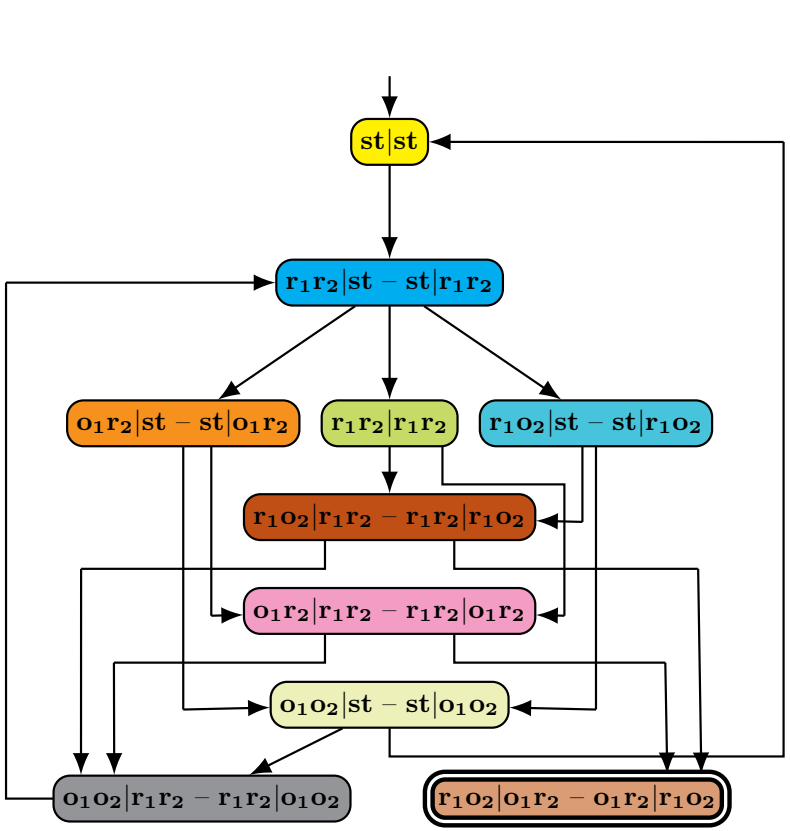


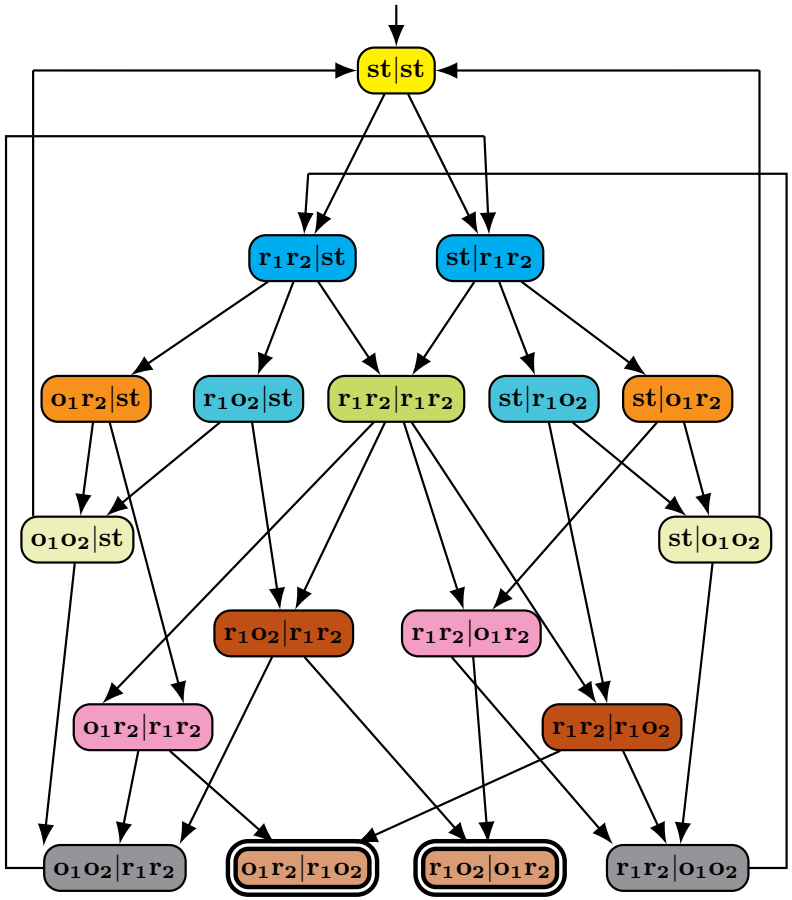
- Symmetry Reduction



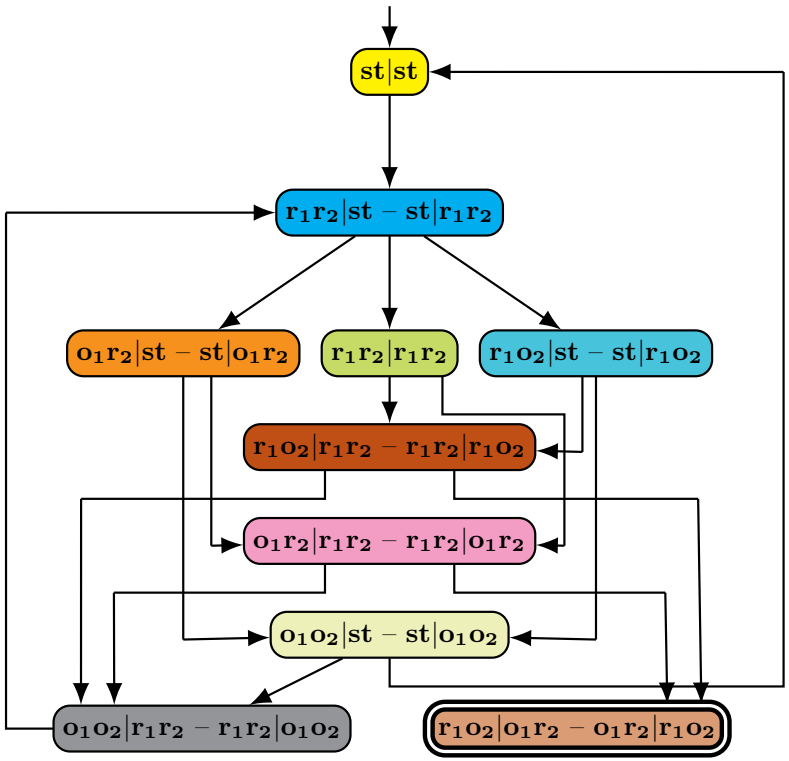


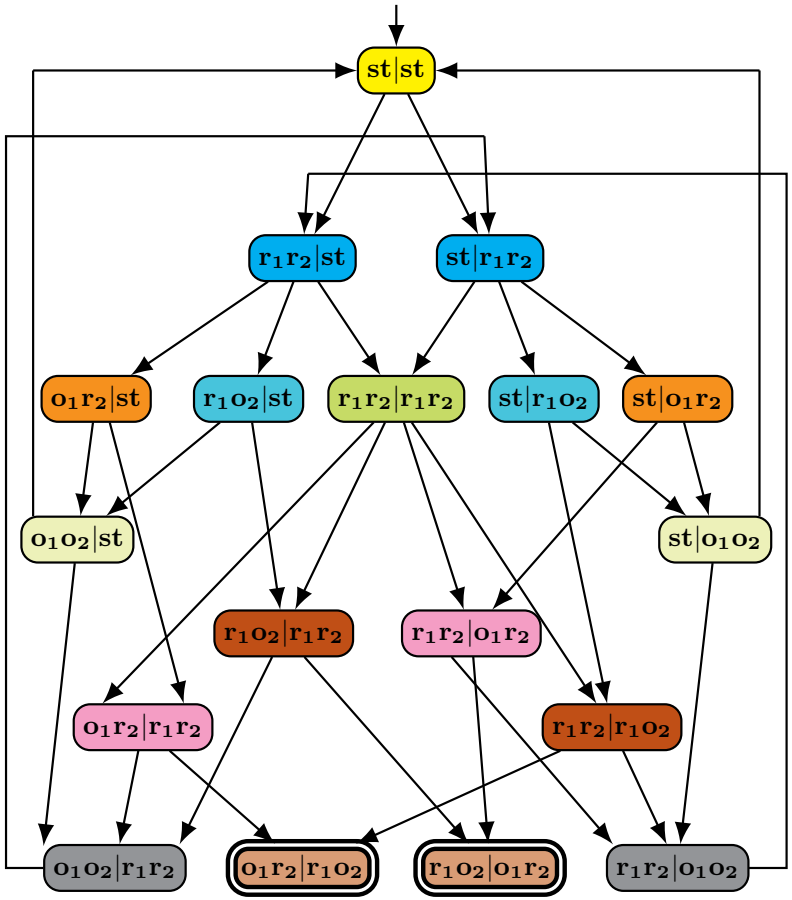
- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved



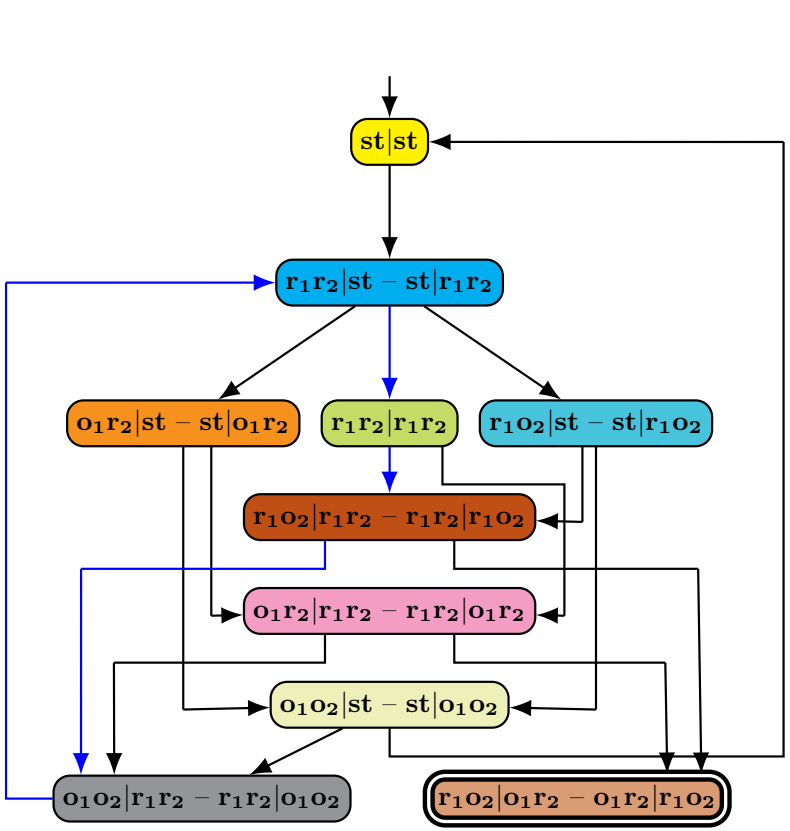


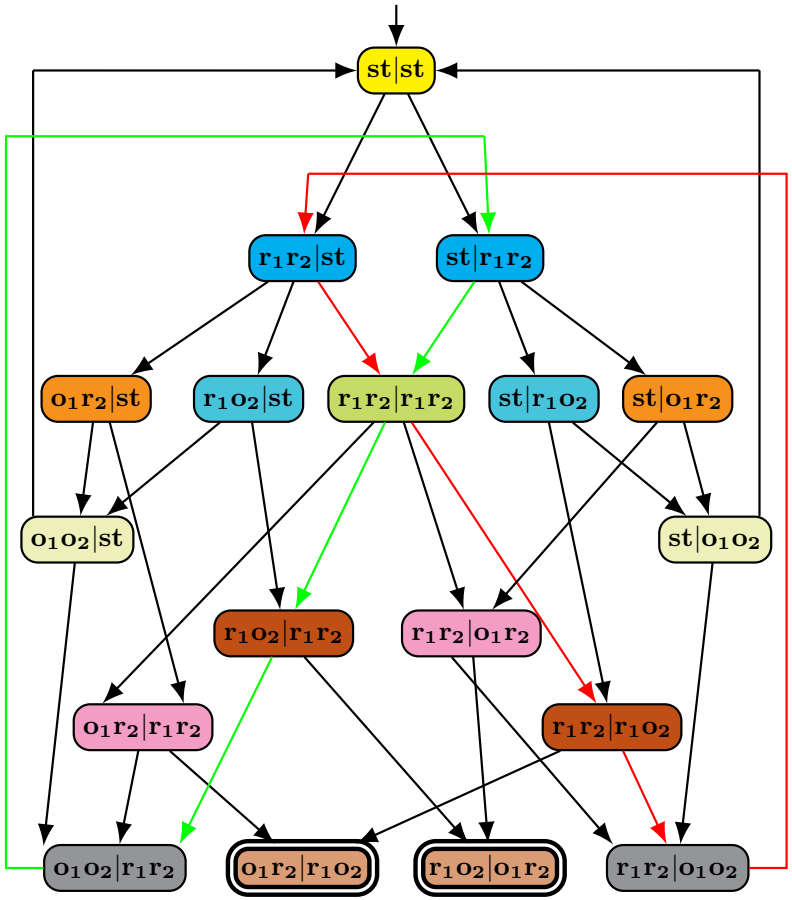
- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved
  - Mutual exclusion (can be checked here)





- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved
  - Mutual exclusion (can be checked here)
  - Fairness (can't be checked here)





- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved
  - Mutual exclusion (can be checked here)
  - Fairness (can't be checked here)
- Represent states of reduced system by counting states of symmetric subsystem

- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved
  - Mutual exclusion (can be checked here)
  - Fairness (can't be checked here)
- Represent states of reduced system by counting states of symmetric subsystem
- $(\mathbf{st}|\mathbf{r_1 r_2} - -\mathbf{r_1 r_2}|\mathbf{st}) \rightsquigarrow \{(\mathbf{st}, \mathbf{1}), (\mathbf{r_1 r_2}, \mathbf{1}), (\mathbf{o_1 r_2}, \mathbf{0}), (\mathbf{r_1 o_2}, \mathbf{0}), (\mathbf{o_1 o_2}, \mathbf{0})\}$

