

Modeling and Analysis of Markov Chains Using Decision Diagrams

Gianfranco Ciardo

**Department of Computer Science and Engineering
University of California at Riverside
Riverside, CA 92521, USA
ciardo@cs.ucr.edu**

(partially based on work with Andrew S. Miner and Paul L. E. Grieco)



- Background on CTMCs
 - Continuous-time Markov chains and high-level CTMC models
 - Matrix-vector description and operations for CTMC solution
 - Sparse matrices
- Structured models
 - High-level CTMC models structured into submodels
 - State space and state indexing: potential vs. actual indices
 - Multiway decision diagrams (MDDs)
- Decision-diagram-based CTMC encodings
 - Multiterminal multiway decision diagrams (MTMDDs)
 - Kronecker descriptors
 - Matrix diagrams (MXDs)
 - Edge-valued multiway decision diagrams (EVMDDs)
- Solution algorithms
 - Vector-matrix multiplication with Kronecker-based encodings
 - Jacobi vs. Gauss-Seidel style iterations

Background on CTMCs

Discrete-state models

A *discrete state model* is fully specified by:

- a *potential state space* $\widehat{\mathcal{S}}$ the “type” of the states
- a set of *initial states* $\mathcal{S}^{init} \subseteq \widehat{\mathcal{S}}$ often there is a single initial state \mathbf{s}^{init}
- a *next-state function* $\mathcal{N} : \widehat{\mathcal{S}} \rightarrow 2^{\widehat{\mathcal{S}}}$ naturally extended to sets: $\mathcal{N}(\mathcal{X}) = \bigcup_{\mathbf{i} \in \mathcal{X}} \mathcal{N}(\mathbf{i})$

The *state space* \mathcal{S} of the model is the smallest set containing \mathcal{S}^{init} and satisfying:

- the *recursive definition* $\mathbf{i} \in \mathcal{S} \wedge \mathbf{j} \in \mathcal{N}(\mathbf{i}) \Rightarrow \mathbf{j} \in \mathcal{S}$
- or the *fixed-point equation* $\mathcal{X} = \mathcal{X} \cup \mathcal{N}(\mathcal{X})$

$$\mathcal{S} = \mathcal{S}^{init} \cup \mathcal{N}(\mathcal{S}^{init}) \cup \mathcal{N}^2(\mathcal{S}^{init}) \cup \mathcal{N}^3(\mathcal{S}^{init}) \cup \dots = \mathcal{N}^*(\mathcal{S}^{init})$$

Definition of continuous-time Markov chain

A stochastic process $\{X(t) : t \geq 0\}$ is a collection of r.v.'s indexed by a *time parameter* t

We say that $X(t)$ is the *state* of the process at time t

The possible values $X(t)$ can ever assume for any t is (a subset of) the *state space* \mathcal{S}

$\{X(t) : t \geq 0\}$ over a *discrete* \mathcal{S} is a *continuous-time Markov chain (CTMC)* if

$$\begin{aligned} \Pr\{X(t_{n+1}) = i_{n+1} \mid X(t_n) = i_n \wedge X(t_{n-1}) = i_{n-1} \wedge \dots \wedge X(t_0) = i_0\} \\ = \Pr\{X(t_{n+1}) = i_{n+1} \mid X(t_n) = i_n\} \end{aligned}$$

for any times $0 \leq t_0 \leq \dots \leq t_{n-1} \leq t_n \leq t_{n+1}$ and states $\{i_0, \dots, i_{n-1}, i_n, j_{n+1}\} \subseteq \mathcal{S}$

Markov property:

“given the present state, the future is independent of the past”

“the most recent knowledge about the state is all we need”

Markov chain description and analysis

A *continuous-time Markov chain (CTMC)* $\{X(t) : t \geq 0\}$ with state space \mathcal{S} is described by

- its *infinitesimal generator* $\mathbf{Q} = \mathbf{R} - \text{diag}(\mathbf{R} \cdot \mathbf{1}) = \mathbf{R} - \text{diag}(\mathbf{h})^{-1} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$
- its *initial probability vector* $\boldsymbol{\pi}(0) \in \mathbb{R}^{|\mathcal{S}|}$

where

- \mathbf{R} is the *transition rate matrix*: $\mathbf{R}[\mathbf{i}, \mathbf{j}]$ is the rate of going to state \mathbf{j} when in state \mathbf{i}
- \mathbf{h} is the *expected holding time vector*: $\mathbf{h}[\mathbf{i}] = 1 / \sum_{\mathbf{j} \in \mathcal{S}} \mathbf{R}[\mathbf{i}, \mathbf{j}]$
- $\boldsymbol{\pi}(0)[\mathbf{i}] = \Pr \{\text{chain is in state } \mathbf{i} \text{ at time } 0, \text{ i.e., initially}\}$

Transient probability vector $\boldsymbol{\pi}(t) \in \mathbb{R}^{|\mathcal{S}|}$: $\boldsymbol{\pi}(t)[\mathbf{i}] = \Pr \{X(t) = \mathbf{i}\}$

- $\boldsymbol{\pi}(t)$ is the solution of $\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t) \cdot \mathbf{Q}$ with initial condition $\boldsymbol{\pi}(0)$

Steady-state probability vector $\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{S}|}$: $\boldsymbol{\pi}[\mathbf{i}] = \lim_{t \rightarrow \infty} \Pr \{X(t) = \mathbf{i}\}$

- $\boldsymbol{\pi}$ is the solution of $\boldsymbol{\pi} \cdot \mathbf{Q} = \mathbf{0}$ subject to $\sum_{\mathbf{i} \in \mathcal{S}} \boldsymbol{\pi}[\mathbf{i}] = 1$ (\mathbf{Q} must be *ergodic*)

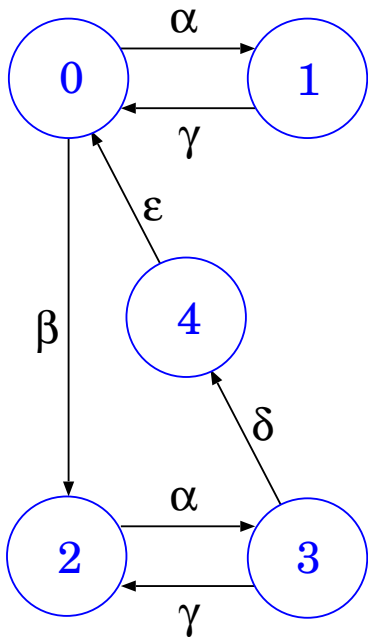
Storing a CTMC explicitly

We need to store:

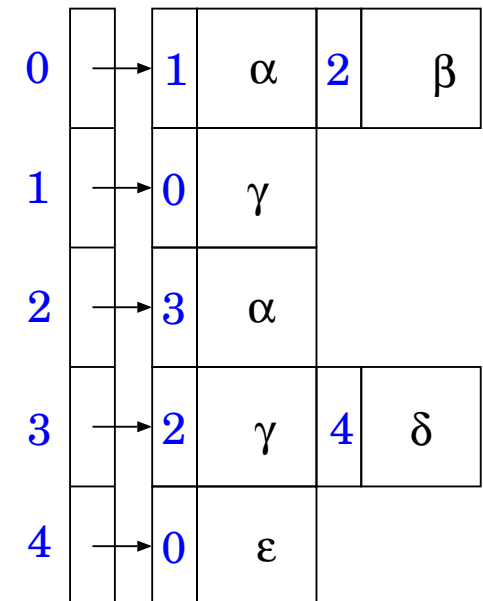
- \mathbf{R} a real matrix of size $|\mathcal{S}| \times |\mathcal{S}|$
- \mathbf{h} a real vector of size $|\mathcal{S}|$

We focus on \mathbf{R} , for which we can employ *sparse storage*:

- Requires memory proportional to $\eta(\mathbf{R})$, the number of nonzeros in \mathbf{R} , instead of $|\mathcal{S}|^2$
- Allows iterative numerical methods to run in time proportional to $\eta(\mathbf{R})$, instead of $|\mathcal{S}|^2$



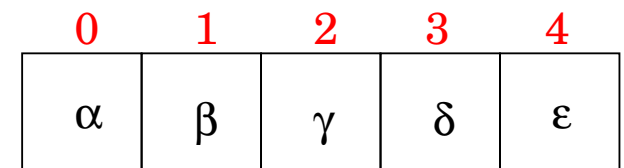
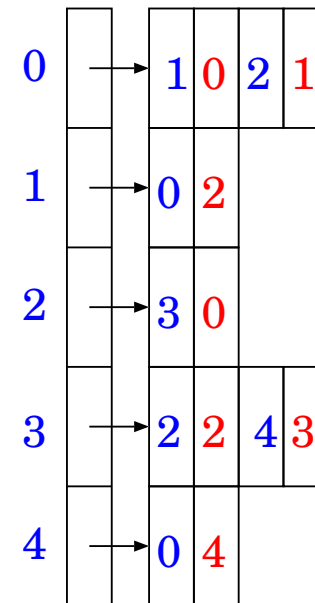
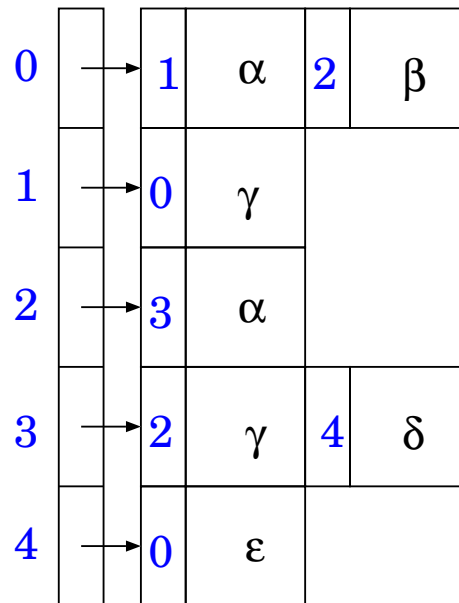
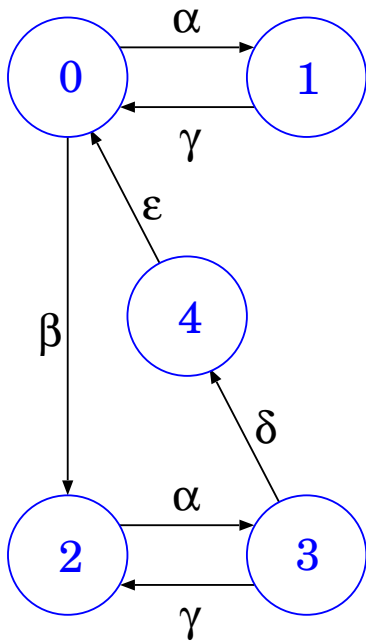
	0	1	2	3	4
0		α	β		
1	γ				
2				α	
3			γ		δ
4	ε				



