CS780 Discrete-State Models

Instructor: Peter Kemper

R 006, phone 221-3462, email:kemper@cs.wm.edu

Office hours: Mon,Wed 3-5 pm

Today:

Some Example Bisimulations

References

Bisimulations for CCS

- R. Milner, Communication and Concurrency, Prentice Hall, 1989.

Inverse Bisimulation for Reachability

- P. Buchholz and P. Kemper. Efficient Computation and Representation of Large Reachability Sets for Composed Automata. Discrete Event Dynamic Systems - Theory and Applications (2002)

Bisimulation for Weighted Automata

- P. Buchholz, P. Kemper. *Weak Bisimulation for (max/+)-Automata and Related Models.* Journal of Automata, Languages and Combinatorics (2003)

Markov Chains, Lumpability

Many, many publications, a Phd that covers many aspects:

- S. Derisavi. Solution of Large Markov Models Using Lumping Techniques and Symbolic Data Structures. Doctoral Dissertation, University of Illinois, 2005.

  http://www.perform.csl.uiuc.edu/papers.html

Bisimulations

- Bisimulations are always defined in a similar manner
  - Examples: Strong and Weak Bisimulation, Observational Congruence, …

  Ingredients:
    - equivalence relations, largest is the interesting one
    - what the one state can do, the related one can simulate and vice versa

Definition (Strong bisimulation)

A relation $\rho \subseteq Prc \times Prc$ is called a strong bisimulation if $P \rho Q$ implies, for every $\alpha \in Act$,

1. $P \xrightarrow{\alpha} P' \implies$ ex. $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \rho Q'$
2. $Q \xrightarrow{\alpha} Q' \implies$ ex. $P' \in Prc$ such that $P \xrightarrow{\alpha} P'$ and $P' \rho Q'$

$P, Q \in Prc$ are called strongly bisimilar (notation: $P \sim Q$) if there exists a strong bisimulation ρ such that $P \rho Q$.

Inverse Bisimulation for Reachability

- Reduction of an Automaton

DEFINITION 3.1 *Let* $A = (S, \delta, s_0, L)$ *be an automaton and* $\mathscr{R}$ *be an equivalence relation on state space* S. *The aggregated automaton according to* $\mathscr{R}$ *is defined as* $A_{\mathscr{R}} = (\tilde{S}, \tilde{\delta}, \tilde{s}_0, \tilde{L})$, *in which* $\tilde{S} = S_{\mathscr{R}}$, $\tilde{s}_0$ *is the unique equivalence class with* $s_0 \in rep(\tilde{s}_0)$, $\tilde{L} = L$, *and* $\tilde{\delta}$ *is defined as follows:* $(\tilde{s}_x, \tilde{s}_y, l) \in \tilde{\delta} \Longleftrightarrow s_x \in rep(\tilde{s}_x)$ *and* $s_y \in rep(\tilde{s}_y)$ *with* $(s_x, s_y, l) \in \delta$ *exist.*

uses representative states.

- Weak Inverse Bisimulation

DEFINITION 3.2 *Let* $A = (S, \delta, s_0, L)$ *be an automaton and* $\mathscr{R}$ *be an equivalence relation on state space* S. $\mathscr{R}$ *is a weak inverse bisimulation* $\Longleftrightarrow 1) (s_0, s_x) \in \mathscr{R}$ *implies* $Q_{\tau *}(0, x)$ *and* 2) *if* $(s_x, s_y) \in \mathscr{R}$, *then* $Q_{l*}(z, x) = 1$ *implies* $Q_{l*}(z', y) = 1$ *for some* $s_{z'}$ *with* $(s_z, s_{z'}) \in \mathscr{R}$ *and vice versa.*

- Preserves reachability
- Let $Q_{\tau *} = \sum_{k=0}^{\infty} (Q_\tau)^k$ $\qquad Q_{l*} = Q_{\tau *} Q_l Q_{\tau *}$
- Inverse? Look for z, z′ position in $Q_{l*}(z, x) = 1$

Inverse Bisimulation for Reachability

- Weak Inverse Bisimulation preserves reachability

THEOREM 3.1 *If $\tilde{A}_{\mathscr{R}}$ results from automaton A by an aggregation with respect to some weak inverse bisimulation $\mathscr{R}$, then in every embedding environment the following relation holds:*

1. *if state $\tilde{s}_x \in \tilde{S}$ is reachable after $\tilde{A}$ is embedded, then all $s_x \in rep(\tilde{s}_x)$ are reachable after A is embedded in the same environment, and*

2. *if state $\tilde{s}_x \in \tilde{S}$ is not reachable after $\tilde{A}$ is embedded, then all $s_x \in rep(\tilde{s}_x)$ are not reachable after A is embedded in the same environment.*