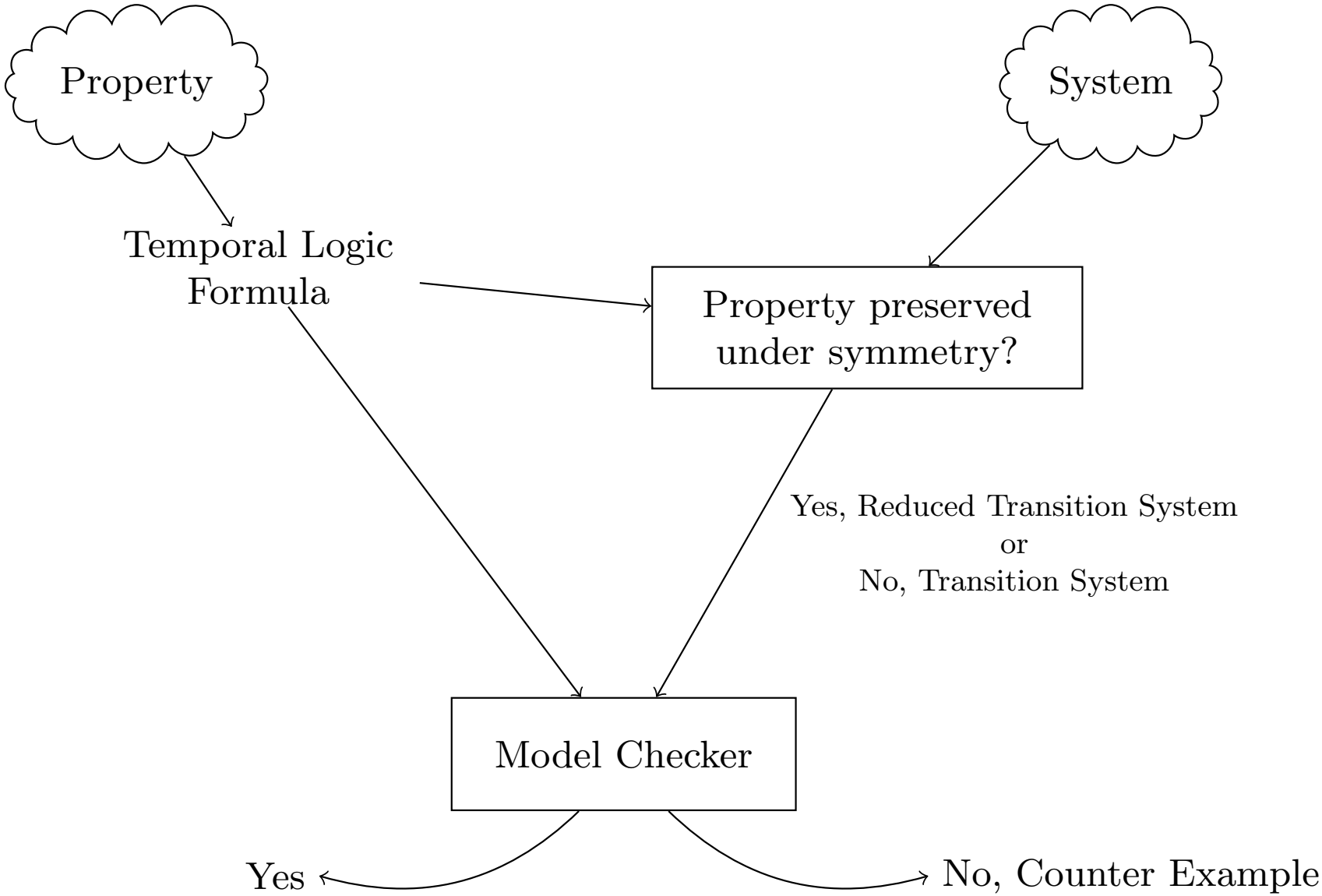
- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved
  - Mutual exclusion (can be checked here)
  - Fairness (can't be checked here)
- Represent states of reduced system by counting states of symmetric subsystem
- $(\mathbf{st}|\mathbf{r_1 r_2} - -\mathbf{r_1 r_2}|\mathbf{st}) \rightsquigarrow \{(\mathbf{st}, \mathbf{1}), (\mathbf{r_1 r_2}, \mathbf{1}), (\mathbf{o_1 r_2}, \mathbf{0}), (\mathbf{r_1 o_2}, \mathbf{0}), (\mathbf{o_1 o_2}, \mathbf{0})\}$
- States of the reduced system: $\mathcal{O}((n+1)^k)$

k = number of states (here 5) n = number of copies (here 2)

- Symmetry Reduction
- Only properties that do not depend on order of symmetric sub-systems are preserved
  - Mutual exclusion (can be checked here)
  - Fairness (can't be checked here)
- Represent states of reduced system by counting states of symmetric subsystem
- $(\mathbf{st}|\mathbf{r_1 r_2} - -\mathbf{r_1 r_2}|\mathbf{st}) \rightsquigarrow \{(\mathbf{st}, \mathbf{1}), (\mathbf{r_1 r_2}, \mathbf{1}), (\mathbf{o_1 r_2}, \mathbf{0}), (\mathbf{r_1 o_2}, \mathbf{0}), (\mathbf{o_1 o_2}, \mathbf{0})\}$
- States of the reduced system: $\mathcal{O}((n+1)^k)$
- We can reduce without computing the composite system

k = number of states (here 5) n = number of copies (here 2)

- Symmetry Reduction



Outline

- Reo (Arbab, 2004)

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow
- Vereofy (Developed in TU-Dresden)

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow
- Vereofy (Developed in TU-Dresden)
  - Model checking tool

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow
- Vereofy (Developed in TU-Dresden)
  - Model checking tool
  - Model specification in two languages based on Reo and constraint automata

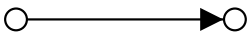
- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow
- Vereofy (Developed in TU-Dresden)
  - Model checking tool
  - Model specification in two languages based on Reo and constraint automata
  - CARML specifies constraint automata of components

- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow
- Vereofy (Developed in TU-Dresden)
  - Model checking tool
  - Model specification in two languages based on Reo and constraint automata
  - CARML specifies constraint automata of components
  - RSL specifies Reo circuit

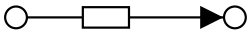
- Reo (Arbab, 2004)
  - Exogenous channel based coordination language
  - Devised as coordination system for component based systems
- Constraint Automata (Baier et. al., 2006)
  - Devised as semantics for Reo
  - Automata model endowed with data-flow locations
  - Transitions with constraints over data-flow
- Vereofy (Developed in TU-Dresden)
  - Model checking tool
  - Model specification in two languages based on Reo and constraint automata
  - CARML specifies constraint automata of components
  - RSL specifies Reo circuit
  - Symmetry reduction approach discussed here is implemented in it

- Reo

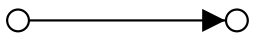
- Reo
  - Channels (channel ends as circles)
    1. Synchronous Channel



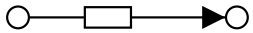
    2. FIFO Channel



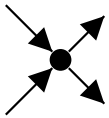
- Reo
  - Channels (channel ends as circles)
    1. Synchronous Channel



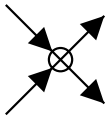
    2. FIFO Channel



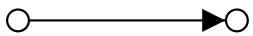
  - Nodes (Collections of channel ends)
    1. Replicate Node (replicates data of a single input to all outputs):



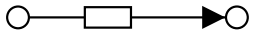
    2. Route Node (replicates data of a single input to a single outputs):



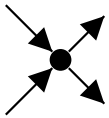
- Reo
  - Channels (channel ends as circles)
    1. Synchronous Channel



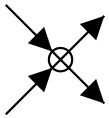
    2. FIFO Channel



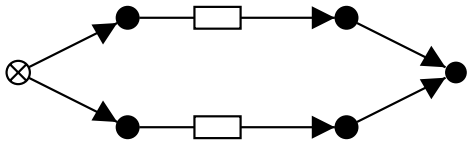
  - Nodes (Collections of channel ends)
    1. Replicate Node (replicates data of a single input to all outputs):



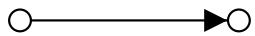
    2. Route Node (replicates data of a single input to a single outputs):



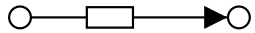
  - Reo circuit



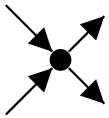
- Reo
  - Channels (channel ends as circles)
    1. Synchronous Channel



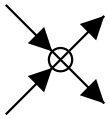
    2. FIFO Channel



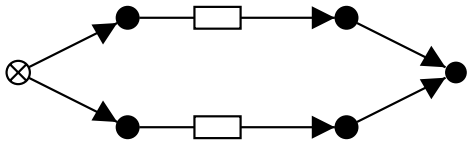
  - Nodes (Collections of channel ends)
    1. Replicate Node (replicates data of a single input to all outputs):



    2. Route Node (replicates data of a single input to a single outputs):

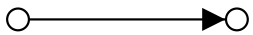


  - Reo circuit

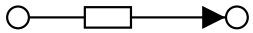


  - Components run in parallel and Synchronize on data-flow (exogenous coordination)

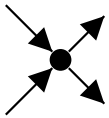
- Reo
  - Channels (channel ends as circles)
    1. Synchronous Channel



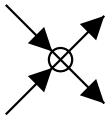
    2. FIFO Channel



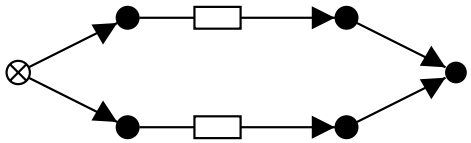
  - Nodes (Collections of channel ends)
    1. Replicate Node (replicates data of a single input to all outputs):



    2. Route Node (replicates data of a single input to a single outputs):

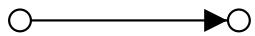


  - Reo circuit

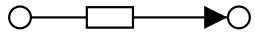


  - Components run in parallel and Synchronize on data-flow (exogenous coordination)
  - State space explosion problem

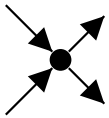
- Reo
  - Channels (channel ends as circles)
    1. Synchronous Channel



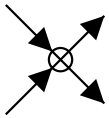
    2. FIFO Channel



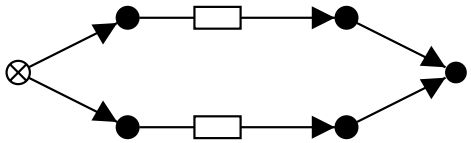
  - Nodes (Collections of channel ends)
    1. Replicate Node (replicates data of a single input to all outputs):



    2. Route Node (replicates data of a single input to a single outputs):



  - Reo circuit



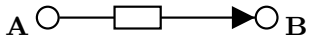
  - Components run in parallel and Synchronize on data-flow (exogenous coordination)
  - State space explosion problem
  - An internal operation can always take place

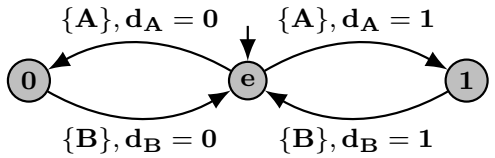
- Constraint Automata

- Constraint Automata
  - Channel ends represented as data-flow locations

- Constraint Automata
  - Channel ends represented as data-flow locations
  - Transitions labeled with constraint on data-flow through data-flow locations

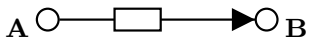
- Constraint Automata
  - Channel ends represented as data-flow locations
  - Transitions labeled with constraint on data-flow through data-flow locations

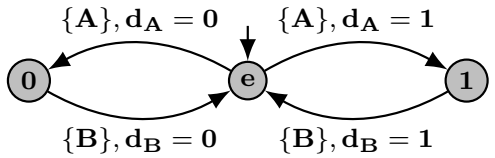




$\mathcal{N} = \{\mathbf{A}, \mathbf{B}\}, \mathcal{N}_{\mathbf{in}} = \{\mathbf{A}\}, \mathcal{N}_{\mathbf{out}} = \{\mathbf{B}\}$

- Constraint Automata
  - Channel ends represented as data-flow locations
  - Transitions labeled with constraint on data-flow through data-flow locations

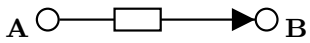


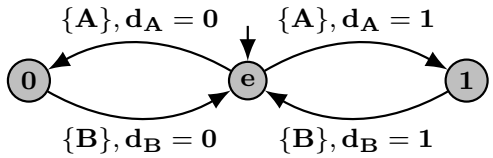


$\mathcal{N} = \{\mathbf{A}, \mathbf{B}\}, \mathcal{N}_{\mathbf{in}} = \{\mathbf{A}\}, \mathcal{N}_{\mathbf{out}} = \{\mathbf{B}\}$

  - Symmetry, defined by transition preservation

- Constraint Automata
  - Channel ends represented as data-flow locations
  - Transitions labeled with constraint on data-flow through data-flow locations

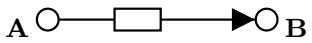


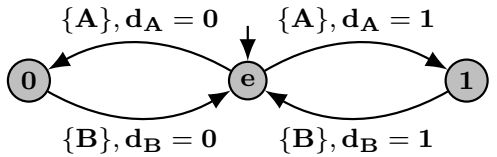


$\mathcal{N} = \{\mathbf{A}, \mathbf{B}\}, \mathcal{N}_{\mathbf{in}} = \{\mathbf{A}\}, \mathcal{N}_{\mathbf{out}} = \{\mathbf{B}\}$

  - Symmetry, defined by transition preservation
  - Constraints considered on semantics level

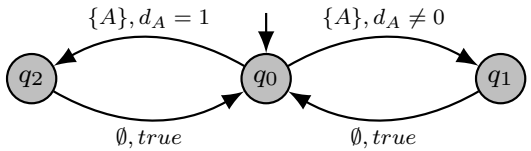
- Constraint Automata
  - Channel ends represented as data-flow locations
  - Transitions labeled with constraint on data-flow through data-flow locations





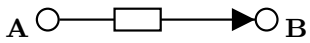
$\mathcal{N} = \{\mathbf{A}, \mathbf{B}\}, \mathcal{N}_{\mathbf{in}} = \{\mathbf{A}\}, \mathcal{N}_{\mathbf{out}} = \{\mathbf{B}\}$

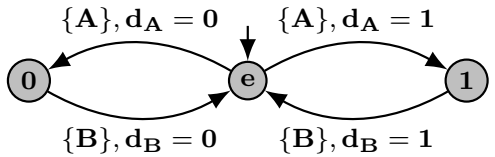
  - Symmetry, defined by transition preservation
  - Constraints considered on semantics level



$\mathcal{N} = \{A\}, \mathcal{N}_{in} = \{A\}, \mathcal{N}_{out} = \emptyset$

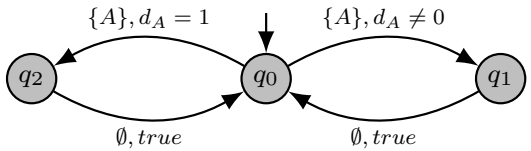
- Constraint Automata
  - Channel ends represented as data-flow locations
  - Transitions labeled with constraint on data-flow through data-flow locations





$\mathcal{N} = \{\mathbf{A}, \mathbf{B}\}, \mathcal{N}_{\mathbf{in}} = \{\mathbf{A}\}, \mathcal{N}_{\mathbf{out}} = \{\mathbf{B}\}$

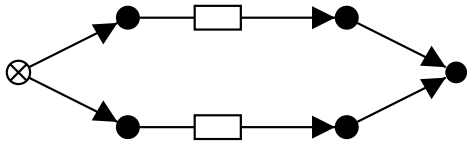
  - Symmetry, defined by transition preservation
  - Constraints considered on semantics level