Exploiting common entries with explicit storage

We can map the real values in \mathbf{R} to (small) integer indices:

- Causes a small runtime overhead
- Reduces memory by a constant factor if \mathbf{R} contains several entries with the same value
- If all nonzero entries have the same value (e.g., 1), this is essentially the *reachability graph*



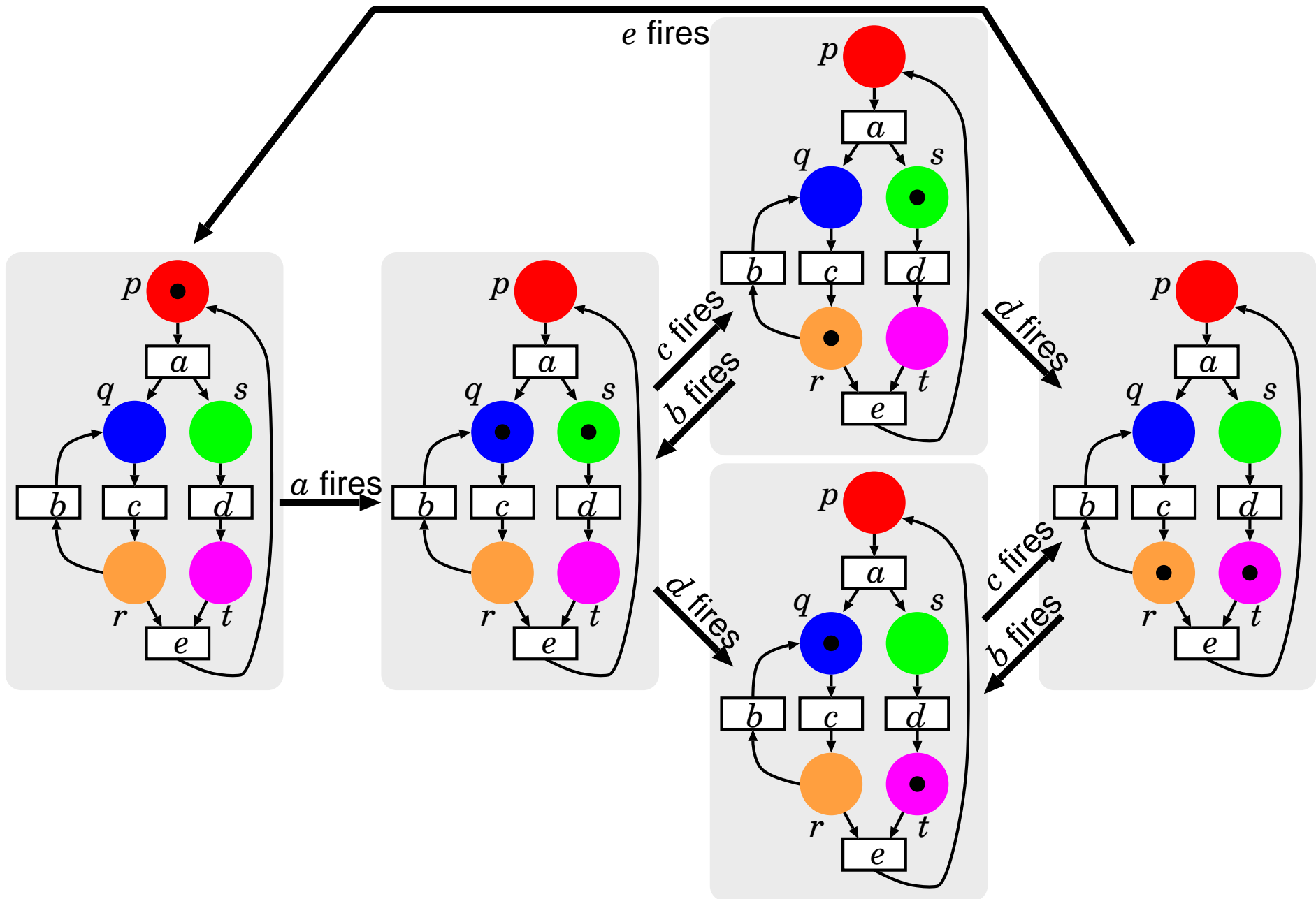
Structured models

Structured discrete-state models

A *structured discrete state model* is specified by

- a *potential state space* $\widehat{\mathcal{S}} = \mathcal{S}_K \times \cdots \times \mathcal{S}_1 = \prod_{K \geq k \geq 1} \mathcal{S}_k$
 - the “type” of the (global) state
 - \mathcal{S}_k is the (discrete) *local state space* for submodel k
- a set of *initial states* $\mathcal{S}^{init} \subseteq \widehat{\mathcal{S}}$
 - often there is a single initial state \mathbf{s}^{init}
- a set of *events* \mathcal{E} defining a *disjunctively-partitioned next-state function*
 - $\mathcal{N}_\alpha : \widehat{\mathcal{S}} \rightarrow 2^{\widehat{\mathcal{S}}}$ $\mathbf{j} \in \mathcal{N}_\alpha(\mathbf{i})$ iff state \mathbf{j} can be reached by *firing* event α in state \mathbf{i}
 - $\mathcal{N} : \widehat{\mathcal{S}} \rightarrow 2^{\widehat{\mathcal{S}}}$ is defined by $\mathcal{N}(\mathbf{i}) = \bigcup_{\alpha \in \mathcal{E}} \mathcal{N}_\alpha(\mathbf{i})$
 - we can extend \mathcal{N} to take sets of states as argument $\mathcal{N}(\mathcal{X}) = \bigcup_{\mathbf{i} \in \mathcal{X}} \mathcal{N}(\mathbf{i})$
 - α is *enabled* in \mathbf{i} iff $\mathcal{N}_\alpha(\mathbf{i}) \neq \emptyset$, otherwise it is *disabled*
 - \mathbf{i} is *absorbing*, or a *trap*, or *dead* iff $\mathcal{N}(\mathbf{i}) = \emptyset$

Petri nets and their state space \mathcal{S} : finite case



If the initial state is $s^{init} = (N, 0, 0, 0, 0, 0)$, \mathcal{S} contains $\frac{(N+1)(N+2)(2N+3)}{6}$ states

State indexing

Let the size of the k^{th} *local state space* be $n_k = |\mathcal{S}_k|$ and map \mathcal{S}_k to $\{0, 1, \dots, n_k - 1\}$

$\widehat{\mathcal{S}}$ contains $|\widehat{\mathcal{S}}| = n_K \cdot n_{K-1} \cdots n_1$ states, but not all of them are actually reachable

A *potential indexing* of a (global) state $\mathbf{i} = (\mathbf{i}_K, \dots, \mathbf{i}_1)$ is simply the *mixed-base value* of \mathbf{i} :

$$\widehat{\psi}(\mathbf{i}) = \sum_{K \geq k \geq 1} \left(\mathbf{i}_k \prod_{k > l \geq 1} n_l \right)$$

An *actual indexing* for the reachable states is instead harder to define, store, and compute

$$\psi : \widehat{\mathcal{S}} \rightarrow \{0, 1, \dots, |\mathcal{S}| - 1\} \cup \{\text{null}\}$$

in reality, we often have no a priori knowledge of \mathcal{S}_k

Explicit generation of \mathcal{S} and \mathbf{R}

Explore(in: $\mathcal{S}^{init}, \mathcal{N}$; out: $\mathcal{S}, \mathbf{R}, \psi$) is

1. $n \leftarrow 0$; *state indices start at 0*
2. $\mathcal{S} \leftarrow \emptyset$; *\mathcal{S} contains the states explored so far*
3. $\mathcal{U} \leftarrow \mathcal{S}^{init}$; *\mathcal{U} contains the unexplored states known so far*
4. for each $\mathbf{i} \in \mathcal{S}^{init}$ do
5. $\psi(\mathbf{i}) \leftarrow n++$; *assign to \mathbf{i} the next available index and increment n*
6. end for
7. while $\mathcal{U} \neq \emptyset$ do
8. choose a state \mathbf{i} in \mathcal{U} and move it from \mathcal{U} to \mathcal{S} ;
9. for each event $\alpha \in \mathcal{E}$ and each state $\mathbf{j} \in \mathcal{N}_\alpha(\mathbf{i})$ do
10. if $\mathbf{j} \notin \mathcal{S} \cup \mathcal{U}$ then *search to determine whether \mathbf{j} is a new state*
11. $\psi(\mathbf{j}) \leftarrow n++$; *assign to \mathbf{j} the next available index and increment n*
12. $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathbf{j}\}$; *remember to explore \mathbf{j} later*
13. end if;
14. $\mathbf{R}[\psi(\mathbf{i}), \psi(\mathbf{j})] \leftarrow \mathbf{R}[\psi(\mathbf{i}), \psi(\mathbf{j})] + \lambda_\alpha(\mathbf{i})\Delta_\alpha(\mathbf{i}, \mathbf{j})$; *ψ is used to **index** \mathbf{R}*
15. end for;
16. end while;

$\psi : \widehat{\mathcal{S}} \rightarrow \{0, \dots, |\mathcal{S}| - 1\} \cup \{\text{null}\}$ is a state indexing function (e.g., discovery order)