- Embedding means parallel composition wrt to transition labels, i.e., synchronization of transitions.
- Proof:
  - Item 1: induction over number of synchronized transitions
    - 1st condition handles reachable states from s_0 before 1st synchronized transition
    - 2nd condition handles subsequent transitions
  - Item 2: follows from def of transitions in aggregated automaton

Weak bisimulation of K-automata (semiring)

An equivalence relation $R \subseteq S \times S$ is a weak bisimulation relation if for all $(s_1, s_2) \in R$, all $l \in L \setminus \{\tau\} \cup \{\varepsilon\}$, all equivalence classes $C \in S/R$

$$\alpha(s_1) = \alpha(s_2)$$
$$\beta'(s_1) = \beta'(s_2)$$
$$T'(s_1, l, C) = T'(s_2, l, C)$$

or in terms of matrices

$$\mathbf{a}(s_1) = \mathbf{a}(s_2)$$
$$\mathbf{b}'(s_1) = \mathbf{b}'(s_2)$$
$$\mathbf{M}'_l(s_1, C) = \mathbf{M}'_l(s_2, C)$$

Two states are weakly bisimilar, $s_1 \approx s_2$, if $(s_1, s_2) \in R$

Two automata are weakly bisimilar, $A_1 \approx A_2$, if there is a weak bisimulation on the union of both automata such that $\alpha(\mathrm{C}_1) = \alpha(\mathrm{C}_2)$ for all $C \in S/R$

Theorem

$$\text{If } A_1 \approx A_2 \text{ for Ki-Automata } A_1, A_2 \text{ then } w_1'(\sigma) = w_2'(\sigma)$$
$$\text{for all } \sigma \in L'^* \text{ where } L' = (L_1 \cup L_2) \setminus \{\tau\} \cup \{\varepsilon\}$$

Weights of sequences are equal in weakly bisimilar automata.

Ki ? commutative and idempotent semiring K

Sequence? sequence considers all paths that have same sequence of labels, may start or stop at any state

Weakly ? Paths can contain subpaths of τ-labeled transitions represented by a single ε-labeled transition.

Theorem

If $A_1 \approx A_2$ and A_3 are finite Ki - Automata then

1. $A_1 + A_3 \approx A_2 + A_3$ — direct sum

2. $A_1 \cdot A_3 \approx A_2 \cdot A_3$ and $A_2 \cdot A_3 \approx A_1 \cdot A_3$ — direct product

3. $A_1 \|_{L_C} A_3 \approx A_2 \|_{L_C} A_3$ and $A_3 \|_{L_C} A_1 \approx A_3 \|_{L_C} A_2$ — synchronized product

and if choice is defined then

4. $A_1 \vee A_3 \approx A_2 \vee A_3$ and $A_3 \vee A_1 \approx A_3 \vee A_2$ — choice

Some notes on proofs:

- proofs are lengthy,
- argumentation based matrices helps,
- argumentation along paths, resp. sequences more tedious
- idempotency simplifies valuation for concatenation of $\tau^* l\, \tau^*$ transitions
- note that algebra does not provide inverse elements wrt + and *

Lumping - Performance Bisimulation for Markov Chains

- Lumping
  - Markov Reward Process:
    Continuous Time Markov Chain with rate rewards
    and initial probabilities
  - Ordinary lumping, exact lumping
- Exploiting lumping at different levels
  - State-level lumping
  - Model-level lumping
  - Compositional lumping

Markov Reward Process (MRP)

- Various steady-state and transient measures can be computed using rate rewards and initial probabilities for states of CTMC
- MRP is 4-tuple $(\mathcal{S}, \mathbf{Q}, \mathbf{r}, \boldsymbol{\pi}^{\text{ini}})$
  - $\mathcal{S} = \{0, \ldots, |\mathcal{S}| - 1\}$: state space
  - $\mathbf{Q}^{(|\mathcal{S}| \times |\mathcal{S}|)}$: generator matrix
  - $\mathbf{r}(s)$: rate reward value of state $s \in \mathcal{S}$
  - $\boldsymbol{\pi}^{\text{ini}}(s)$: probability of state s at time 0
- Ordinary and exact lumping