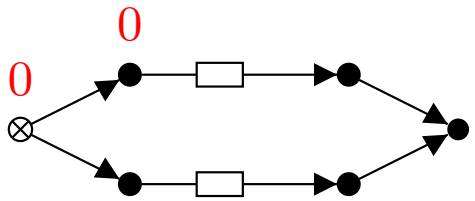
$\mathcal{N} = \{A\}, \mathcal{N}_{in} = \{A\}, \mathcal{N}_{out} = \emptyset$

  - Constraints $\{A\}, d_A \neq 0$ and $\{A\}, d_A = 1$ are equivalent if data domain is $\{0, 1\}$

- Symmetry Blockage

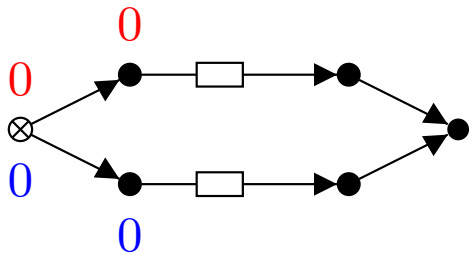


- Symmetry Blockage



$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{0}, \mathbf{e})$$

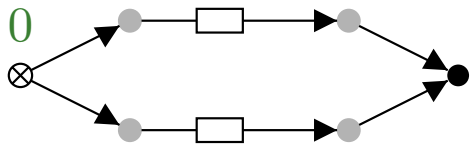
- Symmetry Blockage



$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{0}, \mathbf{e})$$

$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{e}, \mathbf{0})$$

- Symmetry Blockage

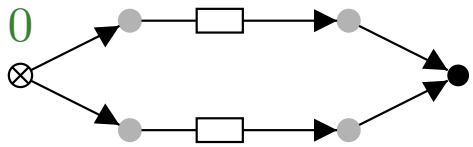


$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{0}, \mathbf{e})$$

$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{e}, \mathbf{0})$$

- We have to hide data-flow locations of symmetric components

- Symmetry Blockage

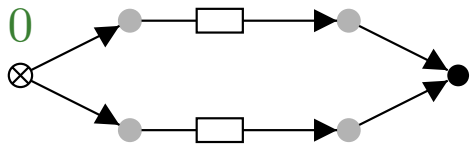


$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{0}, \mathbf{e})$$

$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{e}, \mathbf{0})$$

- We have to hide data-flow locations of symmetric components
- Hiding, in practice removes data-flow locations from CA

- Symmetry Blockage

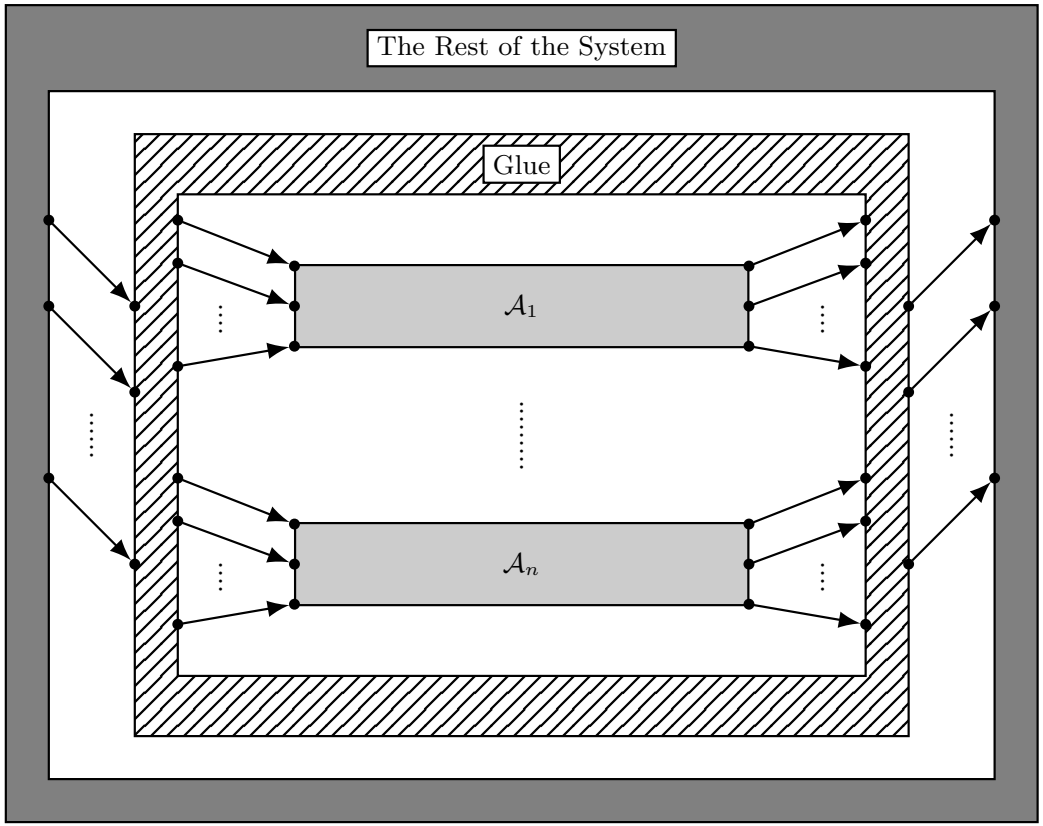


$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{0}, \mathbf{e})$$

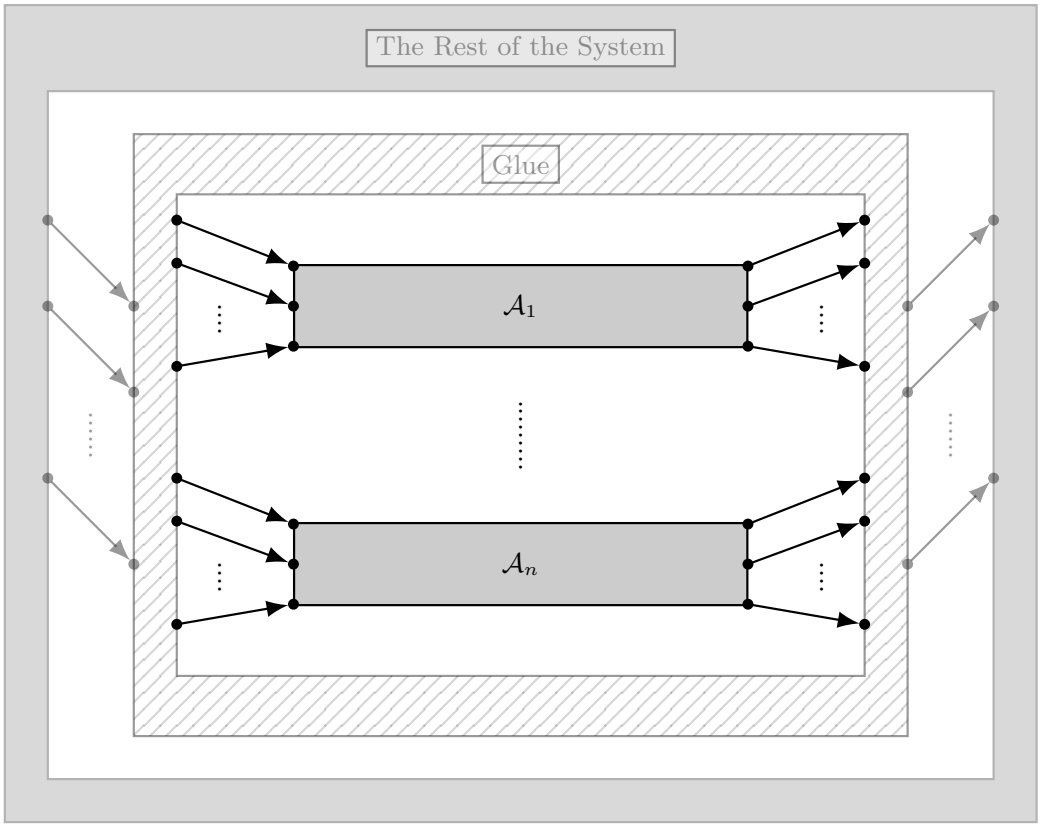
$$(\mathbf{e}, \mathbf{e}) \rightarrow (\mathbf{e}, \mathbf{0})$$

- We have to hide data-flow locations of symmetric components
- Hiding, in practice removes data-flow locations from CA
- But still we have to check transition preservation
- And reduce the whole system (and not only the symmetric fragment)

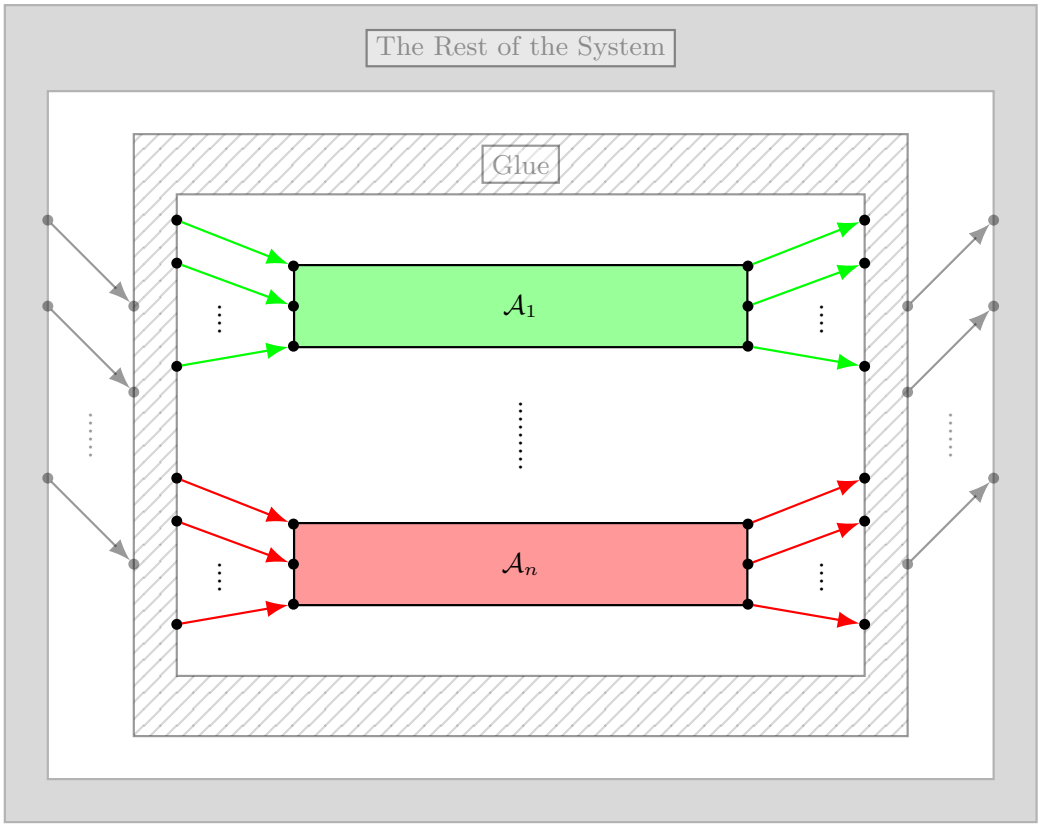
- Three Tier Symmetry



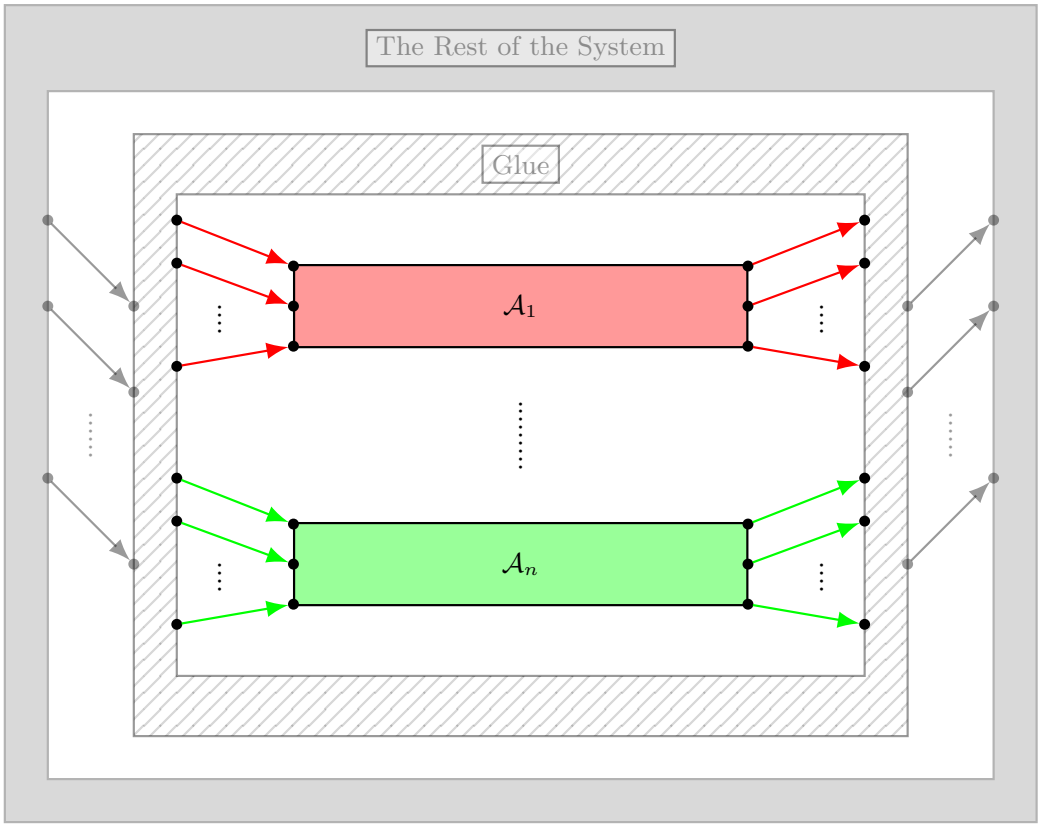
- Three Tier Symmetry
  - Easier Certification