$\lambda_\alpha(\mathbf{i})$ is the *rate* at which event α *fires* in state \mathbf{i}

$\Delta_\alpha(\mathbf{i}, \mathbf{j})$ is the probability that, if event α fires in state \mathbf{i} , the next state is \mathbf{j}

(Quasi-reduced ordered) multiway decision diagrams

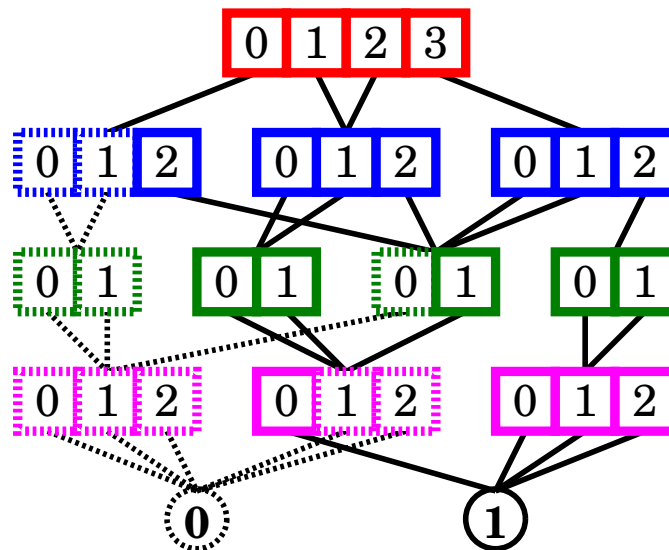
- Nodes are organized into $K + 1$ levels
 - Level K contains only one root node
 - Levels $K - 1$ through 1 contain one or more nodes, NO DUPLICATES
 - Level 0 contains only the two terminal nodes, **0** and **1** (false and true).
- For $k > 0$, a node at level k has $|\mathcal{S}_k|$ arcs pointing to nodes at level $k - 1$

$$\mathcal{S}_4 = \{0, 1, 2, 3\}$$

$$\mathcal{S}_3 = \{0, 1, 2\}$$

$$\mathcal{S}_2 = \{0, 1\}$$

$$\mathcal{S}_1 = \{0, 1, 2\}$$



$$\mathcal{S} = \left\{ \begin{array}{cccccccccccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 \end{array} \right\}$$

[Kam et al. 1998] defined fully-reduced ordered MDDs as an interface to BDDs

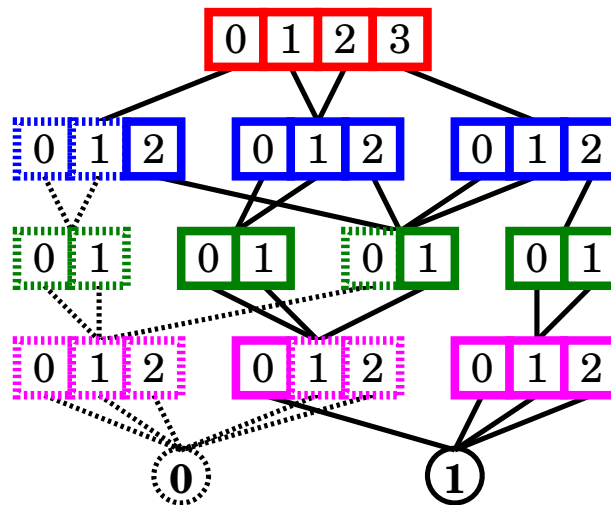
How to define and store the indexing function ψ using MDDs

$$\mathcal{S}_4 = \{0, 1, 2, 3\}$$

$$\mathcal{S}_3 = \{0, 1, 2\}$$

$$\mathcal{S}_2 = \{0, 1\}$$

$$\mathcal{S}_1 = \{0, 1, 2\}$$



$$\mathcal{S} = \left\{ \begin{array}{cccccccccccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 0 & 1 & 2 & 2 & 2 & 2 & 2 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 \end{array} \right\}$$

To compute the index of a state, use *edge-value offsets*:

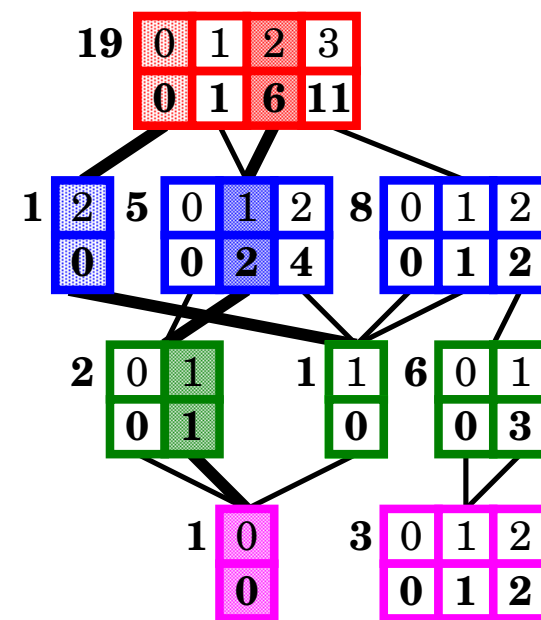
- Sum the offsets found on the corresponding path:

$$\psi(2, 1, 1, 0) = 6 + 2 + 1 + 0 = 9$$

- A state is unreachable if the path is not complete:

$$\psi(0, 2, 0, 0) = 0 + 0 + ? + ? = \text{null}$$

lexicographic, not discovery, order!!!



State indexing options: potential $\hat{\psi}$ vs. actual ψ

Once we know \mathcal{S} :

- We can store the original $\mathcal{N} : \hat{\mathcal{S}} \rightarrow 2^{\hat{\mathcal{S}}}$ or its restriction $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$
- We can store $\hat{\mathbf{R}} : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow \mathbb{R}$ or $\mathbf{R} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$
- We can choose algorithms that use $\hat{\pi} : \hat{\mathcal{S}} \rightarrow \mathbb{R}$ or $\pi : \mathcal{S} \rightarrow \mathbb{R}$

With strictly explicit methods: using **actual \mathbf{R} and π** works best

With (even just partially) implicit methods, there are tradeoffs

- Storing π instead of $\hat{\pi}$ is often unavoidable if we employ a full vector
- Symbolic storage of $\hat{\mathbf{R}}$ is much cheaper than that of \mathbf{R} in terms of memory requirements
- However, using $\hat{\mathbf{R}}$ in conjunction with π complicates indexing...
- ...at the very least, it forces us to store $\psi : \hat{\mathcal{S}} \rightarrow \{0, 1, \dots, |\mathcal{S}| - 1\} \cup \{\text{null}\}$, hence \mathcal{S}

Saturation: an iteration strategy based on the model structure

MDD node p at level k is **saturated** if the set of states it encodes is a fixed point w.r.t. any α s.t. $Top(\alpha) \leq k$ (thus, all nodes below p are also saturated)

- build the K -level MDD encoding of \mathcal{S}^{init} (if $|\mathcal{S}^{init}| = 1$, there is one node per level)
- saturate each node at level 1: fire in them all events α s.t. $Top(\alpha) = 1$
- saturate each node at level 2: fire in them all events α s.t. $Top(\alpha) = 2$
(if this creates nodes at level 1, saturate them immediately upon creation)
- saturate each node at level 3: fire in them all events α s.t. $Top(\alpha) = 3$
(if this creates nodes at levels 2 or 1, saturate them immediately upon creation)
- ...
- saturate the root node at level K : fire in it all events α s.t. $Top(\alpha) = K$
(if this creates nodes at levels $K-1, K-2, \dots, 1$, saturate them immediately upon creation)

states are **not** discovered in breadth-first order

Solution requirements: SMART vs. NuSMV (800MHz P-II)

Time and memory to generate \mathcal{S} using saturation in SMART vs. breadth-first iterations in NuSMV

N	$ \mathcal{S} $	Final memory (kB)		Peak memory (kB)		Time (sec)	
		SMART	NuSMV	SMART	NuSMV	SMART	NuSMV

Dining Philosophers: $K = N$

50	2.23×10^{31}	18	10,800	22	10,819	0.15	5.9
200	2.47×10^{125}	74	27,155	93	72,199	0.68	12,905.7
10,000	4.26×10^{6269}	3,749	—	4,686	—	877.82	—

Slotted Ring Network: $K = N$

10	8.29×10^9	4	5,287	28	10,819	0.13	5.5
15	1.46×10^{15}	10	9,386	80	13,573	0.39	2,039.5
200	8.38×10^{211}	1,729	—	120,316	—	902.11	—

Round Robin Mutual Exclusion: $K = N + 1$

20	4.72×10^7	18	7,300	20	7,306	0.07	0.8
100	2.85×10^{32}	356	16,228	372	26,628	3.81	2,475.3
300	1.37×10^{93}	3,063	—	3,109	—	140.98	—

Flexible Manufacturing System: $K = 19$

10	4.28×10^6	16	1,707	26	11,238	0.05	9.4
20	3.84×10^9	55	14,077	101	31,718	0.20	1,747.8
250	3.47×10^{26}	25,507	—	69,087	—	231.17	—

Decision–diagram–based CTMC encodings

Multiterminal multiway decision diagrams

From BDDs to MDDs: allow multiway choices at each nonterminal node, not just binary choices

From BDDs to MTBDDs: allow multiple terminal nodes, not just 0 and 1

From BDDs to MTMDDs combine both generalizations

We can use a *quasi-reduced* MTMDD to encode a real matrix $\mathbf{A} : \widehat{\mathcal{S}} \times \widehat{\mathcal{S}} \rightarrow \mathbb{R}$

- Nodes are organized into $2K + 1$ levels
 - Variables $\{\mathbf{i}_K, \mathbf{i}_{K-1}, \dots, \mathbf{i}_1, \mathbf{j}_K, \mathbf{j}_{K-1}, \dots, \mathbf{j}_1\}$ are mapped onto $\{2K, 2K-1, \dots, 1\}$
 - Let $v(l)$ be the variable corresponding to level $l \in \{2K, 2K-1, \dots, 1\}$
 - Level $2K$ contains only one root node
 - Levels $2K-1$ through 1 contain one or more nodes, *no duplicate nodes allowed*
 - Level 0 contains as many nodes as the different entries in \mathbf{A}
- A node at level $l > 0$, with $v(l) = \mathbf{i}_k$ or \mathbf{j}_k , has $|\mathcal{S}_k|$ arcs pointing to nodes at level $l-1$