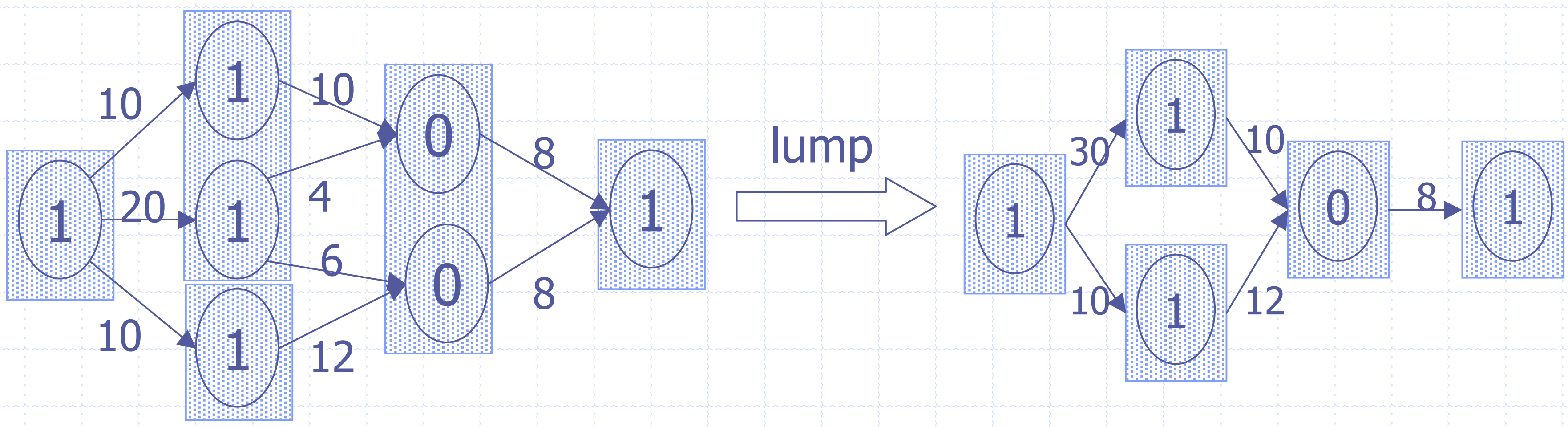
Ordinary Lumping

Definition: $M = (\mathcal{S}, \mathbf{Q}, \mathbf{r}, \boldsymbol{\pi}^{\text{ini}})$ is ordinarily lumpable w.r.t. to partition $\mathcal{P}$ of $\mathcal{S}$ iff

$\boxed{\mathbf{r}(s) = \mathbf{r}(\hat{s}) \text{ and } \sum_{s' \in C'} \mathbf{Q}(s, s') = \sum_{s' \in C'} \mathbf{Q}(\hat{s}, s')}$

for all (equivalence) classes C, C' of $\mathcal{P}$,

and all $s, \hat{s} \in C$

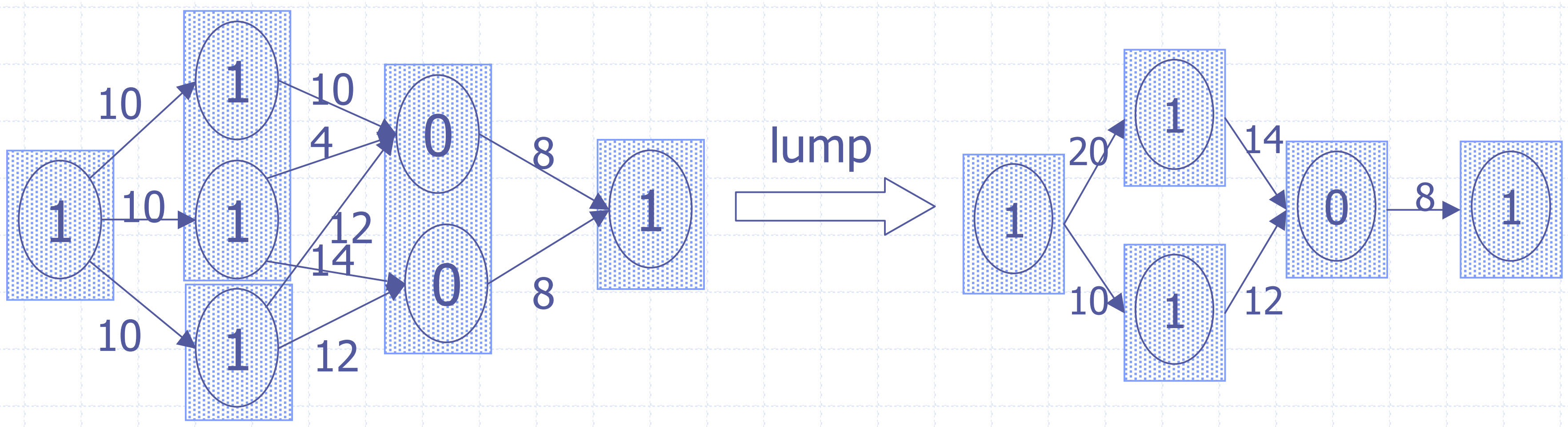


Exact Lumping

Definition: $M = (\mathcal{S}, \mathbf{Q}, \mathbf{r}, \boldsymbol{\pi}^{\text{ini}})$ is **exactly** lumpable w.r.t. to partition $\mathcal{P}$ of $\mathcal{S}$ iff

$$\boxed{\pi(s) = \pi(\hat{s}) \text{ and } \sum_{s' \in C'} \mathbf{Q}(s', s) = \sum_{s' \in C'} \mathbf{Q}(s', \hat{s})}$$

for all (equivalence) classes C, C' of $\mathcal{P}$,
and all $s, \hat{s} \in C$



Exact and ordinary lumping for DTMC

Definition 1. *Let* $\boldsymbol{P}$ *be the irreducible transition matrix of a finite Markov chain* X *on state space* Z *and* $\Omega = \{\Omega(1) \cdots \Omega(N)\}$ *a partition of the state space with collector matrix* $\boldsymbol{V}$.