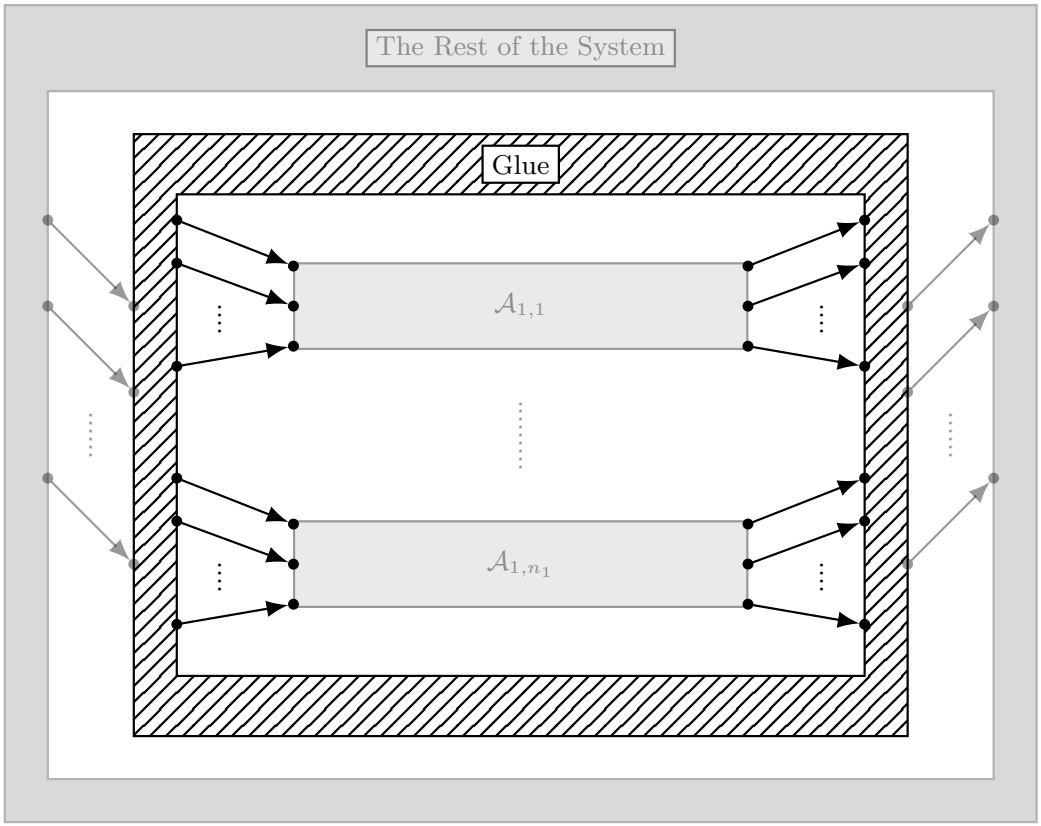
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations



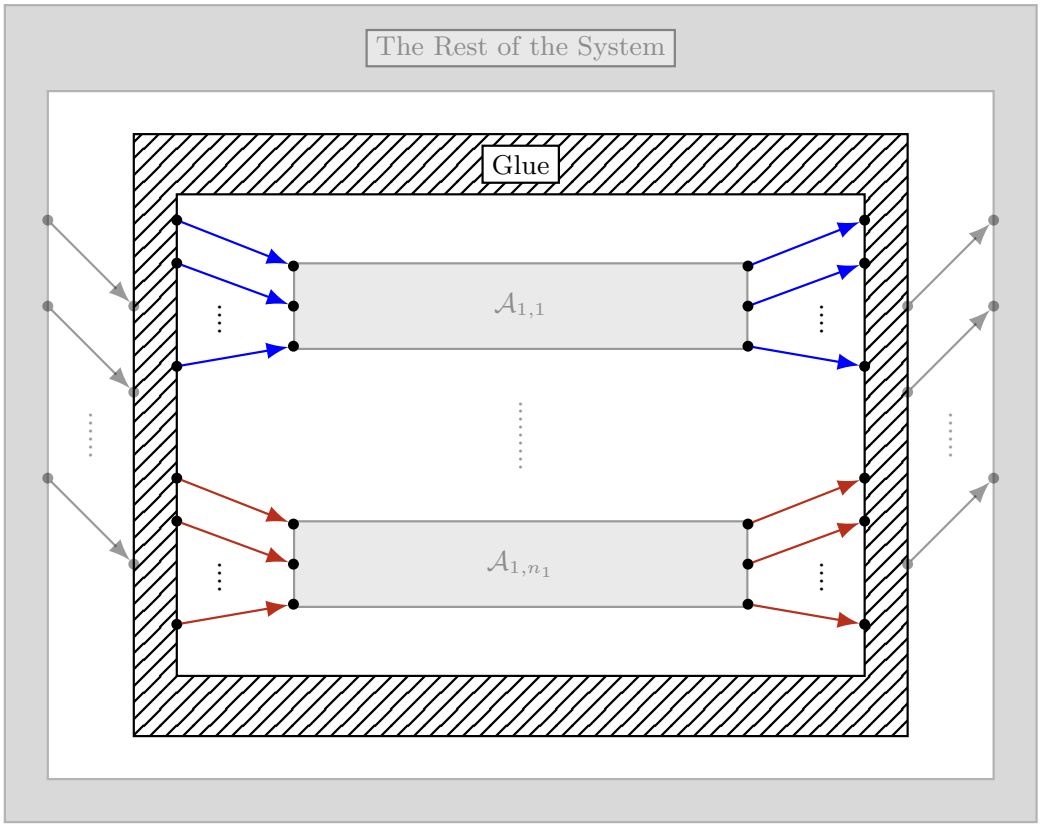
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations



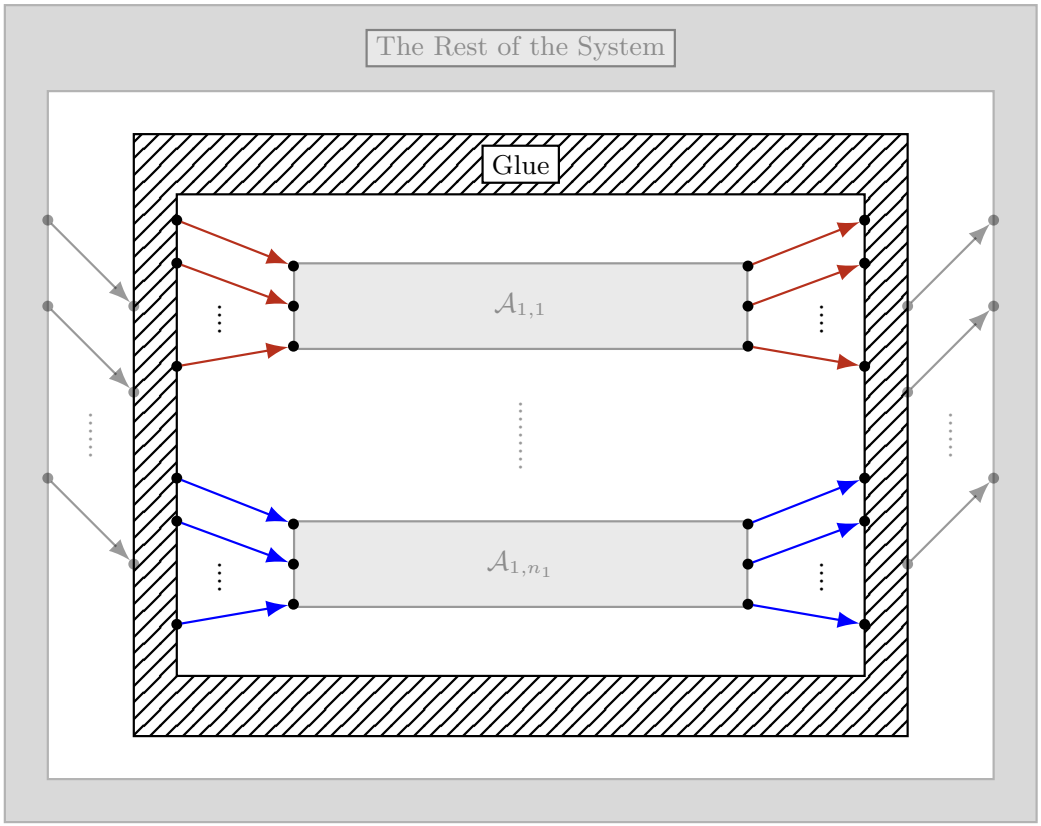
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations



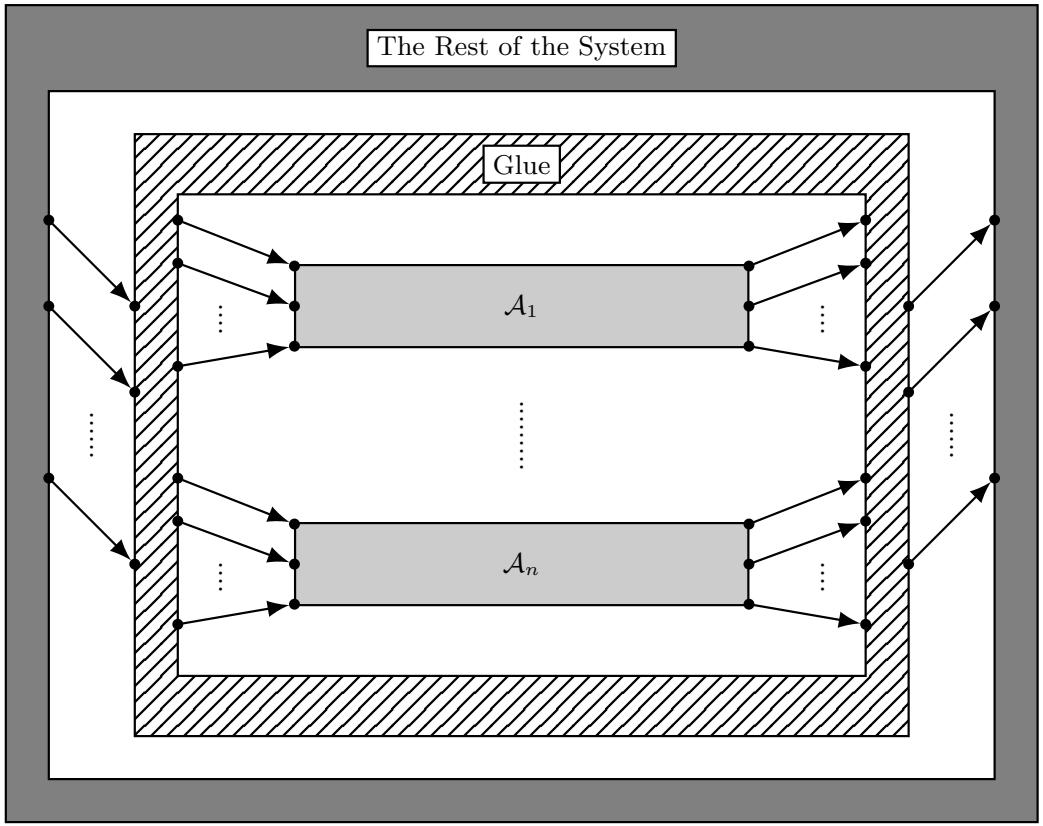
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations



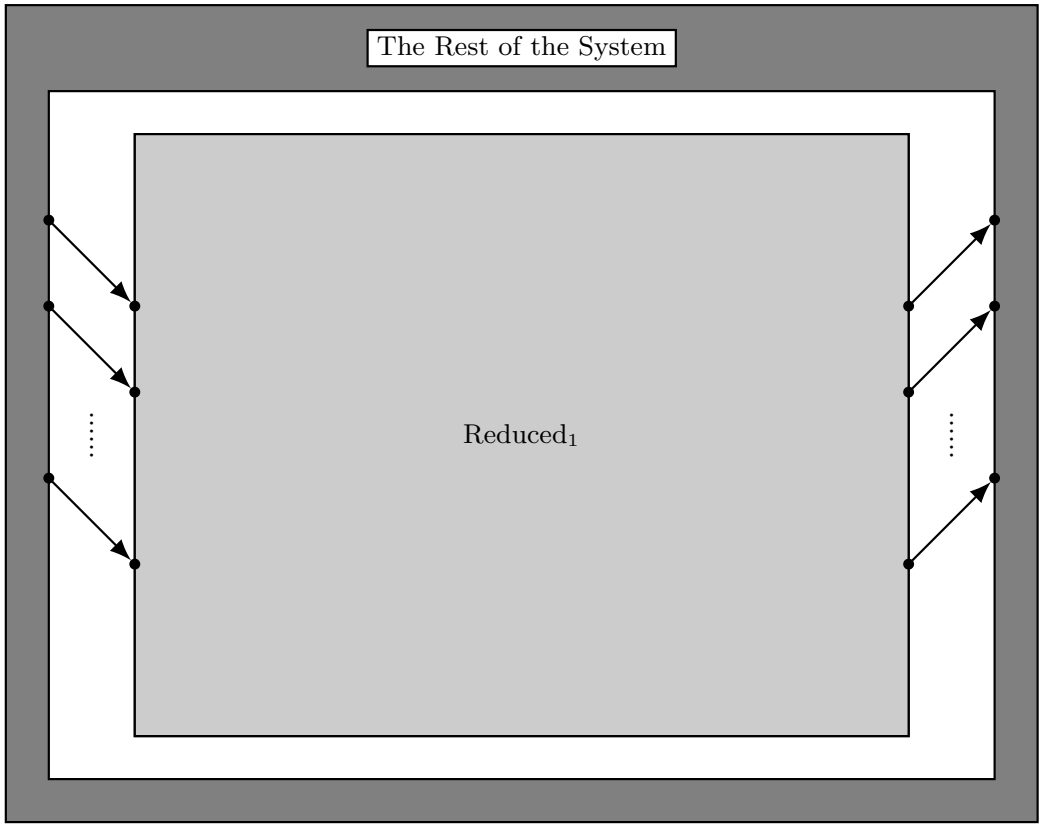
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations



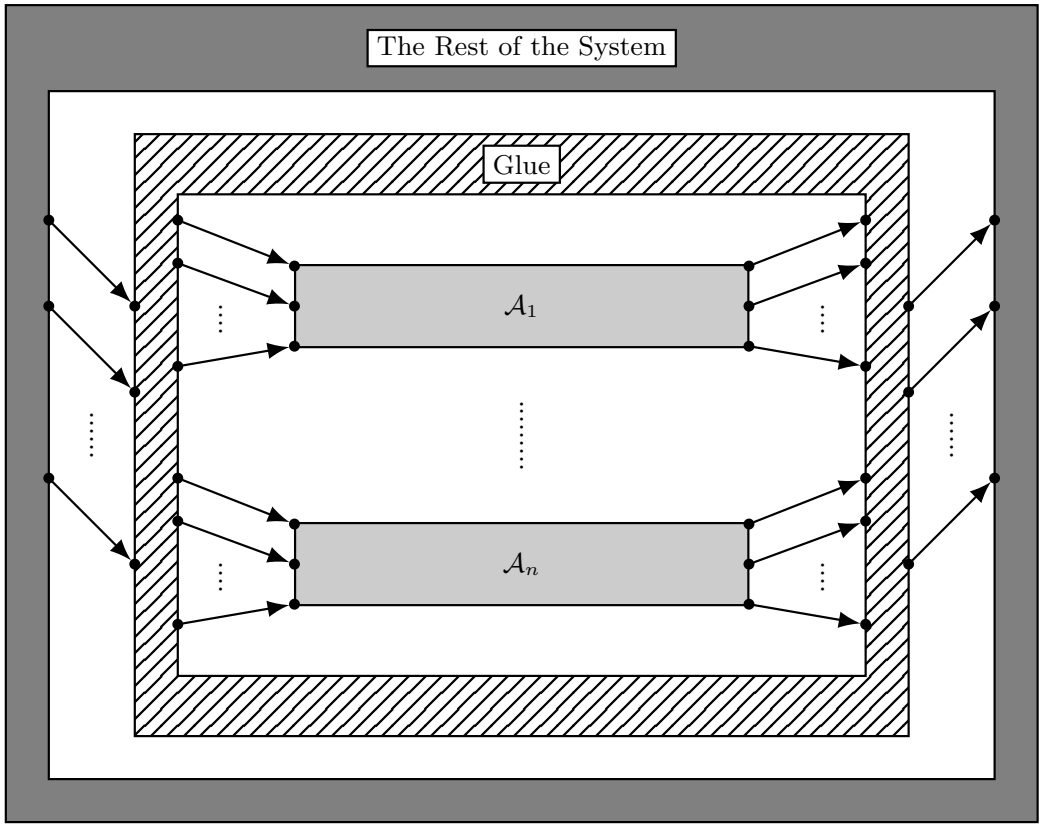
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction



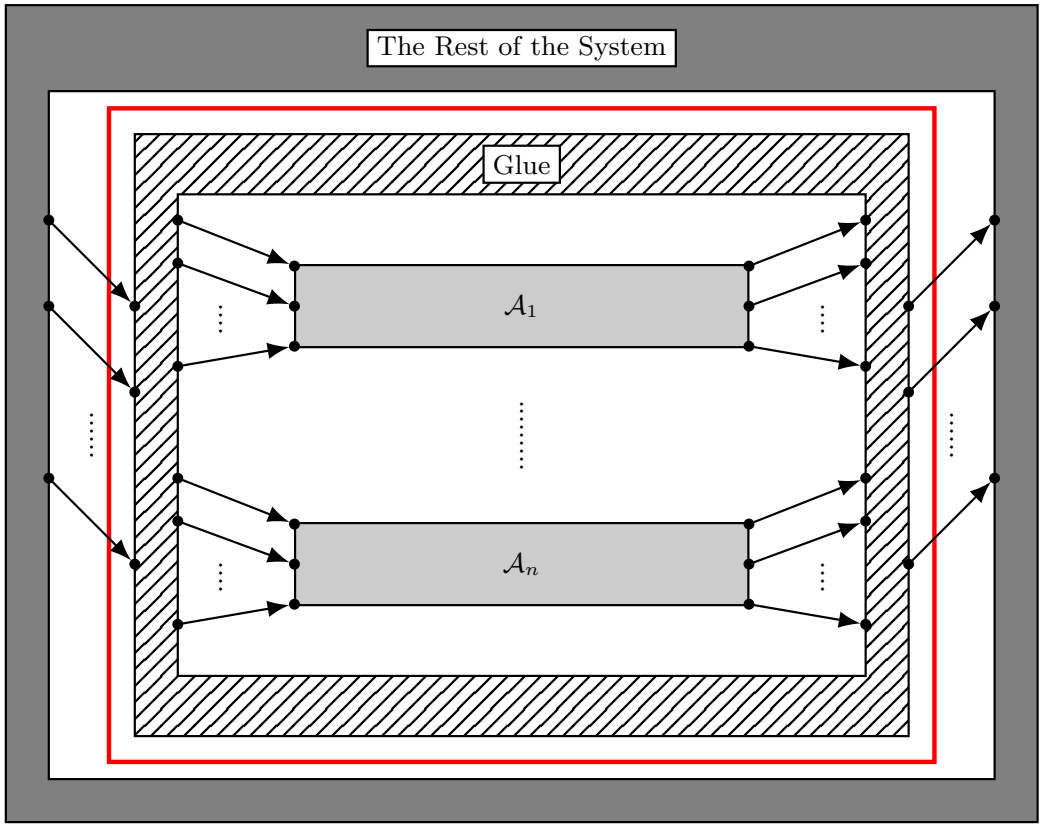
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion



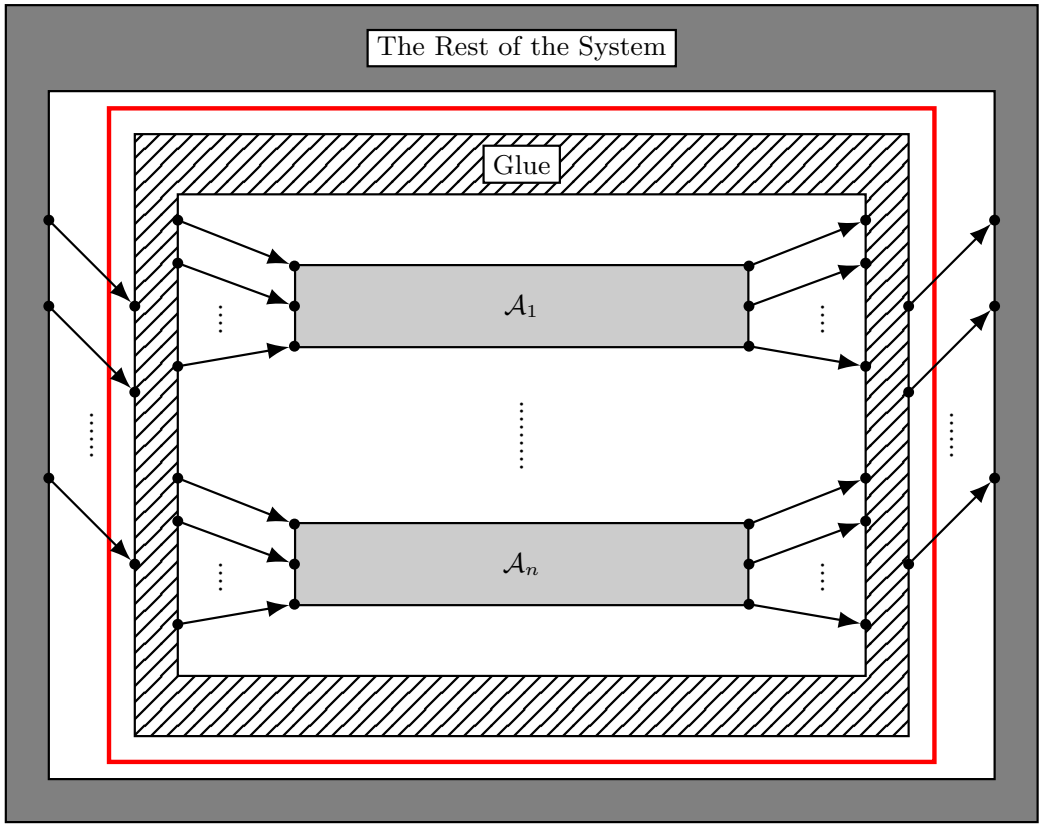
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion



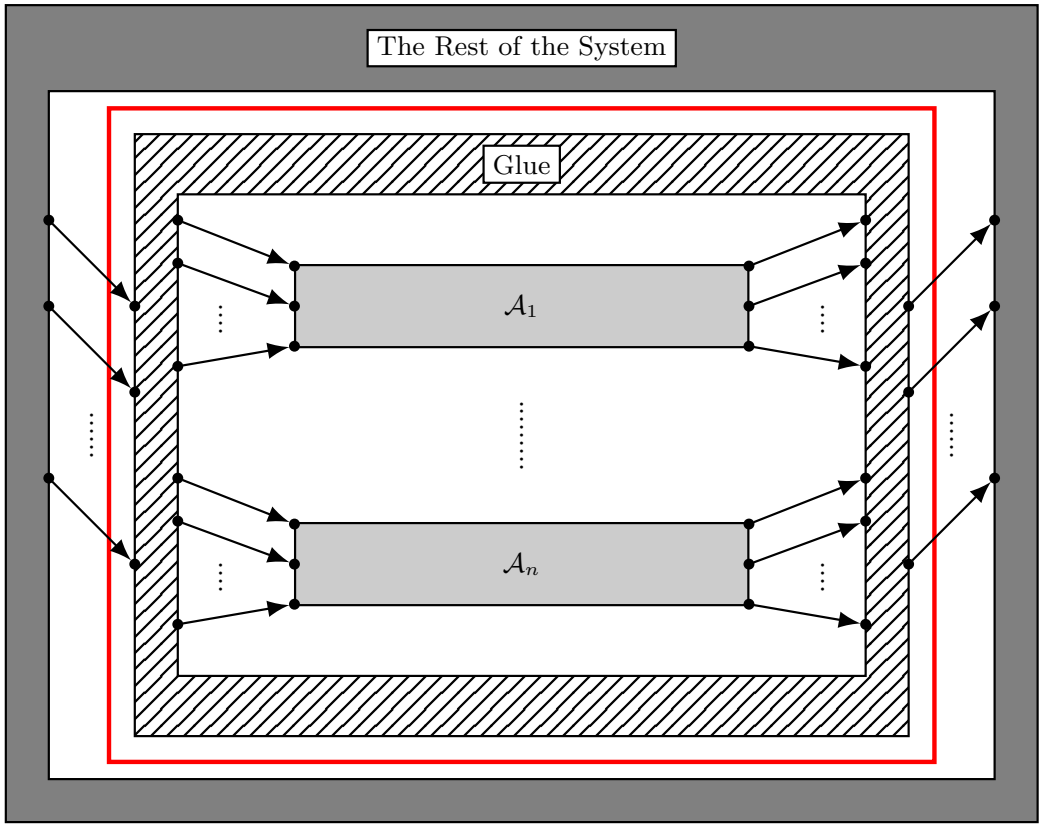
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion
- Symmetry Scenarios



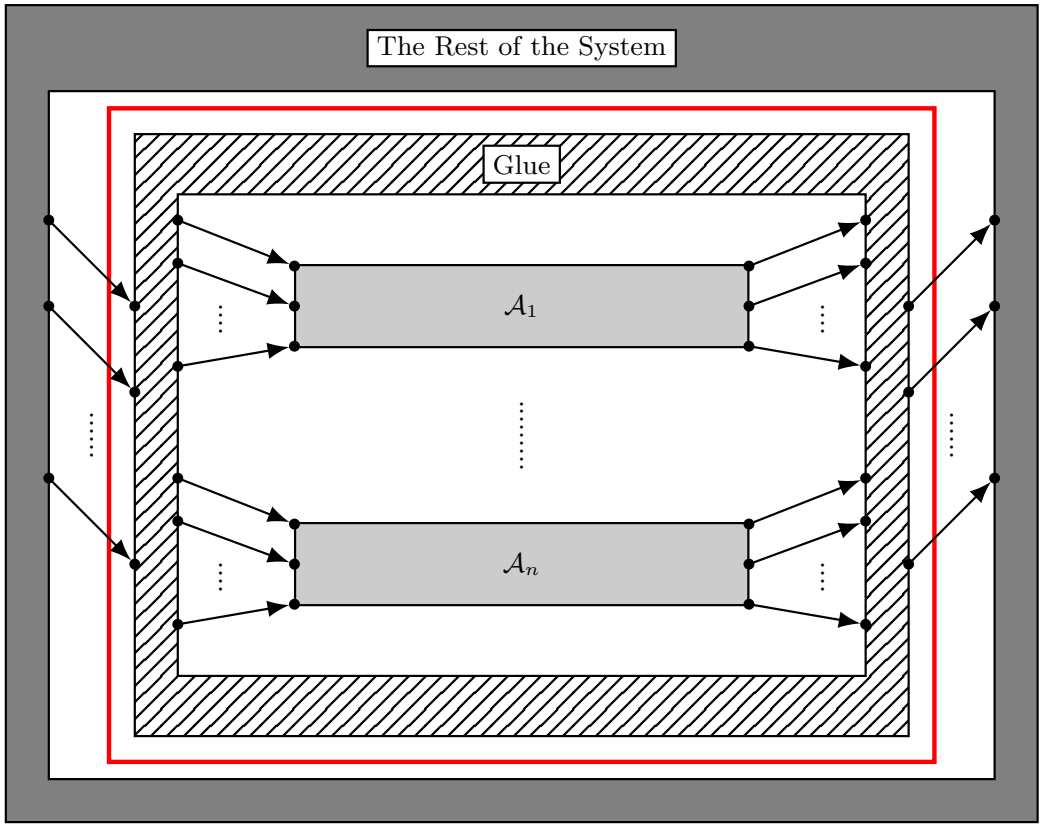
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion
- Symmetry Scenarios
  - A Reo circuit for glue and symmetric tier of each set of symmetric components



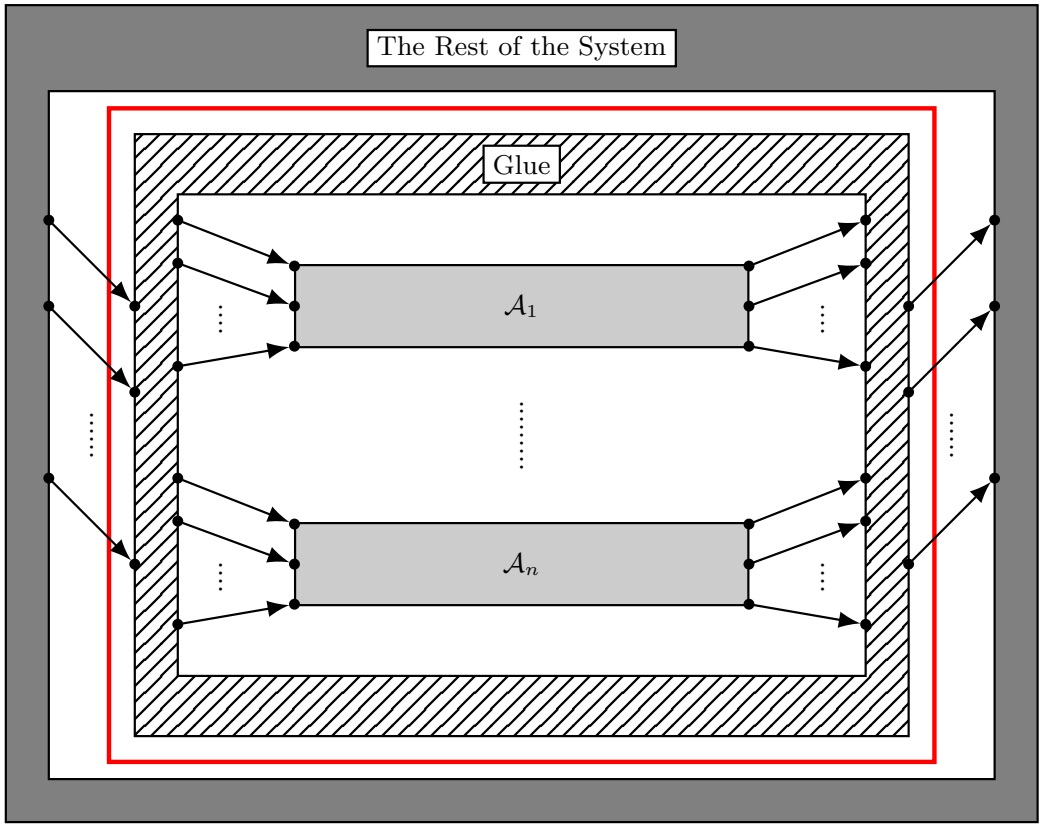
- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion
- Symmetry Scenarios
  - A Reo circuit for glue and symmetric tier of each set of symmetric components
  - Restrictions on symmetric component



- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion
- Symmetry Scenarios
  - A Reo circuit for glue and symmetric tier of each set of symmetric components
  - Restrictions on symmetric component
  - Guaranteed symmetry

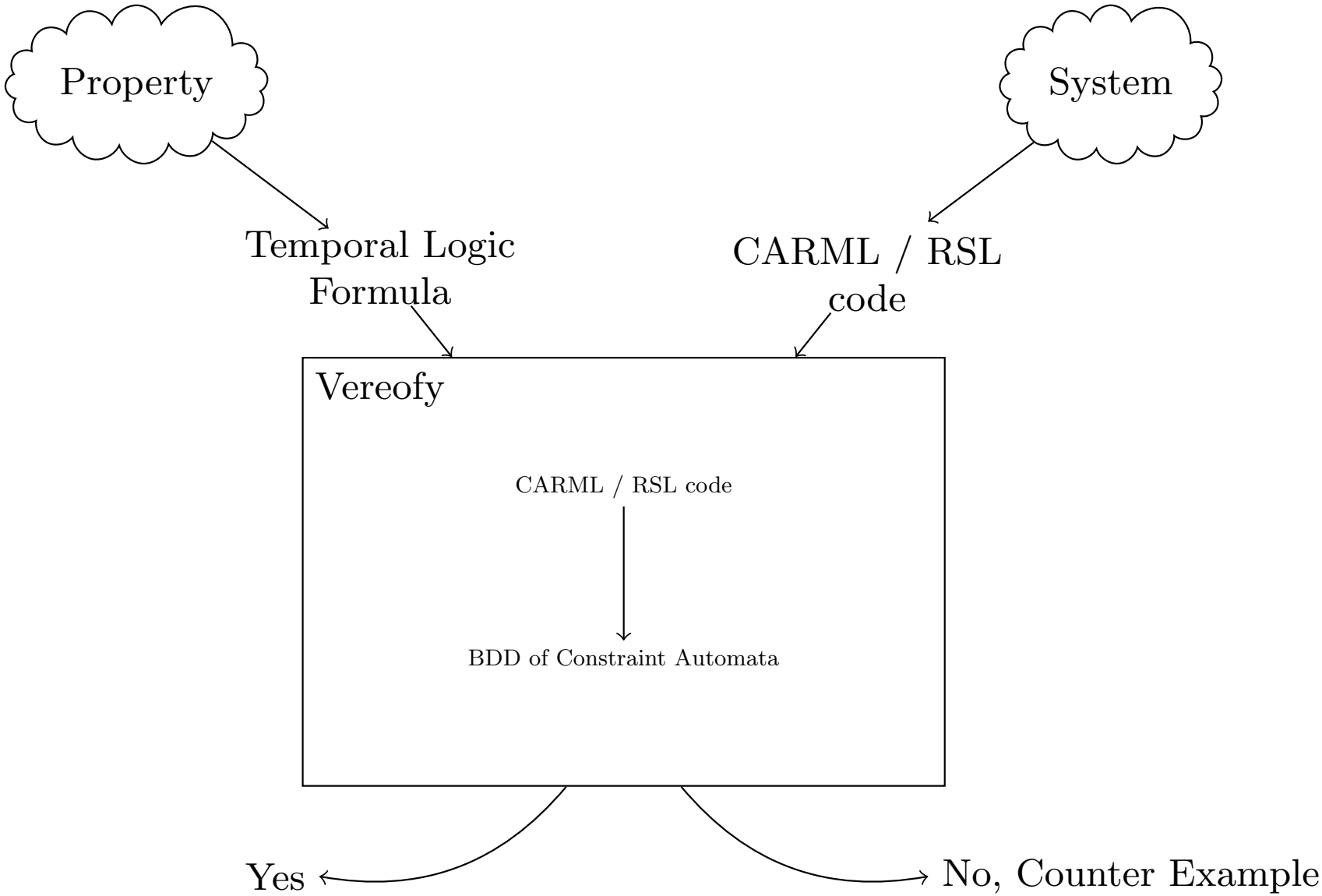


- Three Tier Symmetry
  - Easier Certification
    - Swapping states $\rightsquigarrow$ swapping data-flow locations
    - Symmetric if glue supports swaps of data-flow locations
  - Easier Reduction
    - Only reduce symmetric portion
- Symmetry Scenarios
  - A Reo circuit for glue and symmetric tier of each set of symmetric components
  - Restrictions on symmetric component
  - Guaranteed symmetry
  - Efficient and tailor-made reduction

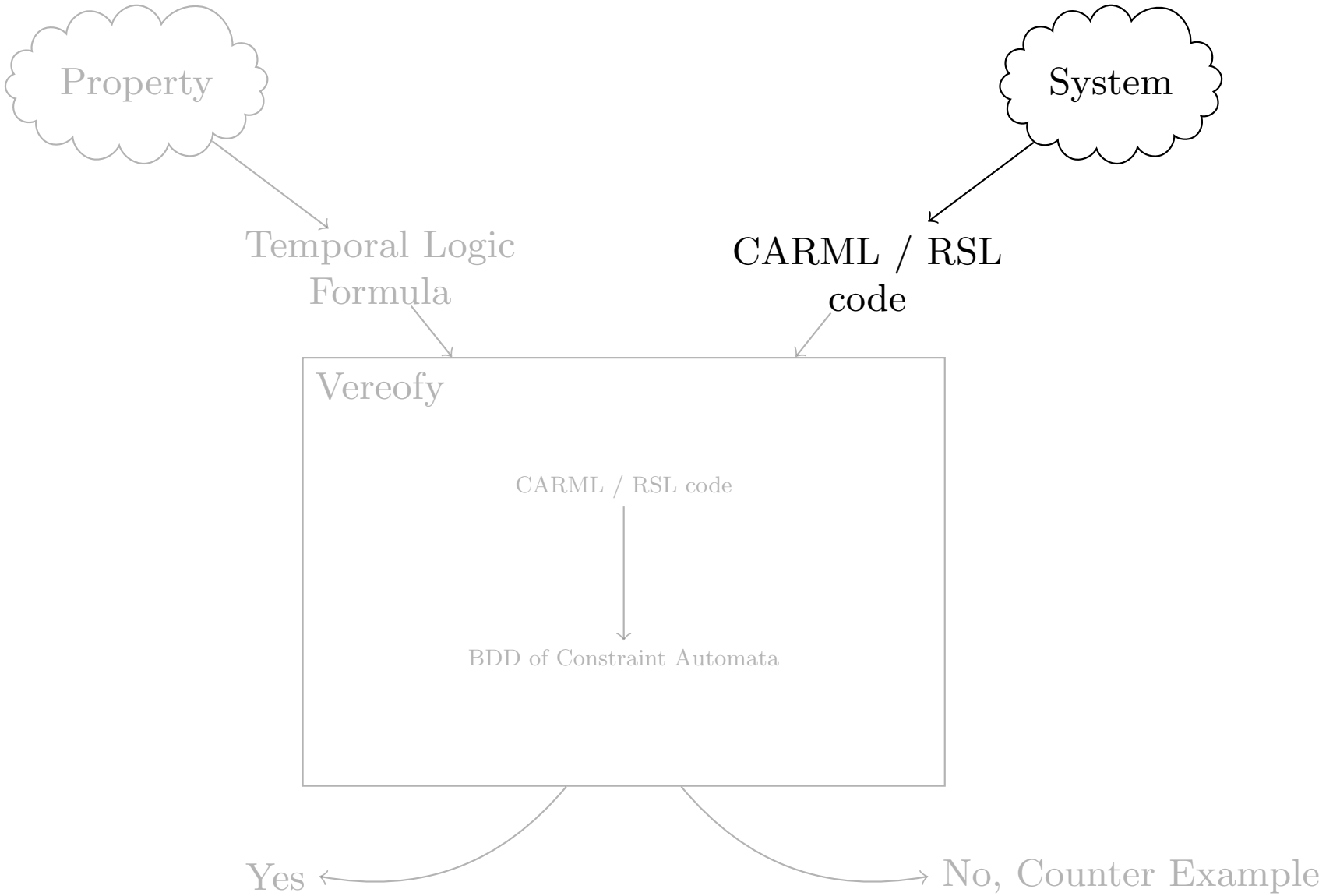


Outline

- Vereofy


Property
System
Temporal Logic Formula
CARML / RSL code
Vereofy
CARML / RSL code
BDD of Constraint Automata
Yes
No, Counter Example

- Vereofy


Property
System
Temporal Logic Formula
CARML / RSL code
Vereofy
CARML / RSL code
BDD of Constraint Automata
Yes
No, Counter Example

- Special Reo circuits for symmetry scenarios

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c
  - Writes to `file`

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c
  - Writes to `file`
- Counting to represent reduced state space

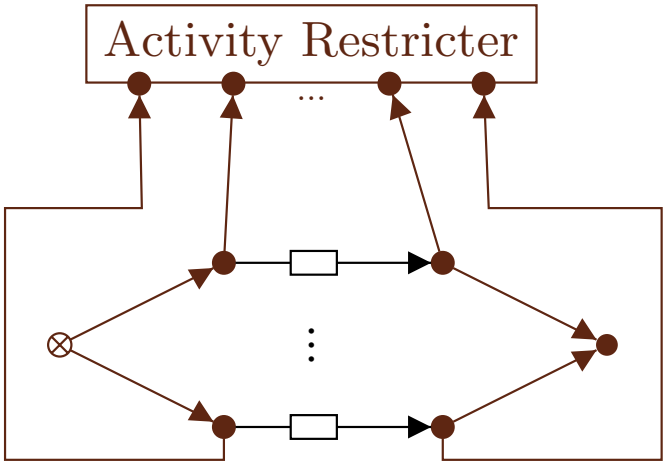
- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c
  - Writes to `file`
- Counting to represent reduced state space
- For simultaneous transitions – l Transition from q and k to q

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c
  - Writes to `file`
- Counting to represent reduced state space
- For simultaneous transitions – l Transition from q and k to q
  - Can fire in U, if $(q, i) \in U$ and $i \geq l$

- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c
  - Writes to `file`
- Counting to represent reduced state space
- For simultaneous transitions – l Transition from q and k to q
  - Can fire in U, if $(q, i) \in U$ and $i \geq l$
  - Results in V, where $(q, j) \in V$ and $j = i + k - l$

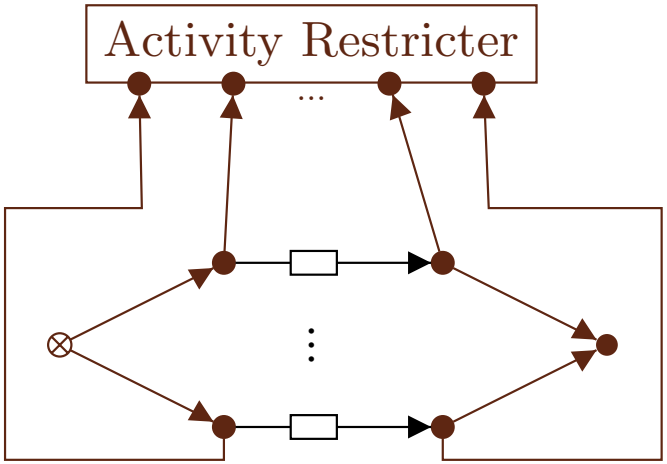
- Special Reo circuits for symmetry scenarios
- CARML to represent symmetric components
- Run Vereofy with command: `--process-symmetries="file"`
- Looks for instances of symmetry scenarios in input
- Symmetry scenario i with n copies of component c
  - Produce CARML code for scenario i of n copies of c
  - Writes to `file`
- Counting to represent reduced state space
- For simultaneous transitions – l Transition from q and k to q
  - Can fire in U, if $(q, i) \in U$ and $i \geq l$
  - Results in V, where $(q, j) \in V$ and $j = i + k - l$
- We considered 4 symmetry scenarios in implementation

- Symmetry Scenario 1



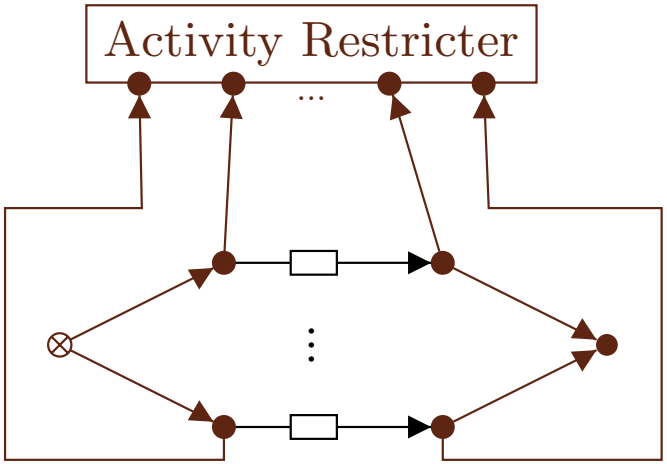
Glue and Symmetric Components

- Symmetry Scenario 1
  - Only one component can make a transition with data-flow



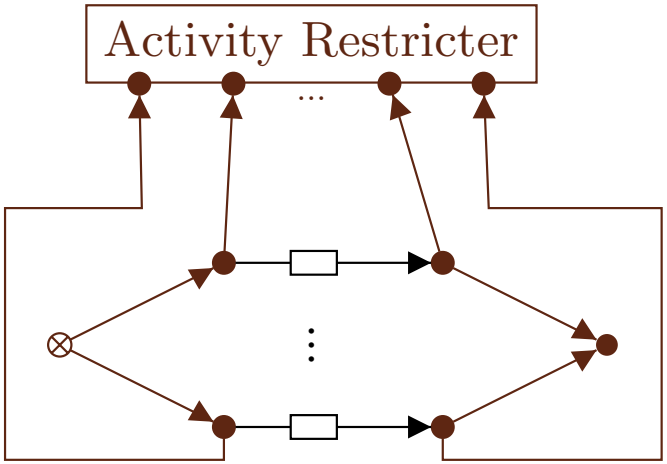
Glue and Symmetric Components

- Symmetry Scenario 1
  - Only one component can make a transition with data-flow
  - No epsilon transitions



Glue and Symmetric Components

- Symmetry Scenario 1
  - Only one component can make a transition with data-flow
  - No epsilon transitions
  - Symmetric component: FIFO channel



Glue and Symmetric Components

- Results for scenario 1
  - Before reduction

Copies	BDD Nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	78	16	0.006s	61
4	267	256	0.006s	189
10	1536	1.04858e+6	0.019s	626
20	4971	1.09951e+12	0.179s	1913
30	13306	1.15292e+18	0.492s	1583
50	36676	1.26765e+30	2.407s	2659
100	145851	1.60694e+60	20s	5358
150	327526	2.03704e+90	418s	–
500	–	–	–	–
2000	–	–	–	–