$\mathbf{A}[\mathbf{i}, \mathbf{j}] = x \Leftrightarrow$ path labelled $\{\mathbf{i}_K, \mathbf{i}_{K-1}, \dots, \mathbf{i}_1, \mathbf{j}_K, \mathbf{j}_{K-1}, \dots, \mathbf{j}_1\}$ leads to node x at level 0

MTMDDs encoding of the transition rate matrix

In principle, the mapping v can be any permutation of the $2K$ variables

In practice, the order $(\mathbf{i}_K, \mathbf{j}_K, \mathbf{i}_{K-1}, \mathbf{j}_{K-1}, \dots, \mathbf{i}_1, \mathbf{j}_1)$ is usually the best choice
(we still need to decide a good order for the K “from-to” pairs)

When using MTMDDs to store the transition rate matrix, we have a choice:

- Store $\hat{\mathbf{R}} : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow \mathbb{R}$ (note that $\hat{\mathbf{R}}[\mathbf{i}, \mathbf{j}] = 0$ if $\mathbf{i} \in \mathcal{S}$ and $\mathbf{j} \notin \mathcal{S}$)
 - a natural choice if we use a compositional approach
- Store $\mathbf{R} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ (actually, $\hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow \mathbb{R}$, but $\mathbf{R}[\mathbf{i}, \mathbf{j}] = 0$ if $\mathbf{i} \notin \mathcal{S}$ or $\mathbf{j} \notin \mathcal{S}$)
 - usually requires more MTMDD nodes
 - can be built by enumerating the entries explicitly and storing them implicitly in an MTMDD
 - or by zeroing the rows corresponding to $\hat{\mathcal{S}} \setminus \mathcal{S}$ in the MTMDD encoding of $\hat{\mathbf{R}}$
i.e., premultiplying $\hat{\mathbf{R}}$ by a filtering diagonal matrix $\hat{\mathbf{F}}[\mathbf{i}, \mathbf{i}] = 1$ if $\mathbf{i} \in \mathcal{S}$, 0 if $\mathbf{i} \in \hat{\mathcal{S}} \setminus \mathcal{S}$

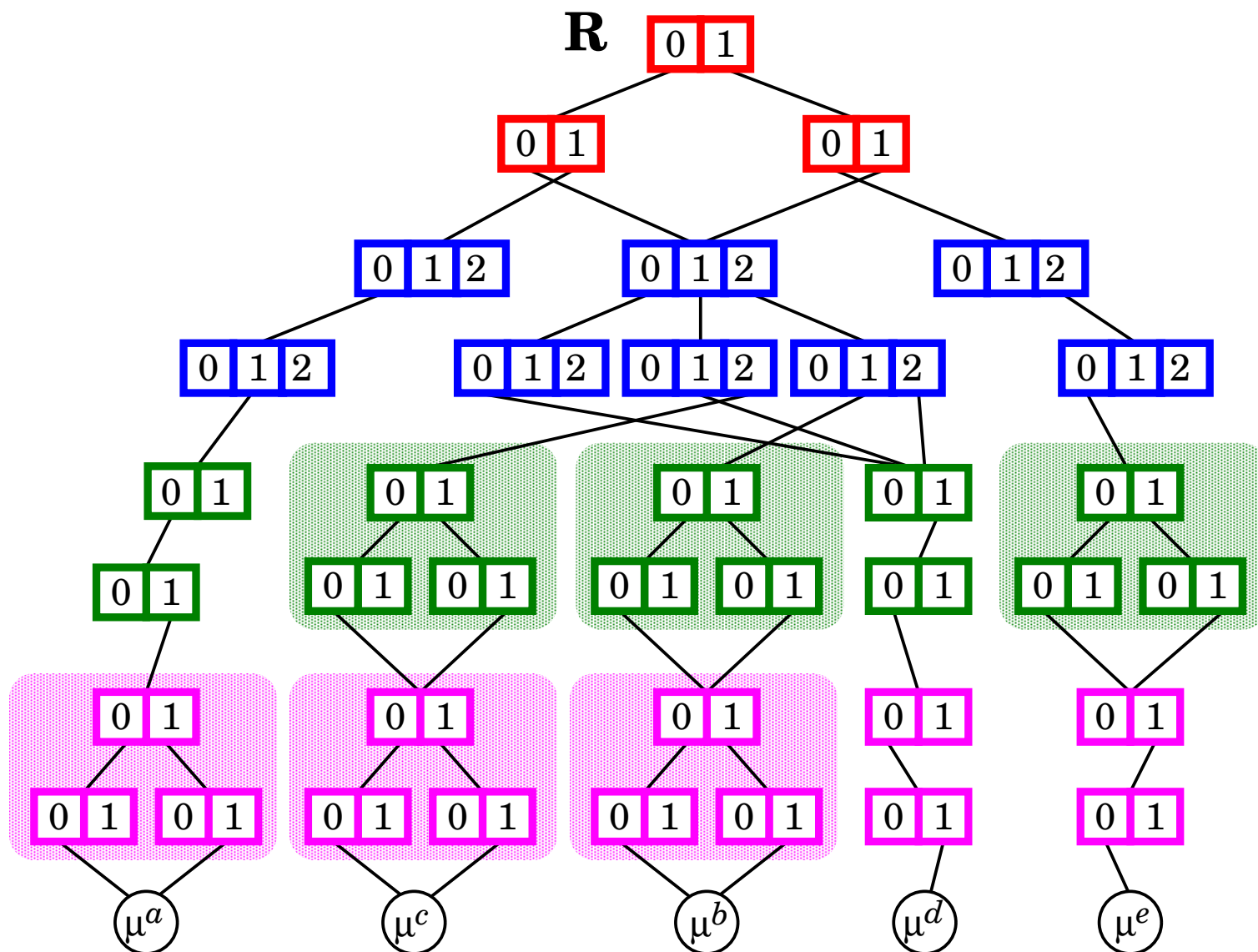
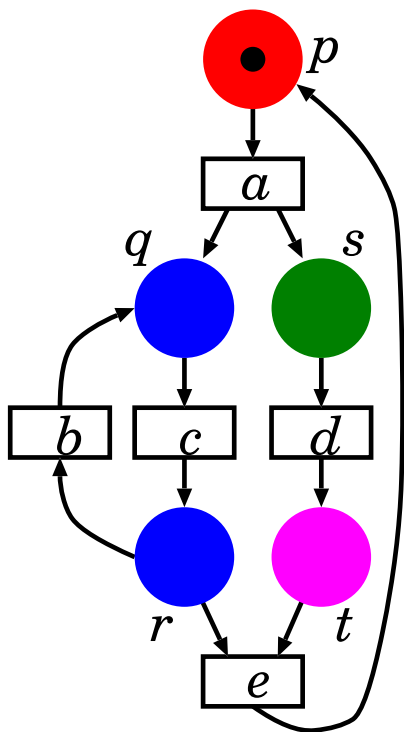
An example of MTMDD

$$\mathcal{S}_4: \{p^1, p^0\} \equiv \{0, 1\}$$

$$\mathcal{S}_3: \{q^0 r^0, q^1 r^0, q^0 r^1\} \equiv \{0, 1, 2\}$$

$$\mathcal{S}_2: \{s^0, s^1\} \equiv \{0, 1\}$$

$$\mathcal{S}_1: \{t^0, t^1\} \equiv \{0, 1\}$$



Definition of Kronecker product

Given K matrices $\mathbf{A}_k \in \mathbb{R}^{n_k \times n_k}$, their *Kronecker product* is

$$\mathbf{A} = \bigotimes_{K \geq k \geq 1} \mathbf{A}_k \in \mathbb{R}^{n_K \cdots n_1 \times n_K \cdots n_1}$$

where

- $\mathbf{A}[\mathbf{i}, \mathbf{j}] = \mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K] \cdot \mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}] \cdots \mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1]$
- using the mixed-base numbering scheme (indices start at 0)

$$\mathbf{i} = (\dots((\mathbf{i}_K) \cdot n_{K-1} + \mathbf{i}_{K-1}) \cdot n_{K-2} \cdots) \cdot n_1 + \mathbf{i}_1 = \sum_{K \geq k \geq 1} \mathbf{i}_k \cdot \prod_{k > l \geq 1} n_l$$

nonzeros: $\eta \left(\bigotimes_{K \geq k \geq 1} \mathbf{A}_k \right) = \prod_{K \geq k \geq 1} \eta(\mathbf{A}_k)$

Kronecker product by example

Given the real matrices $\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$

$$\mathbf{A} \otimes \mathbf{B} = \left[\begin{array}{c|c} a_{00}\mathbf{B} & a_{01}\mathbf{B} \\ \hline a_{10}\mathbf{B} & a_{11}\mathbf{B} \end{array} \right] = \begin{bmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{00}b_{02} & a_{01}b_{00} & a_{01}b_{01} & a_{01}b_{02} \\ a_{00}b_{10} & a_{00}b_{11} & a_{00}b_{12} & a_{01}b_{10} & a_{01}b_{11} & a_{01}b_{12} \\ a_{00}b_{20} & a_{00}b_{21} & a_{00}b_{22} & a_{01}b_{20} & a_{01}b_{21} & a_{01}b_{22} \\ \hline a_{10}b_{00} & a_{10}b_{01} & a_{10}b_{02} & a_{11}b_{00} & a_{11}b_{01} & a_{11}b_{02} \\ a_{10}b_{10} & a_{10}b_{11} & a_{10}b_{12} & a_{11}b_{10} & a_{11}b_{11} & a_{11}b_{12} \\ a_{10}b_{20} & a_{10}b_{21} & a_{10}b_{22} & a_{11}b_{20} & a_{11}b_{21} & a_{11}b_{22} \end{bmatrix}$$