- Ω *is ordinarily lumpable, iff for all* $I \in \{1 \cdots N\}$ *and all* $i, j \in \Omega(I)$: $(\boldsymbol{e}_i - \boldsymbol{e}_j)\boldsymbol{P}\boldsymbol{V} = \boldsymbol{0}$;
- Ω *is exactly lumpable, iff for all* $I \in \{1 \cdots N\}$ *and all* $i, j \in \Omega(I)$: $(\boldsymbol{e}_i - \boldsymbol{e}_j)\boldsymbol{P}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{0}$;
- Ω *is strictly lumpable, iff it is ordinarily and exactly lumpable.*

$\boldsymbol{e}_i$ *is a row vector with* 1.0 *in position* i *and* 0 *elsewhere.*

The above definition of ordinary/exact lumpability defines unique constants $\xi_{I,J} = \boldsymbol{e}_i \boldsymbol{P}_{I,J} \boldsymbol{e}^{\mathrm{T}}$ for ordinarily lumpable partitions and $\eta_{I,J} = \boldsymbol{e}\boldsymbol{P}_{I,J}\boldsymbol{e}_i^{\mathrm{T}}$ for exactly lumpable partitions, which are independent from $i \in \Omega(I)$.

Theorem 3 ([15], Section 5). *If* Ω *is an exactly lumpable partition on the state space* Z *of a finite Markov chain with transition matrix* $\boldsymbol{P}$ *and* $\hat{\boldsymbol{P}} = \boldsymbol{W}\boldsymbol{P}\boldsymbol{V}$ *is the transition matrix of the aggregated chain according to* Ω *and resulting from* $\boldsymbol{W} = \mathrm{diag}(\boldsymbol{e}\boldsymbol{V})^{-1}\boldsymbol{V}^{\mathrm{T}}$, *then* $\hat{\boldsymbol{\pi}} = \boldsymbol{\Pi}$ *and* $\boldsymbol{\pi}_I = \hat{\pi}(I)/n_I\boldsymbol{e}$. *The elements of* $\hat{\boldsymbol{P}}$ *are given by* $\hat{P}(I, J) = (n_J/n_I)\eta_{IJ}$.

Theorem 4 ([16], §3]). *If* Ω *is an ordinarily lumpable partition on the state space* Z *of a finite Markov chain with transition matrix* $\boldsymbol{P}$, *then* $\hat{\boldsymbol{P}} = \boldsymbol{W}\boldsymbol{P}\boldsymbol{V}$ *the transition matrix of the aggregated chain according to* Ω *is independent from the weight vector* $\boldsymbol{\alpha}$ *and* $\hat{\boldsymbol{\pi}} = \boldsymbol{\Pi}$. *The elements* $\hat{P}(I, J)$ *are equal to* $\xi_{I,J}$.

Exact and ordinary lumping

- Lumping works for both CTMCs and DTMCs
- Main motivation:
  - Solution of reduced MC yields smaller vector π which is the basis to compute rewards like utilization, throughput, population (e.g. in buffers), …
  - Exact lumping:
    - Detailed distribution inside equivalence class is known to be uniform
    - Reward measure may differ for different states in same equivalence class
  - Ordinary lumping:
    - Detailed distribution inside equivalence class is unknown
    - Reward measures can only be evaluated if they do not distinguish among states in same equivalence class
- Lumping can be a very effective reduction technique!

Types of Lumping Algorithms

- State-level lumping
  - First generate the overall CTMC, then lump
- Model-level lumping
  - Exploit symmetry among components and directly generate a lumped CTMC
- Compositional lumping
  - State-level lumping at component level
  - Often formalism-dependent
- All three types are complementary

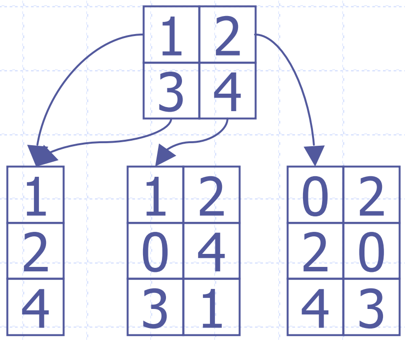
More Details

- Compositional lumping
  - Local and global equivalences for Matrix Diagrams
  - Compositional lumping theorem
  - Computation of local equivalence
  - Case study