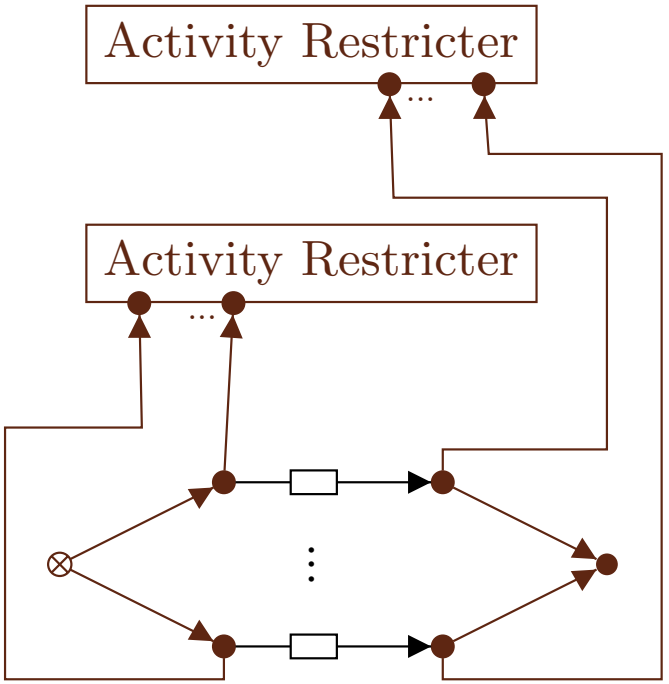
  - After reduction

Copies	Reduction Time	BDD Nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	0.002s	530	10	0.004s	97
4	0.001s	2187	35	0.005s	180
10	0.001s	4218	286	0.007s	448
20	0.001s	4093	1771	0.006s	479
30	0.001s	5202	5456	0.005s	411
50	0.001s	6186	23426	0.005s	617
100	0.001s	6061	176851	0.006s	3679
150	0.002s	8154	585276	0.006s	1079
500	0.002s	8029	2.10843e+7	0.007s	1887
2000	0.007s	9445	1.33734e+9	0.010s	1517

- Results for scenario 1
  - Bisimulation results

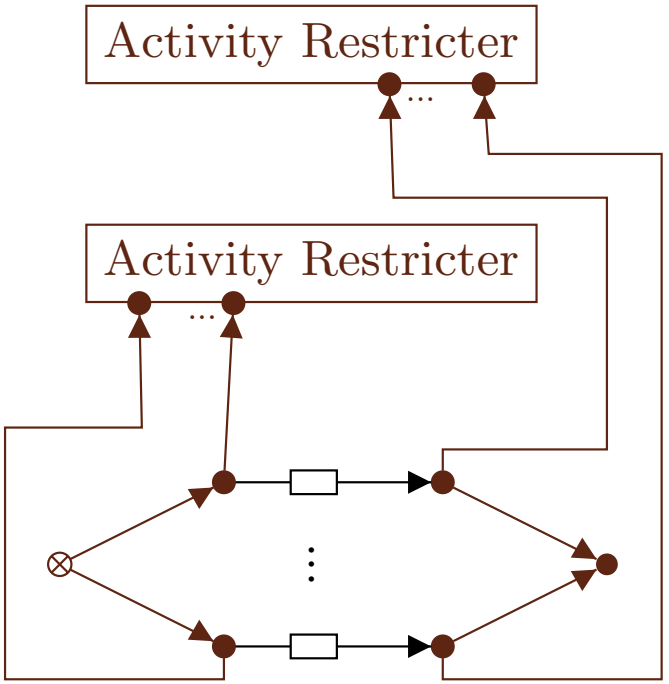
Copies	Time	Number of Classes
2	0.005s	6
4	0.02s	15
10	0.776s	66
20	40s	231
30	783s	496
50	–	–
150	–	–
500	–	–
2000	–	–

- Symmetry Scenario 2



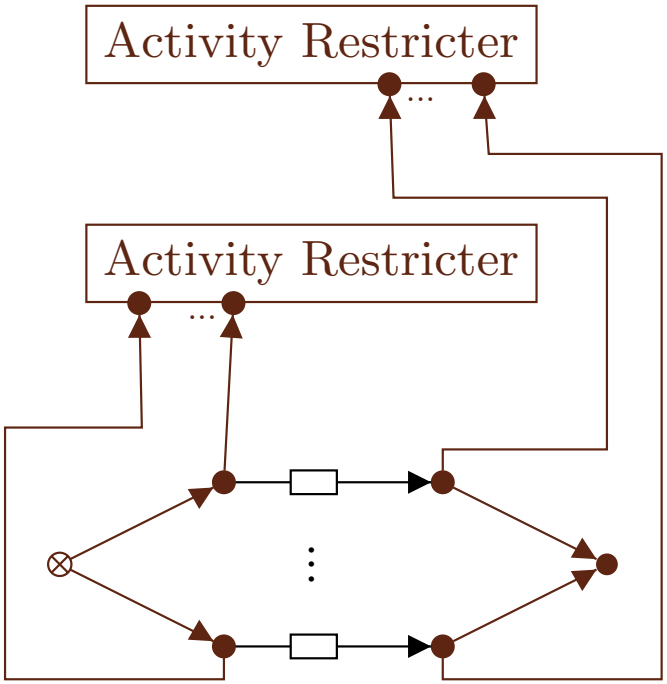
Glue and Symmetric Components

- Symmetry Scenario 2
  - At most one component can make an input transitions
  - At most one component can make an output transitions



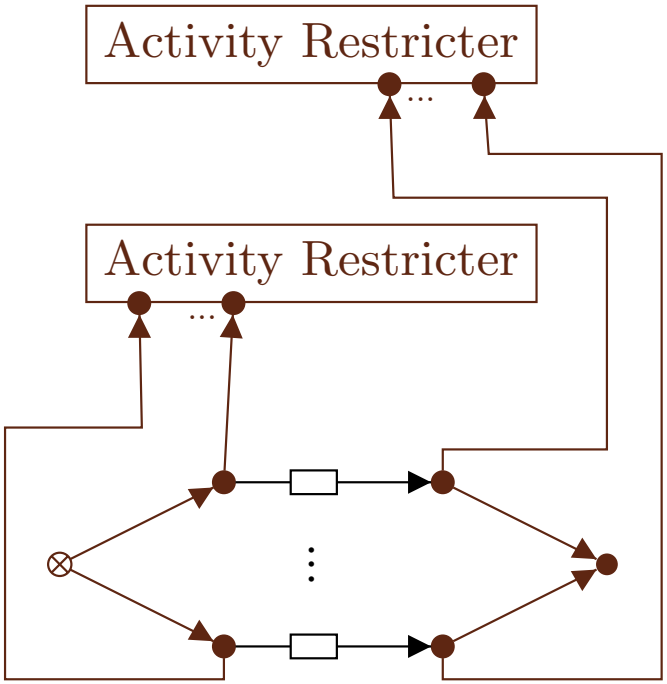
Glue and Symmetric Components

- Symmetry Scenario 2
  - At most one component can make an input transitions
  - At most one component can make an output transitions
  - These can be the same components



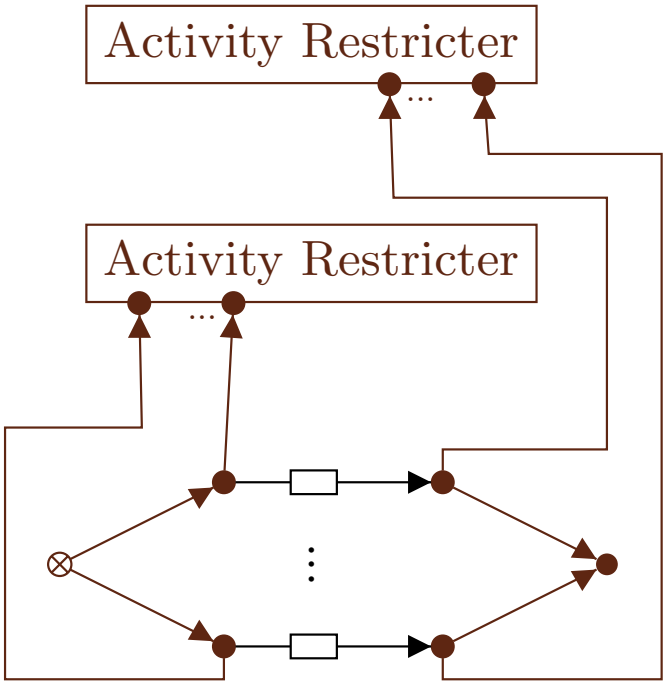
Glue and Symmetric Components

- Symmetry Scenario 2
  - At most one component can make an input transitions
  - At most one component can make an output transitions
  - These can be the same components
  - No epsilon transitions



Glue and Symmetric Components

- Symmetry Scenario 2
  - At most one component can make an input transitions
  - At most one component can make an output transitions
  - These can be the same components
  - No epsilon transitions
  - Symmetric component: FIFO channel



Glue and Symmetric Components

- Results for scenario 2
  - Before reduction

Copies	BDD Nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	168	16	0.005s	111
4	855	256	0.008s	445
10	10083	1.04858e+6	0.024s	1155
20	75223	1.09951e+12	0.336s	2413
30	249363	1.15292e+18	1.569s	2740
50	1140643	1.26765e+30	16s	4674
100	–	–	–	–
150	–	–	–	–
500	–	–	–	–
2000	–	–	–	–

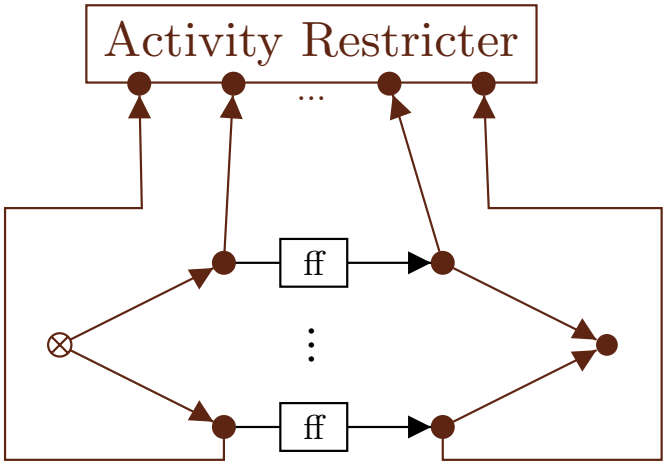
  - After reduction

Copies	Reduction Time	BDD Nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	0.002s	950	10	0.006s	144
4	0.002s	4185	35	0.008s	271
10	0.002s	10030	286	0.007s	434
20	0.002s	8250	1771	0.010s	593
30	0.002s	12186	5456	0.010s	663
50	0.002s	14342	23426	0.009s	798
100	0.003s	12562	176851	0.013s	951
150	0.009s	18654	585276	0.09s	1866
500	0.003s	16874	2.10843e+07	0.010s	1399
2000	0.002s	19602	1.33734e+09	0.014s	1658

- Results for scenario 2
  - Bisimulation results

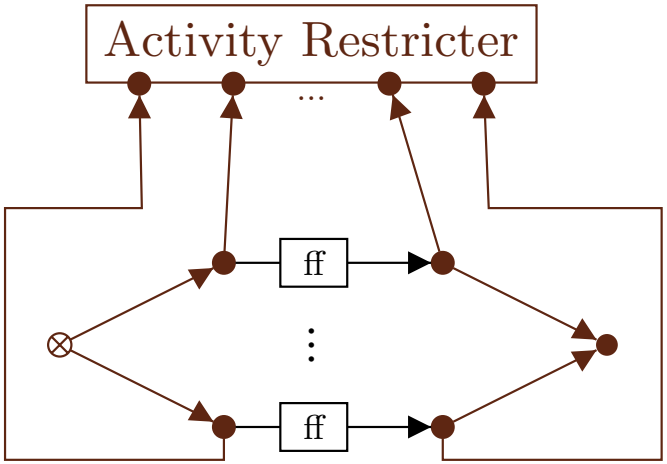
Copies	Time	Number of Classes
2	0.003s	6
4	0.03s	15
10	1.727s	66
20	153s	231
30	4134s	496
50	–	–
100	–	–
150	–	–
500	–	–
2000	–	–

- Symmetry Scenario 3



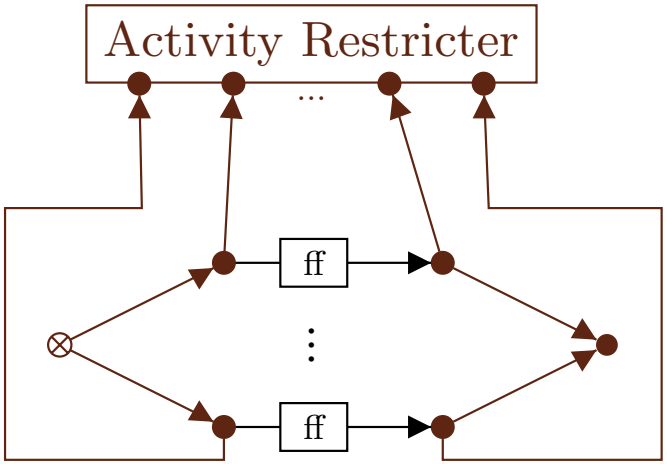
Glue and Symmetric Components

- Symmetry Scenario 3
  - Only one component can make a transition with data-flow



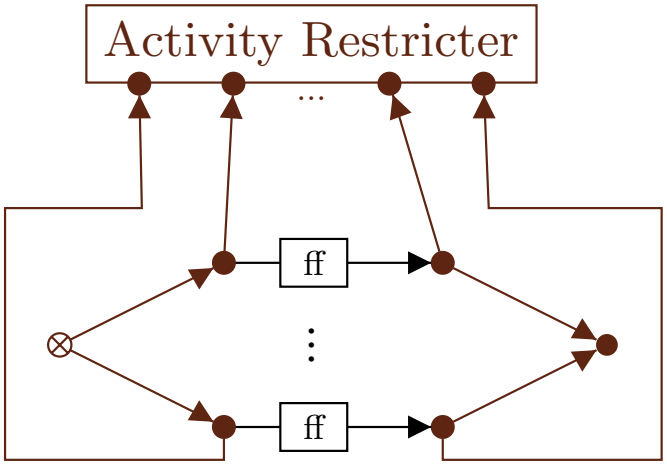
Glue and Symmetric Components

- Symmetry Scenario 3
  - Only one component can make a transition with data-flow
  - Epsilon transitions permitted



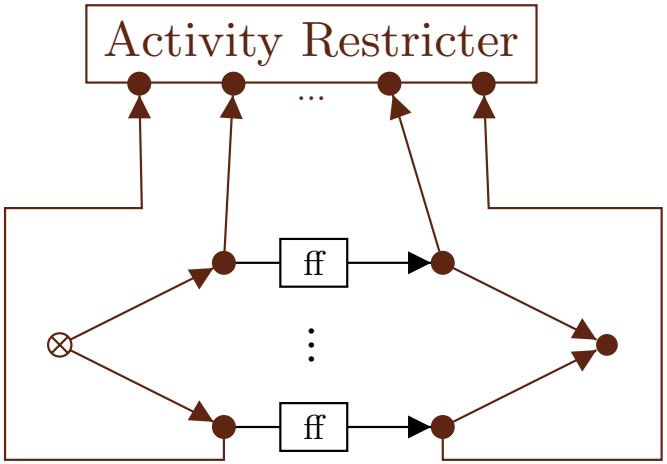
Glue and Symmetric Components

- Symmetry Scenario 3
  - Only one component can make a transition with data-flow
  - Epsilon transitions permitted
  - Exponential number of transitions in reduction