Kronecker product expresses *contemporaneity* or *synchronization*

If \mathbf{A} and \mathbf{B} are the transition probability matrices of independent discrete-time Markov chains

$\Rightarrow \mathbf{A} \otimes \mathbf{B}$ is the transition probability matrix of their composition

Kronecker-consistent decomposition of a CTMC model

A decomposition of a discrete-state model describing a CTMC is *Kronecker-consistent* if:

- the potential transition rate matrix $\hat{\mathbf{R}}$ is *additively partitioned*

$$\hat{\mathbf{R}} = \sum_{\alpha \in \mathcal{E}} \hat{\mathbf{R}}_{\alpha}$$

- $\hat{\mathcal{S}} = \mathcal{S}_K \times \cdots \times \mathcal{S}_1$, a *global* state \mathbf{i} consists of K *local* states

$$\mathbf{i} = (\mathbf{i}_K, \dots, \mathbf{i}_1)$$

- and, most importantly, we can *multiplicatively partition* each $\hat{\mathbf{R}}_{\alpha}$, that is, we can write

$$\lambda_{\alpha}(\mathbf{i}) = \lambda_{K,\alpha}(\mathbf{i}_K) \cdots \lambda_{1,\alpha}(\mathbf{i}_1)$$

and

$$\Delta_{\alpha}(\mathbf{i}, \mathbf{j}) = \Delta_{K,\alpha}(\mathbf{i}_K, \mathbf{j}_K) \cdots \Delta_{1,\alpha}(\mathbf{i}_1, \mathbf{j}_1)$$

$$\hat{\mathbf{R}}_{\alpha} = \mathbf{R}_{K,\alpha} \otimes \cdots \otimes \mathbf{R}_{1,\alpha}$$

We encode the potential transition rate matrix $\hat{\mathbf{R}}$ with $|\mathcal{E}| \times K$ small matrices

$$\mathbf{R}_{k,\alpha} \in \mathbb{R}^{n_k \times n_k}$$

for stochastic Petri nets with transition rates depending on at most one place, any partition of the places into K subsets is consistent (even with inhibitor, reset, or probabilistic arcs)

Kronecker description of the transition rate matrix of a CTMC

- Parallel composition of K submodels with overall *event* set \mathcal{E} (synchronizing vs. local)
- *Global* state \mathbf{i} is a K -tuple $(\mathbf{i}_K, \dots, \mathbf{i}_1)$ of *local* states $\mathcal{S} \subseteq \widehat{\mathcal{S}} = \mathcal{S}_K \times \dots \times \mathcal{S}_1$
- *Transition rate matrix* $\mathbf{R} = \widehat{\mathbf{R}}[\mathcal{S}, \mathcal{S}]$ where $\widehat{\mathbf{R}} = \sum_{\alpha \in \mathcal{E}} \bigotimes_{K \geq k \geq 1} \mathbf{R}_{k,\alpha}$
- $\mathbf{R}_{k,\alpha}[\mathbf{i}_k, \mathbf{j}_k] = \begin{cases} \lambda_{k,\alpha}(\mathbf{i}_k) \cdot \Delta_{k,\alpha}(\mathbf{i}_k, \mathbf{j}_k) & \text{if } \alpha \text{ is in submodel } k \\ 1 & \text{if } \alpha \text{ is not in submodel } k \text{ and } \mathbf{i}_k = \mathbf{j}_k \\ 0 & \text{if } \alpha \text{ is not in submodel } k \text{ and } \mathbf{i}_k \neq \mathbf{j}_k \end{cases}$

encode a huge \mathbf{R} with $K \cdot |\mathcal{E}|$ “small” matrices

“On the stochastic structure of parallelism and synchronisation models for distributed algorithms”
Plateau (SIGMETRICS 1985)

factor K slowdown, still needs a probability vector of size $|\mathcal{S}|$

“Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models” Buchholz, Ciardo, Donatelli, Kemper (INFORMS J. Comp., 2000)

Kronecker encoding of \mathcal{R} ($K = 5$)

$\mathcal{S}_5 = ?$

$\mathcal{S}_4 = ?$

$\mathcal{S}_3 = ?$

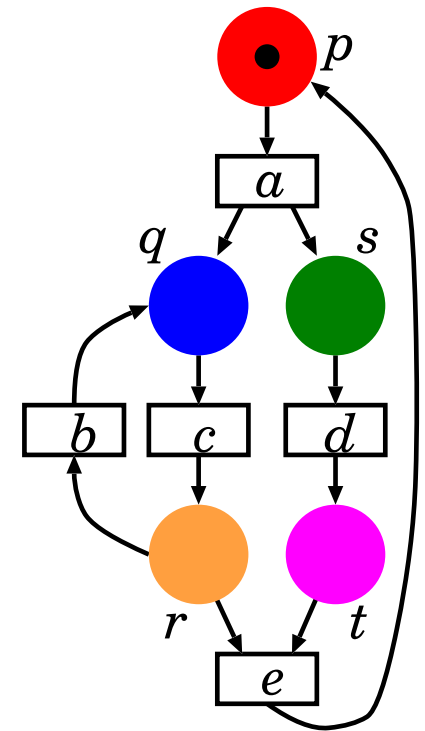
$\mathcal{S}_2 = ?$

$\mathcal{S}_1 = ?$

EVENTS \rightarrow

LEVELS \downarrow

$\mathbf{R}_{5,a}:?$	I	I	I	$\mathbf{R}_{5,e}:?$
$\mathbf{R}_{4,a}:?$	$\mathbf{R}_{4,b}:?$	$\mathbf{R}_{4,c}:?$	I	I
I	$\mathbf{R}_{3,b}:?$	$\mathbf{R}_{3,c}:?$	I	$\mathbf{R}_{3,e}:?$
$\mathbf{R}_{2,a}:?$	I	I	$\mathbf{R}_{2,d}:?$	I
I	I	I	$\mathbf{R}_{1,d}:?$	$\mathbf{R}_{1,e}:?$

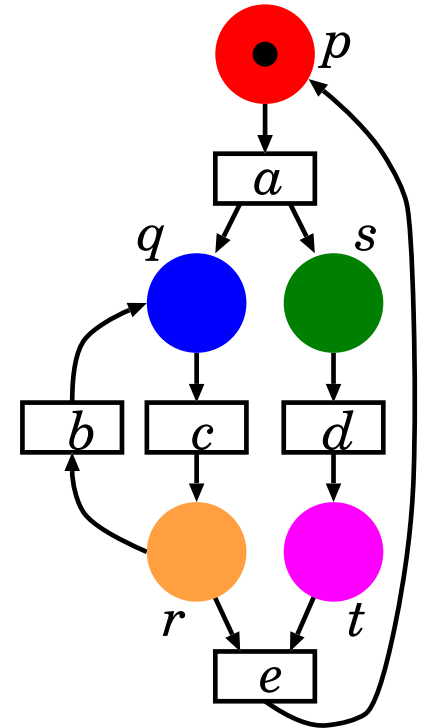


we determine a priori from the model whether $\mathbf{R}_{k,\alpha} = \mathbf{I}$

Kronecker encoding of $\mathbf{R} = \sum_{\alpha \in \{a,b,c,d,e\}} \bigotimes_{5 \geq k \geq 1} \mathbf{R}_{k,\alpha}$

$$\mathcal{S}_5 : \{p^1, p^0\} \equiv \{0, 1\} \quad \mathcal{S}_4 : \{q^0, q^1\} \equiv \{0, 1\} \quad \mathcal{S}_3 : \{r^0, r^1\} \equiv \{0, 1\} \quad \mathcal{S}_2 : \{s^0, s^1\} \equiv \{0, 1\} \quad \mathcal{S}_1 : \{t^0, t^1\} \equiv \{0, 1\}$$

		EVENTS \rightarrow				
LEVELS \downarrow	$\mathbf{R}_{5,a} : \begin{bmatrix} 0 & \gamma_5^a \\ 0 & 0 \end{bmatrix}$	I	I	I	$\mathbf{R}_{5,e} : \begin{bmatrix} 0 & 0 \\ \gamma_5^e & 0 \end{bmatrix}$	
	$\mathbf{R}_{4,a} : \begin{bmatrix} 0 & \gamma_4^a \\ 0 & 0 \end{bmatrix}$	$\mathbf{R}_{4,b} : \begin{bmatrix} 0 & \gamma_4^b \\ 0 & 0 \end{bmatrix}$	$\mathbf{R}_{4,c} : \begin{bmatrix} 0 & 0 \\ \gamma_4^c & 0 \end{bmatrix}$	I	I	
	I	$\mathbf{R}_{3,b} : \begin{bmatrix} 0 & 0 \\ \gamma_3^b & 0 \end{bmatrix}$	$\mathbf{R}_{3,c} : \begin{bmatrix} 0 & \gamma_3^c \\ 0 & 0 \end{bmatrix}$	I	$\mathbf{R}_{3,e} : \begin{bmatrix} 0 & 0 \\ \gamma_3^e & 0 \end{bmatrix}$	
	$\mathbf{R}_{2,a} : \begin{bmatrix} 0 & \gamma_2^a \\ 0 & 0 \end{bmatrix}$	I	I	$\mathbf{R}_{2,d} : \begin{bmatrix} 0 & 0 \\ \gamma_2^d & 0 \end{bmatrix}$	I	
	I	I	I	$\mathbf{R}_{1,d} : \begin{bmatrix} 0 & \gamma_1^d \\ 0 & 0 \end{bmatrix}$	$\mathbf{R}_{1,e} : \begin{bmatrix} 0 & 0 \\ \gamma_1^e & 0 \end{bmatrix}$	