Refresher: Matrix Diagram

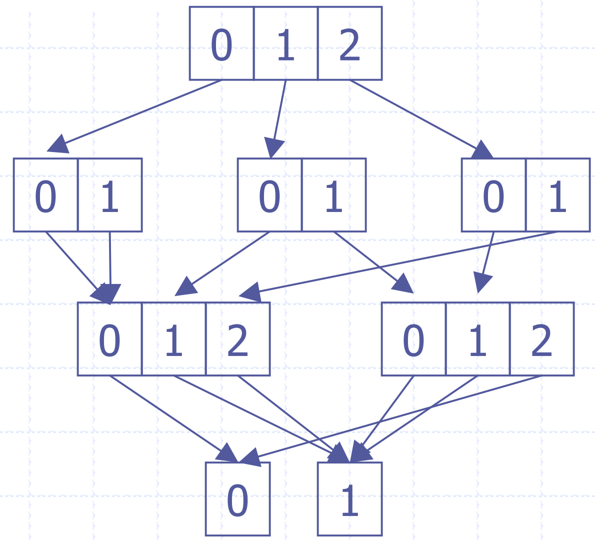
- Different elements multiplied by different matrices
- Generalization of Kronecker product
- Structurally similar to MDDs

  Multi-valued Decision Diagram
- May represent a supermatrix of the state transition rate matrix
  - Accompanied by state space represented as MDD
  - When projected on the MDD gives the exact state transition rate matrix



MDD: Refresher

- Represents function $f : \bigtimes_{i=1}^{m} S_i \rightarrow \{0, 1, \ldots, n\}$
  $S_i = \{0, \ldots, |S_i| - 1\}$
- Special case: n = 1
  - f represents a set of vectors

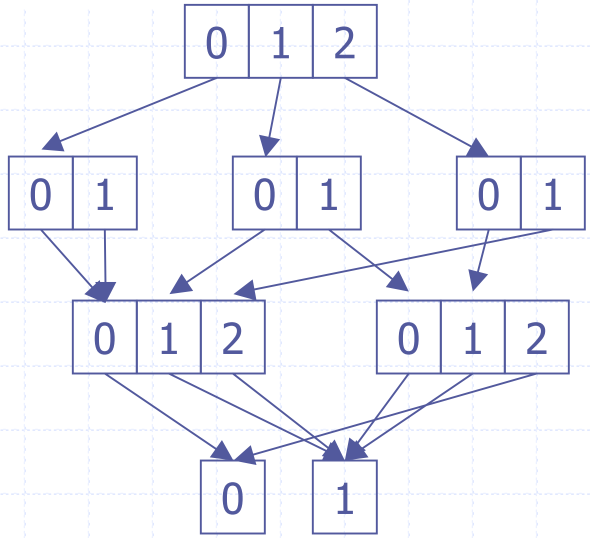


{(0,0,1), (0,0,2), (0,1,1), (0,1,2),
(1,0,1), (1,0,2), (1,1,0), (1,1,1),
(2,0,0), (2,0,1), (2,1,1), (2,1,2)}

MDD: Refresher

- Represents function $f : \bigtimes_{i=1}^{m} S_i \rightarrow \{0, 1, \ldots, n\}$