Glue and Symmetric Components

- Symmetry Scenario 3
  - Only one component can make a transition with data-flow
  - Epsilon transitions permitted
  - Exponential number of transitions in reduction
  - Symmetric component: forgetful FIFO channel



Glue and Symmetric Components

- Results for scenario 3
  - Before reduction

Copies	BDD nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	85	16	0.005s	75
4	272	256	0.005s	217
10	1433	1.04858e+06	0.028s	589
20	5368	1.09951e+12	0.028s	1449
30	11803	1.15292e+18	0.58s	1556
50	32173	1.26765e+30	3.777s	2606
100	126848	1.60694e+60	60s	–
150	–	–	–	–
200	–	–	–	–
250	–	–	–	–
300	–	–	–	–
350	–	–	–	–

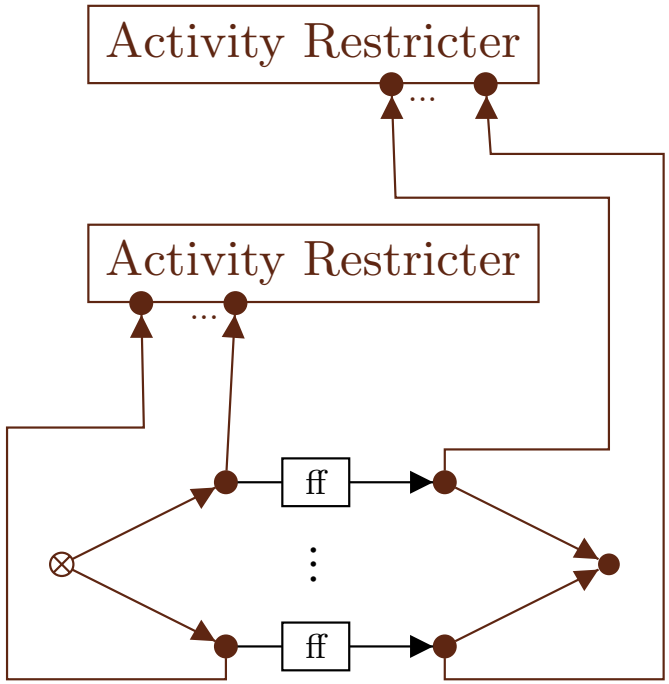
  - After reduction

Copies	Reduction Time	File Size of Reduced System	BDD Nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	0.001s	2KB	1024	10	0.008s	156
4	0.002s	7KB	7492	35	0.008s	738
10	0.005s	46KB	27820	286	3.310s	2041
20	0.022s	187KB	47383	1771	1.139s	12727
30	0.052s	426KB	46899	5456	3.489s	16904
50	0.002s	1.2MB	–	–	–	–
100	0.634s	4.8MB	–	–	–	–
150	1.444s	11MB	–	–	–	–
200	2.889s	19.7MB	–	–	–	–
250	4.751s	19.7MB	–	–	–	–
300	7.777s	44.7MB	–	–	–	–
350	408s	61.1MB	–	–	–	–

- Results for scenario 3
  - Bisimulation results

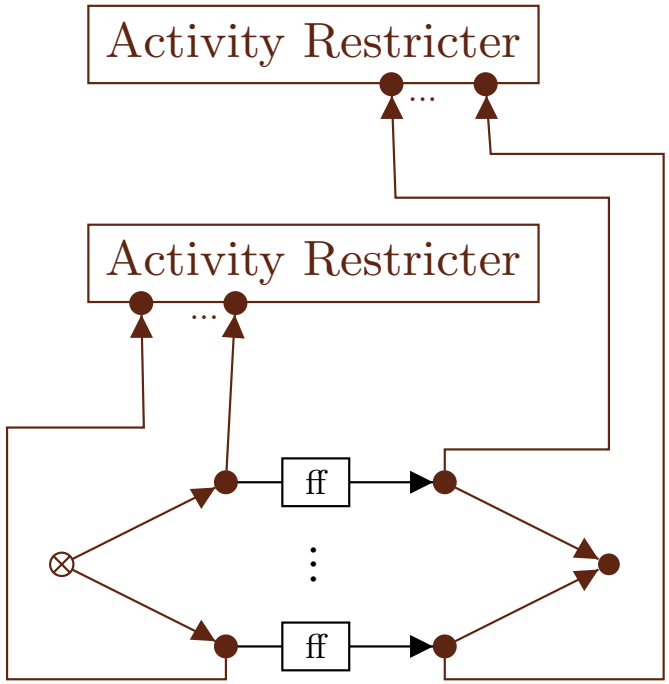
Copies	Time	Number of Classes
2	0.008s	6
4	0.054s	15
10	2.767s	66
20	250s	231
30	–	–
50	–	–
100	–	–
150	–	–
200	–	–
250	–	–
300	–	–
350	–	–

- Symmetry Scenario 4



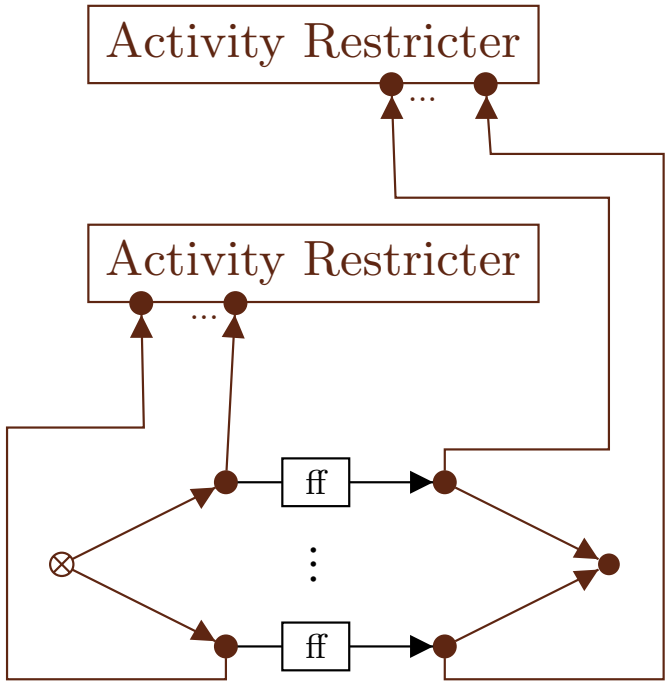
Glue and Symmetric Components

- Symmetry Scenario 4
  - At most one component can make in input transitions
  - At most one component can make in output transitions



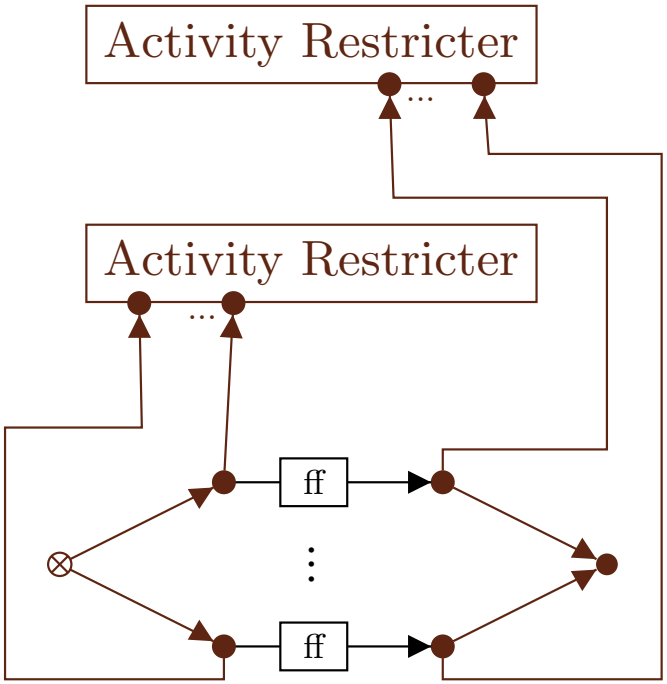
Glue and Symmetric Components

- Symmetry Scenario 4
  - At most one component can make in input transitions
  - At most one component can make in output transitions
  - These can be the same components



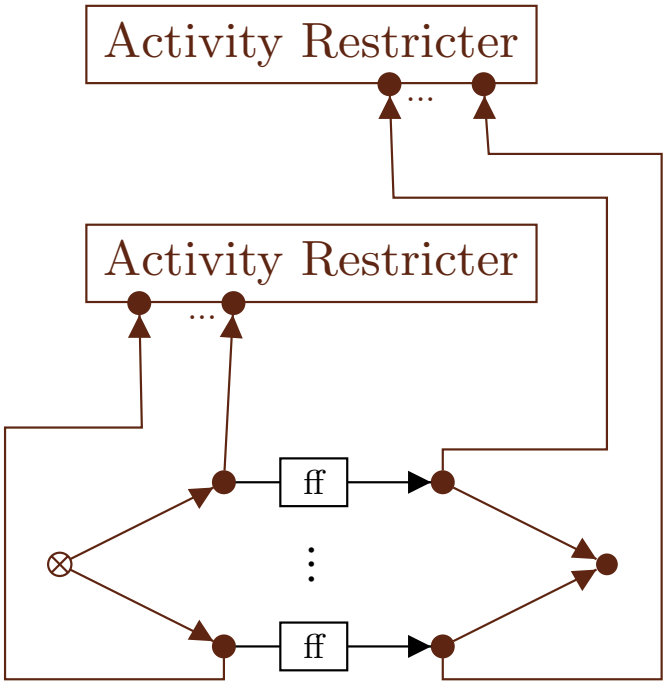
Glue and Symmetric Components

- Symmetry Scenario 4
  - At most one component can make in input transitions
  - At most one component can make in output transitions
  - These can be the same components
  - Epsilon transitions permitted



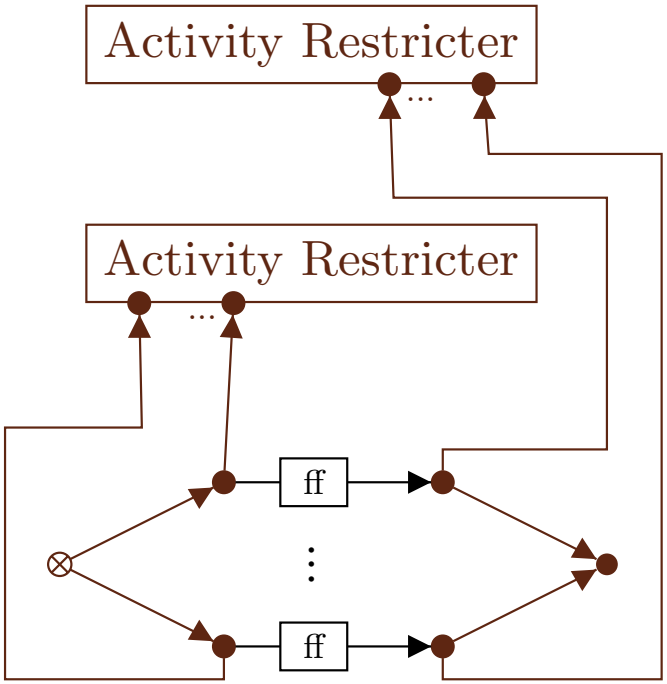
Glue and Symmetric Components

- Symmetry Scenario 4
  - At most one component can make in input transitions
  - At most one component can make in output transitions
  - These can be the same components
  - Epsilon transitions permitted
  - Exponential number of transitions in reduction



Glue and Symmetric Components

- Symmetry Scenario 4
  - At most one component can make in input transitions
  - At most one component can make in output transitions
  - These can be the same components
  - Epsilon transitions permitted
  - Exponential number of transitions in reduction
  - Symmetric component: forgetful FIFO channel



Glue and Symmetric Components

- Results for scenario 4
  - Before reduction

Copies	BDD nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	183	16	0.005s	114
4	833	256	0.006s	368
10	9157	1.04858e+06	0.006s	1003
20	66337	1.09951e+12	0.835s	2584
30	217517	1.15292e+18	3.734s	3752
50	985877	1.26765e+30	44s	4896
100	–	–	–	–
150	–	–	–	–
200	–	–	–	–

  - After reduction

Copies	Reduction Time	File Size of Reduced System	BDD nodes	Reachable States	Build Time	BDD Nodes After Dynamic Reorder
2	0.001s	4KB	1350	10	0.008s	170
4	0.004s	21KB	10562	35	0.091s	755
10	0.031s	176KB	38898	286	0.091s	2454
20	0.09s	783KB	–	–	–	–
30	0.193s	1.8MB	–	–	–	–
50	0.574s	5.3MB	–	–	–	–
100	2.5s	21.7MB	–	–	–	–
150	6.326s	49.4MB	–	–	–	–
200	216s	88.8MB	–	–	–	–

- Results for scenario 4
  - Bisimulation results

Copies	Time	Number of Classes
2	0.004s	6
4	0.062s	15
10	5.667s	66
20	–	–
30	–	–
50	–	–
100	–	–
150	–	–
200	–	–

Outline

- Conclusion

- Conclusion
  - Symmetry reduction to tackle state space explosion problem for constraint automata and Reo

- Conclusion
  - Symmetry reduction to tackle state space explosion problem for constraint automata and Reo
  - Preserved properties that do not depend on the order

- Conclusion
  - Symmetry reduction to tackle state space explosion problem for constraint automata and Reo
  - Preserved properties that do not depend on the order
  - Symmetry blockage

- Conclusion
  - Symmetry reduction to tackle state space explosion problem for constraint automata and Reo
  - Preserved properties that do not depend on the order
  - Symmetry blockage
  - Three tier symmetry

- Conclusion
  - Symmetry reduction to tackle state space explosion problem for constraint automata and Reo
  - Preserved properties that do not depend on the order
  - Symmetry blockage
  - Three tier symmetry
  - Outstanding performance in case of simple scenarios (scenarios 1 and 2)

- Conclusion
  - Symmetry reduction to tackle state space explosion problem for constraint automata and Reo
  - Preserved properties that do not depend on the order
  - Symmetry blockage
  - Three tier symmetry
  - Outstanding performance in case of simple scenarios (scenarios 1 and 2)
  - Poor performance for exponential number of transitions (scenarios 3 and 4), at least not for small number of copies

- Further Research

- Further Research
  - Different and more complicated scenarios

- Further Research
  - Different and more complicated scenarios
  - Symmetry reduction for non-symmetric properties

- Further Research
  - Different and more complicated scenarios
  - Symmetry reduction for non-symmetric properties
  - Symmetry detection form the structure of Reo circuits

- Further Research
  - Different and more complicated scenarios
  - Symmetry reduction for non-symmetric properties
  - Symmetry detection form the structure of Reo circuits
  - Slicing techniques to remove non-necessary symmetry breaking parts