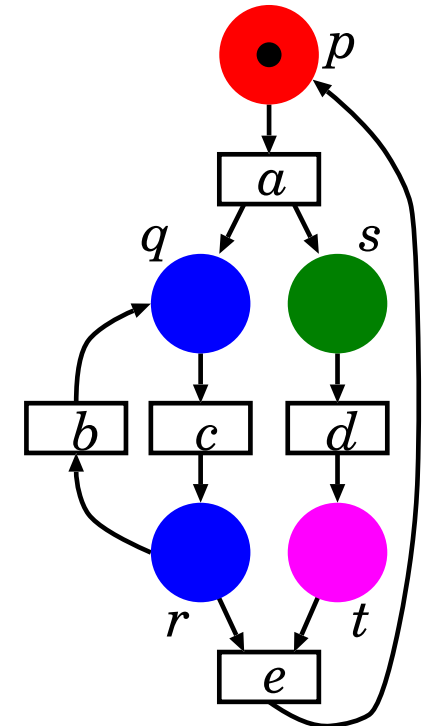
$$\mathbf{R}[00i_30i_1, 11i_31i_1] = \gamma_5^a \cdot \gamma_4^a \cdot \gamma_2^a$$

Not a canonical representation: changing to $\gamma_5^a \cdot 2$ and $\gamma_4^a / 2$ would describe the same \mathbf{R}

Kronecker encoding of \mathbb{R} ($K = 4$)

 $\mathcal{S}_4 = ?$ $\mathcal{S}_3 = ?$ $\mathcal{S}_2 = ?$ $\mathcal{S}_1 = ?$

		EVENTS \rightarrow				
LEVELS \downarrow	$\mathbf{R}_{4,a} : ?$	I	I	I	$\mathbf{R}_{4,e} : ?$	
	$\mathbf{R}_{3,a} : ?$	$\mathbf{R}_{3,b} : ?$	$\mathbf{R}_{3,c} : ?$	I	$\mathbf{R}_{3,e} : ?$	
	$\mathbf{R}_{2,a} : ?$	I	I	$\mathbf{R}_{2,d} : ?$	I	
	I	I	I	$\mathbf{R}_{1,d} : ?$	$\mathbf{R}_{1,e} : ?$	



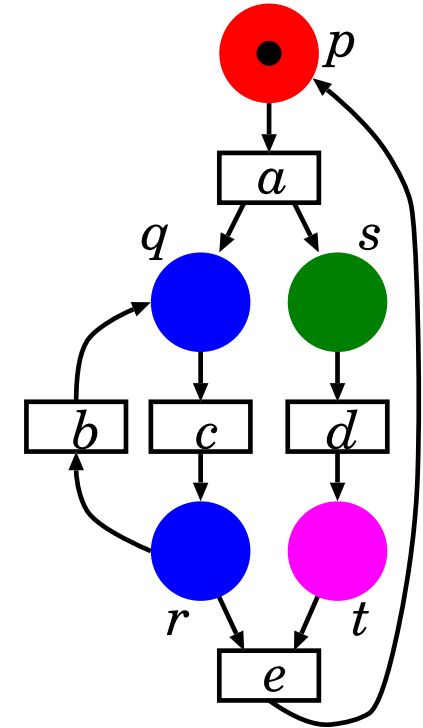
The matrices for b and c differ only at level **3**: we can merge them into a single (local) event l

we determine automatically from the model whether $\mathbf{R}_{k,\alpha} = \mathbf{I}$

Kronecker encoding of $\mathbf{R} = \sum_{\alpha \in \{a,l,d,e\}} \bigotimes_{4 \geq k \geq 1} \mathbf{R}_{k,\alpha}$

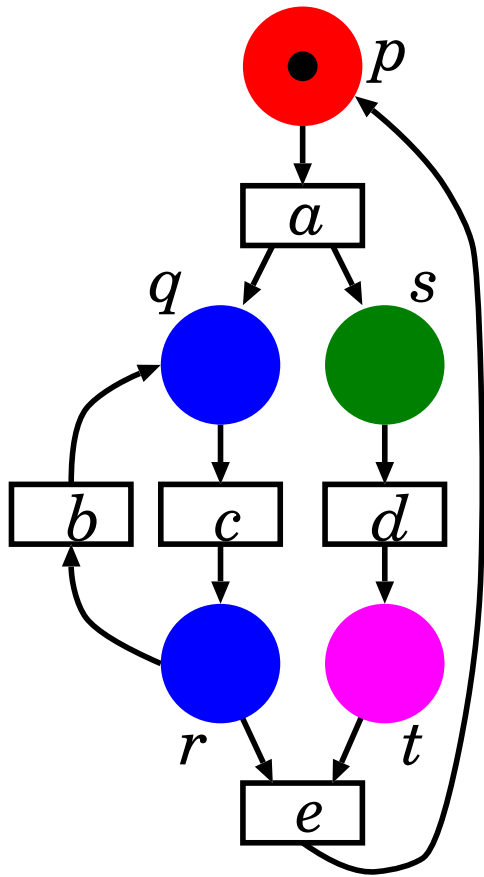
$$\mathcal{S}_4: \{p^1, p^0\} \equiv \{0,1\} \quad \mathcal{S}_3: \{q^0 r^0, q^1 r^0, q^0 r^1\} \equiv \{0,1,2\} \quad \mathcal{S}_2: \{s^0, s^1\} \equiv \{0,1\} \quad \mathcal{S}_1: \{t^0, t^1\} \equiv \{0,1\}$$

		EVENTS \rightarrow			
LEVELS \downarrow	$\mathbf{R}_{4,a}: \begin{bmatrix} 0 & \mu_4^a \\ 0 & 0 \end{bmatrix}$	\mathbf{I}	\mathbf{I}	$\mathbf{R}_{4,e}: \begin{bmatrix} 0 & 0 \\ \mu_4^e & 0 \end{bmatrix}$	
	$\mathbf{R}_{3,a}: \begin{bmatrix} 0 & \mu_3^a & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\mathbf{R}_{3,l}: \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \mu_3^c \\ 0 & \mu_3^b & 0 \end{bmatrix}$	\mathbf{I}	$\mathbf{R}_{3,e}: \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu_3^e & 0 & 0 \end{bmatrix}$	
	$\mathbf{R}_{2,a}: \begin{bmatrix} 0 & \mu_2^a \\ 0 & 0 \end{bmatrix}$	\mathbf{I}	$\mathbf{R}_{2,d}: \begin{bmatrix} 0 & 0 \\ \mu_2^d & 0 \end{bmatrix}$	\mathbf{I}	
	\mathbf{I}	\mathbf{I}	$\mathbf{R}_{1,d}: \begin{bmatrix} 0 & \mu_1^d \\ 0 & 0 \end{bmatrix}$	$\mathbf{R}_{1,e}: \begin{bmatrix} 0 & 0 \\ \mu_1^e & 0 \end{bmatrix}$	



We have merged b and c into a single (local) event l

The matrix \mathbb{R} encoded by the Kronecker descriptor ($K = 4$)



$$\{p^1, p^0\} \equiv \{0, 1\}$$

$$\{q^0 r^0, q^1 r^0, q^0 r^1\} \equiv \{0, 1, 2\}$$

$$\{s^0, s^1\} \equiv \{0, 1\}$$

$$\{t^0, t^1\} \equiv \{0, 1\}$$

	00	00	00	00	00	00	11	11	11	11	11	11
	00	00	11	11	22	22	00	00	11	11	22	22
	00	11	00	11	00	11	00	11	00	11	00	11
	01	01	01	01	01	01	01	01	01	01	01	01
0000	·	·	·	·	·	·	·	·	·	·	a	·
0001	·	·	·	·	·	·	·	·	·	·	a	·
0010	d	·	·	·	·	·	·	·	·	·	·	·
0011	·	·	·	·	·	·	·	·	·	·	·	·
0100	·	·	·	·	c	·	·	·	·	·	·	·
0101	·	·	·	·	c	·	·	·	·	·	·	·
0110	·	·	d	·	·	c	·	·	·	·	·	·
0111	·	·	·	·	·	c	·	·	·	·	·	·
0200	·	·	b	·	·	·	·	·	·	·	·	·
0201	·	·	b	·	·	·	·	·	·	·	·	·
0210	·	·	·	b	·	d	·	·	·	·	·	·
0211	·	·	·	b	·	·	·	·	·	·	·	·
1000	·	·	·	·	·	·	·	·	·	·	·	·
1001	·	·	·	·	·	·	·	·	·	·	·	·
1010	·	·	·	·	·	·	d	·	·	·	·	·
1011	·	·	·	·	·	·	·	·	·	·	·	·
1100	·	·	·	·	·	·	·	·	·	·	c	·
1101	·	·	·	·	·	·	·	·	·	·	c	·
1110	·	·	·	·	·	·	·	d	·	·	·	c
1111	·	·	·	·	·	·	·	·	·	·	·	c
1200	·	·	·	·	·	·	·	·	b	·	·	·
1201	e	·	·	·	·	·	·	·	b	·	·	·
1210	·	·	·	·	·	·	·	·	·	b	·	·
1211	·	e	·	·	·	·	·	·	·	b	d	·

Matrix diagrams (MXDs)

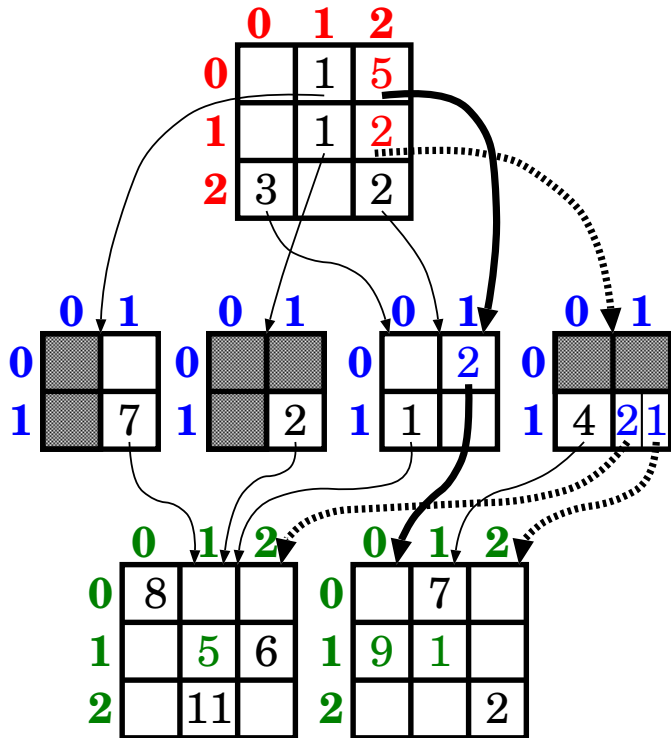
A generalization of the idea of Kronecker encoding of a matrix