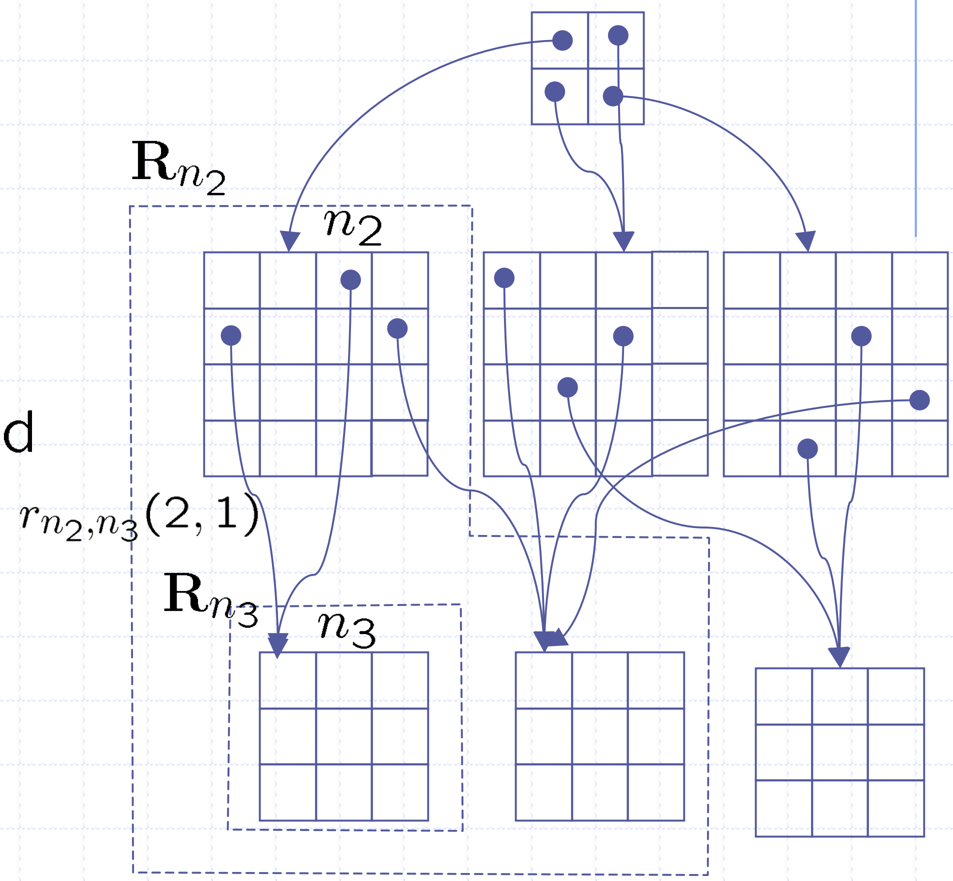
  $S_i = \{0, \ldots, |S_i| - 1\}$
- Special case: n = 1
  - f represents a set of vectors
  - Representation of a set of states of a discrete-state model
    - Partition set of state var.
    - Assign index to unique value assignment of variables of each block
    - Vector of indices represents a state



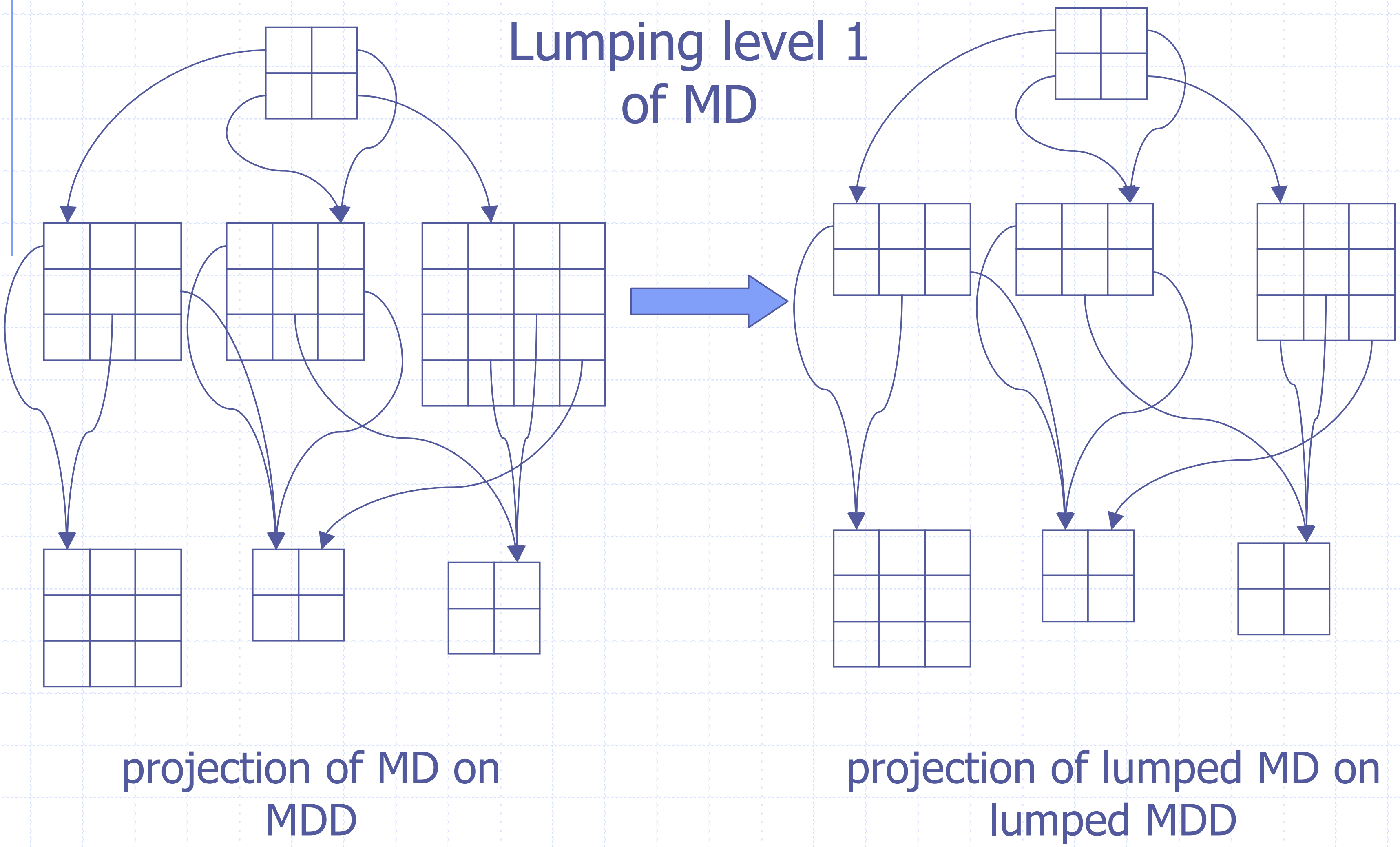
{(0,0,1), (0,0,2), (0,1,1), (0,1,2), (1,0,1), (1,0,2), (1,1,0), (1,1,1), (2,0,0), (2,0,1), (2,1,1), (2,1,2)}

MD Notation

- N_c: nodes of level c
- $n_c \in N_c$: a node ID in level c
- $\mathbf{R}_{n_c}$: matrix represented by node n_c
- $r_{n_c,n_{c+1}}(s_c, s'_c)$: real number in element (s_c, s'_c) of node n_c that points to n_{c+1}



Goal: Compositional Lumping at individual levels



Simplified Notation

- Consider level c of MD for lumping conditions
  - All levels above/below c can be merged into one level
- Without loss of generality:
  - Discussing 3-level MD and focusing on level 2 instead of discussing m-level MD and focusing on level c
  - Makes notation and main concepts straightforward to understand and theorems easier to prove
- State represented as vector of substates, i.e.,

$s = (s_1, s_2, s_3)$ where $s \in \mathcal{S}$, $s_c \in \mathcal{S}_c$, and $c \in \{1, 2, 3\}$

$$\mathbf{r}(s) = g(f_1(s_1), f_2(s_2), f_3(s_3))$$

Local and Global Equivalences

- Local equivalence $\approx_{lo}$ (partition $\mathcal{P}_{lo}$) on $\mathcal{S}_2$

  $s_2 \approx_{lo} \hat{s}_2$ if $\forall n_2 \in N_2, C'_2 \in \mathcal{P}_{lo}$

$$f_2(s_2) = f_2(\hat{s}_2) \text{ and}$$

$$\sum_{s'_2 \in C'_2} \mathbf{R}_{n_2}(s_2, s'_2) = \sum_{s'_2 \in C'_2} \mathbf{R}_{n_2}(\hat{s}_2, s'_2)$$

$$\boxed{\begin{array}{l} \mathbf{r}(s) = \mathbf{r}(\hat{s}) \text{ and} \\ \sum_{s' \in C'} \mathbf{Q}(s, s') = \sum_{s' \in C'} \mathbf{Q}(\hat{s}, s') \end{array}}$$

- Global equivalence $\approx_{go}$: Extending $\approx_{lo}$ to complete state space $\mathcal{S}$:

  $s \approx_{go} \hat{s}$ if $s_1 = \hat{s}_1$, $s_2 \approx_{lo} \hat{s}_2$, and $s_3 = \hat{s}_3$

Compositional Lumping Theorem

- **Theorem**: CTMC represented by MD is ordinarily lumpable with respect to global equivalence $\approx_{go}$
  - Therefore, $\approx_{lo}$ satisfies sufficient conditions on matrices of one level of MD such that it gives ordinary lumpability for overall MD
  - Theorem holds for any local and global equivalences stricter than $\approx_{lo}$ and $\approx_{go}$
  - Similar sufficient conditions and theorem for exact lumpability

Computation of Local Equivalence (1)

- Should satisfy

$$f_2(s_2) = f_2(\hat{s}_2) \text{ and}$$
$$\sum_{s'_2 \in C'_2} \mathbf{R}_{n_2}(s_2, s'_2) = \sum_{s'_2 \in C'_2} \mathbf{R}_{n_2}(\hat{s}_2, s'_2)$$

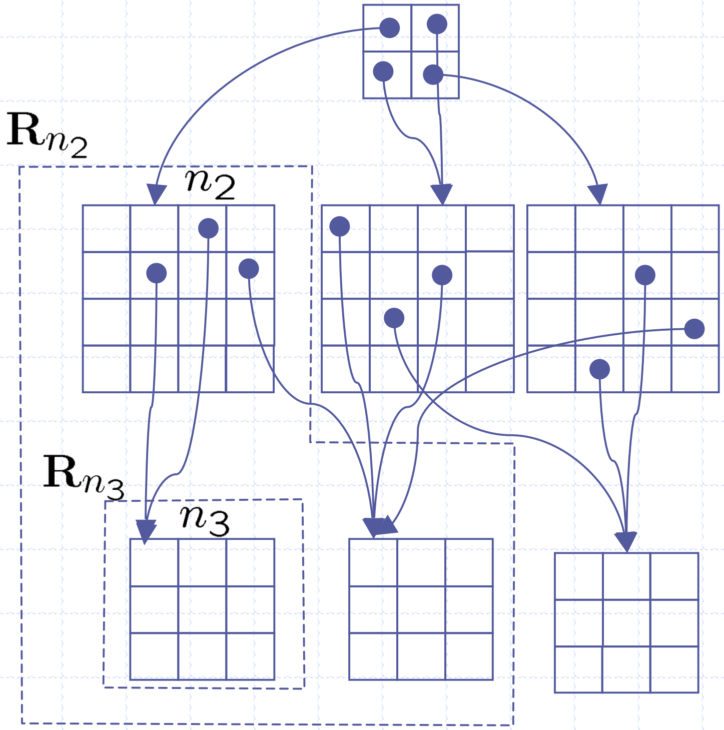
- Testing the second condition involves matrices of size $|\mathcal{S}_2 \times \mathcal{S}_3| \times |\mathcal{S}_2 \times \mathcal{S}_3|$
$\Rightarrow$ computationally too expensive

- Resort to easier-to-compute stricter (i.e., sufficient) condition

Computation of Local Equivalence (2)

- $$\sum_{s_2' \in C_2'} \mathbf{R}_{n_2}(s_2, s_2') = \sum_{s_2' \in C_2'} \mathbf{R}_{n_2}(\hat{s}_2, s_2') \text{ iff}$$

$$\sum_{s_2' \in C_2'} \sum_{n_3 \in N_3} r_{n_2,n_3}(s_2, s_2')\mathbf{R}_{n_3}(s_3, s_3') = \sum_{s_2' \in C_2'} \sum_{n_3 \in N_3} r_{n_2,n_3}(\hat{s}_2, s_2')\mathbf{R}_{n_3}(s_3, s_3')$$



Computation of Local Equivalence (3)

$$\sum_{s_2'\in C_2}\sum_{n_3\in N_3} r_{n_2,n_3}(s_2,s_2')\mathbf{R}_{n_3}(s_3,s_3') = \sum_{s_2'\in C_2}\sum_{n_3\in N_3} r_{n_2,n_3}(\widehat{s}_2,s_2')\mathbf{R}_{n_3}(s_3,s_3')$$

$$\text{iff} \quad \sum_{n_3\in N_3}\left[\sum_{s_2'\in C_2} r_{n_2,n_3}(s_2,s_2')\right]\mathbf{R}_{n_3}(s_3,s_3') = \sum_{n_3\in N_3}\left[\sum_{s_2'\in C_2} r_{n_2,n_3}(\widehat{s}_2,s_2')\right]\mathbf{R}_{n_3}(s_3,s_3') \qquad (1)$$

- (1) holds if $\sum_{s_2'\in C_2} r_{n_2,n_3}(s_2,s_2') = \sum_{s_2'\in C_2} r_{n_2,n_3}(\widehat{s}_2,s_2')$
  - Equality testing involves $|\mathcal{S}_2|\times|\mathcal{S}_2|$ matrices, i.e., local conditions

Compositional: Performance Study

- Tandem network
  - Jobs are served in two phases
    - MSMQ polling-based system (4 queues, 3 servers)
    - Hypercube multiprocessor
  - 3-level MD and MDD

J	SS size					
	unlumped			lumped		
	$\lvert\mathcal{S}_1\rvert$	$\lvert\mathcal{S}_2\rvert$	$\lvert\mathcal{S}_3\rvert$	$\lvert\tilde{\mathcal{S}}_1\rvert$	$\lvert\tilde{\mathcal{S}}_2\rvert$	$\lvert\tilde{\mathcal{S}}_3\rvert$
1	2	650	160	2	30	40
2	3	3,575	700	3	178	175
3	4	14,300	2,220	4	803	555

J	overall SS size		# of MD nodes	generation time (s)	lumping time (s)
	unlumped	lumped			
1	22,100	395	7	0.05	0.04
2	197,600	4,075	10	0.8	0.26
3	1,236,300	28,090	13	12.1	1.8

Conclusion

- State-level lumping
  - Suffers from handling very large state spaces, matrices
- Model-level lumping
  - Various options, formalism dependent
    - Stochastic Well-formed Nets (SWNs)
    - Mobius Rep/Join and Graph Composed models
    - Superposed GSPNs
- Compositional lumping
  - Based on congruence:
    - Automata with parallel composition
      - PEPA, Superposed GSPNs, …
  - Based on symbolic matrix representation

    Work by S. Derisavi …