Allows us to enforce knowledge of the reachable states \mathcal{S} to the potential transition rate matrix $\hat{\mathbf{R}}$

An example of (non-canonical) MXD:

$$\mathbf{R}[\mathbf{001}, \mathbf{210}] = 5 * 2 * 9 = 90$$

$$\mathbf{R}[\mathbf{111}, \mathbf{211}] = 2 * (2 * 5 + 1 * 1) = 22$$



	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	
	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1
	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
000																		70
001																	(90)	10
002																		20
010									56			40						
011										35	42		25	30				
012										77			55					
100																		
101																		
102																		
110									16				56		32	14		
111										10	12	72	8		18	22	24	
112										22				16		44	4	
200											42							28
201											54	6					36	4
202												12						8
210	24													16				
211		15	18												10	12		
212		33													22			

How to build an MXD

Two methods for building MXDs in our tool SMART:

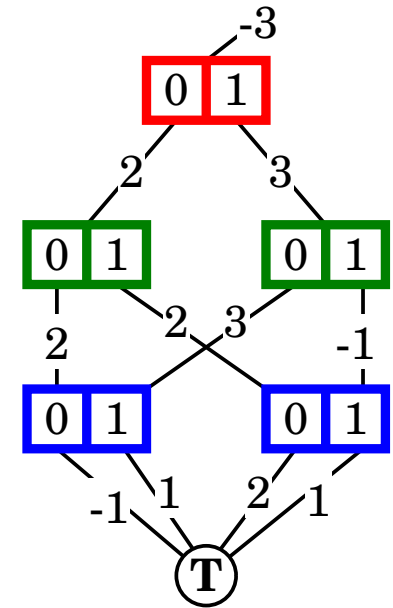
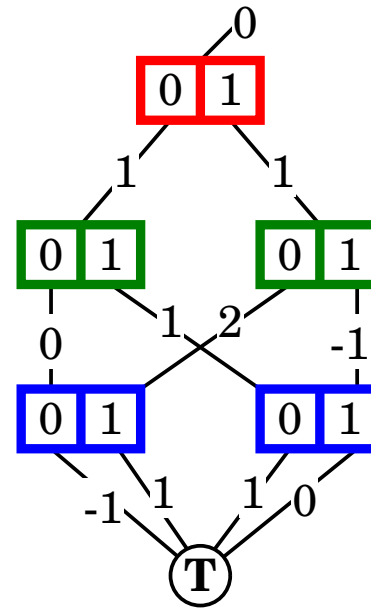
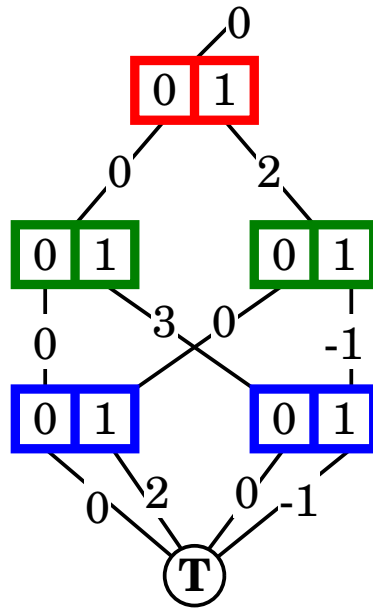
- **From the Kronecker encoding**
 - Build the matrices for the Kronecker encoding
 - Insert the matrices in an MXD, removing duplicate (and redundant?) ones
 - Since we start from a Kronecker encoding, we have the same limitations
 - Similar memory requirements to a Kronecker encoding in practice
 - We can use them to store \mathbf{R} instead of $\widehat{\mathbf{R}}$
- **From an explicit enumeration of the entries**
 - Requires the use of *canonical MXDs* [Miner PNPM'01]
 - Entries are added individually (or in *batches* for greater efficiency)
 - Requires more time (additional overhead prior to numerical solution) and memory
 - **They are fully general, can encode *any* matrix, do not need Kronecker consistency**

MXDs exploit the presence of identity matrices $\mathbf{R}_{k,\alpha}$
(skipped levels)

EVBDDs by example

[Lai et al. 1992] defined *edge-valued binary decision diagrams*

i_3	0 0 0 0 1 1 1 1
i_2	0 0 1 1 0 0 1 1
i_1	0 1 0 1 0 1 0 1
f	0 2 3 2 2 4 1 0



Canonicity: *all nodes have 0-value on the 0-arc*

(only the first EVBDD is canonical)

In canonical form, the root incoming edge has value $f(0 \dots 0)$

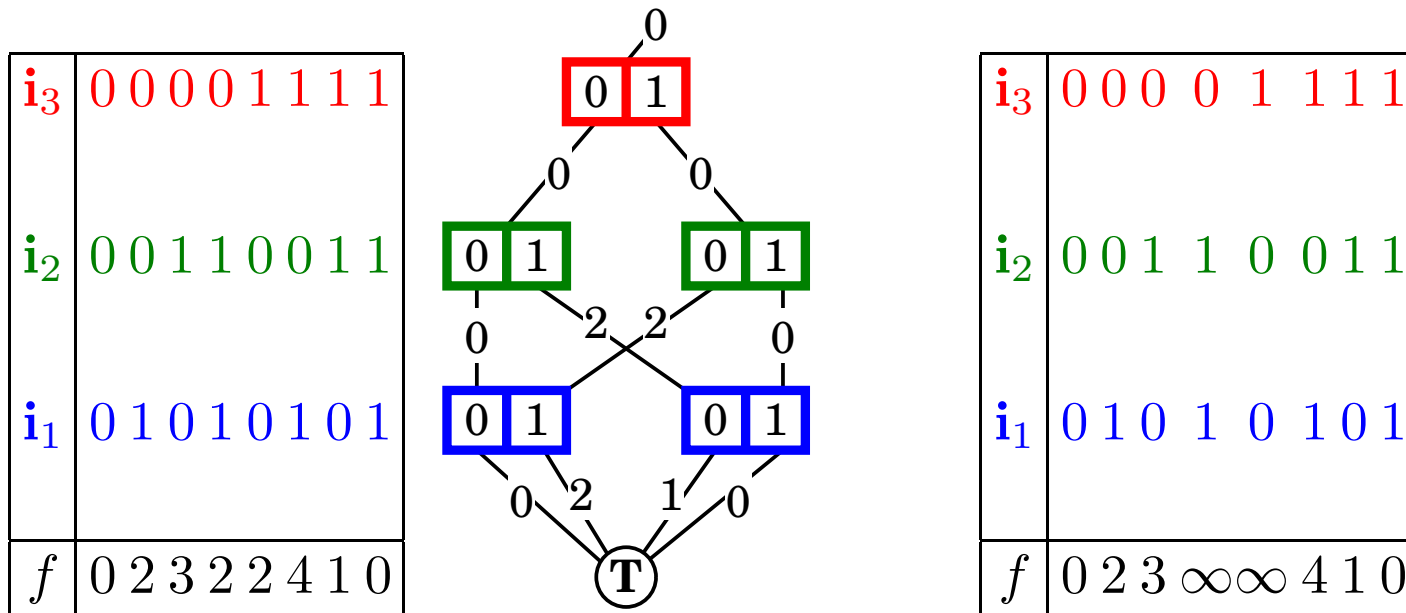
EV⁺MDDs by example

[Ciardo and Siminiceanu 2002] defined *edge-valued positive multiway decision diagrams*

From BDD to MDD: the usual extension

We allow ∞ -edge values: can store partial arithmetic functions

New canonization rule: essential to encode partial arithmetic functions



Canonicity: *all edge values are non-negative and at least one is zero*

In canonical form, the root incoming edge has value $\min_{\mathbf{i} \in \hat{S}} f(\mathbf{i})$

$$f(1, 0, 0) = \infty \quad \text{but} \quad f(1, 0, 1) = 4$$

the traditional EVMDD normalization cannot represent this function

Definition of EV^+ MDDs

(differences from EVBDDs of [Lai et al. 1992] are in blue)

Given $f: \widehat{\mathcal{S}} \rightarrow \mathbb{Z} \cup \{\infty\}$, an EV^+ MDD encoding f is a DAG with labelled edges such that:

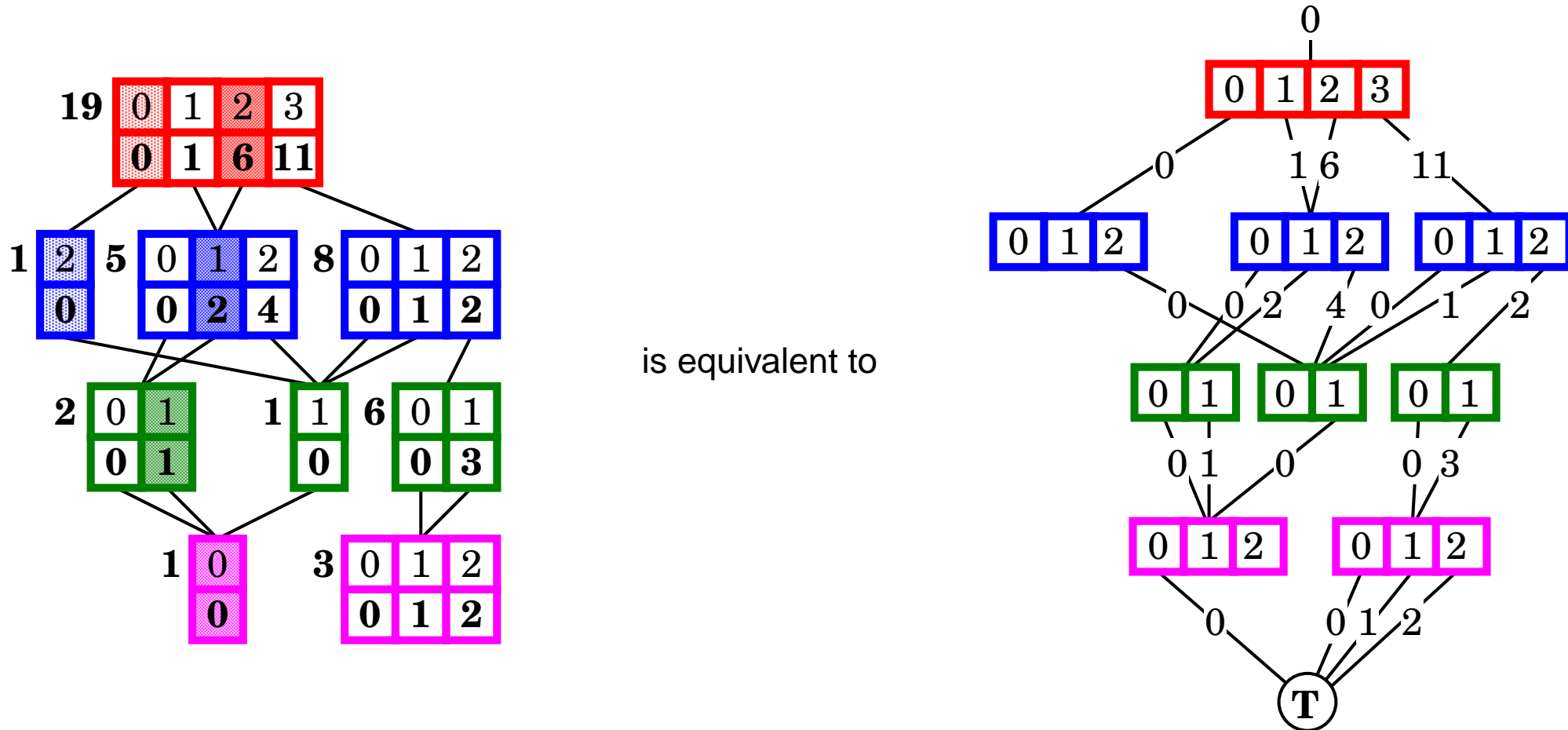
- There is a single terminal node $\langle 0 | \mathbf{T} \rangle$
- There is a single root $\langle K | r \rangle$ with a “dangling” incoming edge having value $\rho \in \mathbb{Z} \cup \{\infty\}$
- Non-terminal node $\langle k | p \rangle$ has n_k edges, $\langle k | p \rangle[\mathbf{i}_k].child$ has value $\langle k | p \rangle[\mathbf{i}_k].val \in \mathbb{N} \cup \{\infty\}$
- If $\langle k | p \rangle[\mathbf{i}_k].val = \infty$, the value of $\langle k | p \rangle[\mathbf{i}_k].child$ is irrelevant
- If $\langle k | p \rangle[\mathbf{i}_k].val \in \mathbb{N}$, $\langle k | p \rangle[\mathbf{i}_k].child$ is the index of a node at level $k - 1$
- Each non-terminal node has at least one outgoing edge labelled with 0 (if not all ∞)
- All nodes are unique (taking into account both $\langle k | p \rangle[\mathbf{i}_k].child$ and $\langle k | p \rangle[\mathbf{i}_k].val$)

analogous definition uses real values instead of integers

Theorem: EV^+ MDDs are canonical

Using an EV⁺MDD to store the indexing function ψ

The “MDD with offsets” used to store and evaluate ψ can be formalized as an EV⁺MDD



lexicographic order for $\mathbf{i} \in \mathcal{S}$

$$\psi(\mathbf{i}) = \infty \Leftrightarrow \mathbf{i} \notin \mathcal{S}$$

EV*MDDs by example

One way to think about EV*MDDs is “**EV⁺MDD** = $-\log(\text{EV}^*\text{MDD})$ ”:

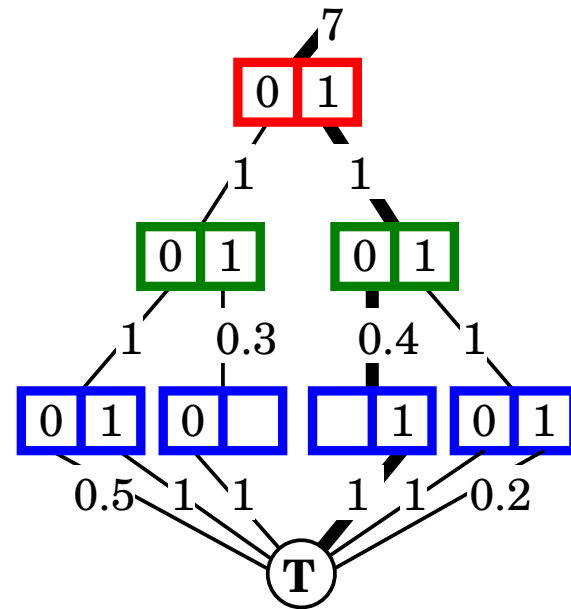
$$0 \Leftrightarrow 1$$

$$\text{edge values} \in [0, +\infty] \Leftrightarrow \text{edge values} \in [0, 1]$$

$$\text{root incoming edge} \in (-\infty, +\infty] \Leftrightarrow \text{root incoming edge} \in [0, +\infty)$$

$$\text{values add along the path} \Leftrightarrow \text{values multiply along the path}$$

i_3	0	0	0	0	1	1	1	1
i_2	0	0	1	1	0	0	1	1
i_1	0	1	0	1	0	1	0	1
f	3.5	7	2.1	0	0	2.8	7	1.4



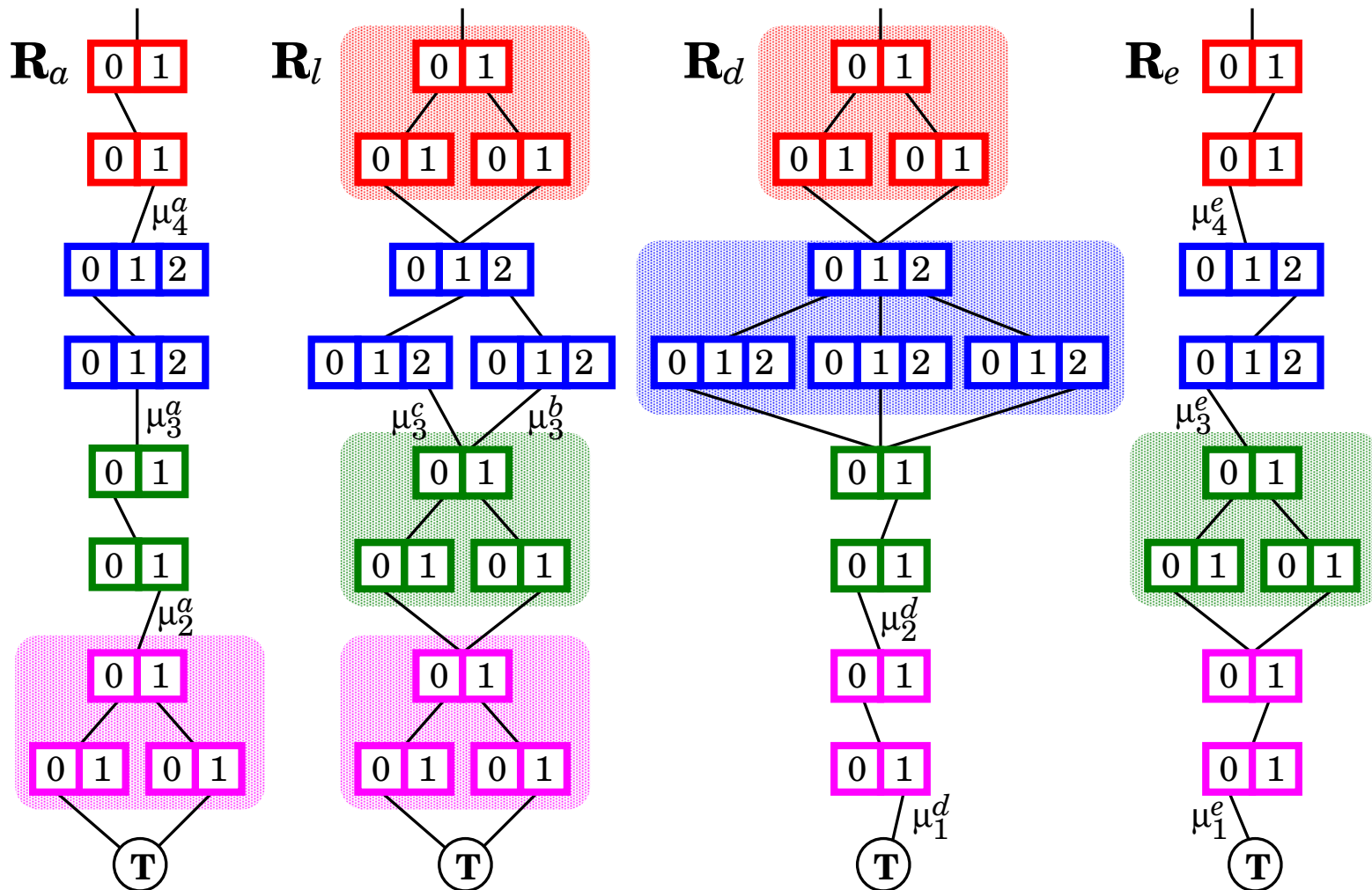
Canonicity: *all edge values are in $[0, 1]$ and at least one is 1*

In canonical form, the root incoming edge has value $\max_{i \in \hat{S}} f(i)$

Encoding \mathbf{R} with an EV^* MDD: initial non-canonical EV^* MDDs

We can store \mathbf{R} with a $2K$ -level EV^* MDD: consider the example of Kronecker encoding

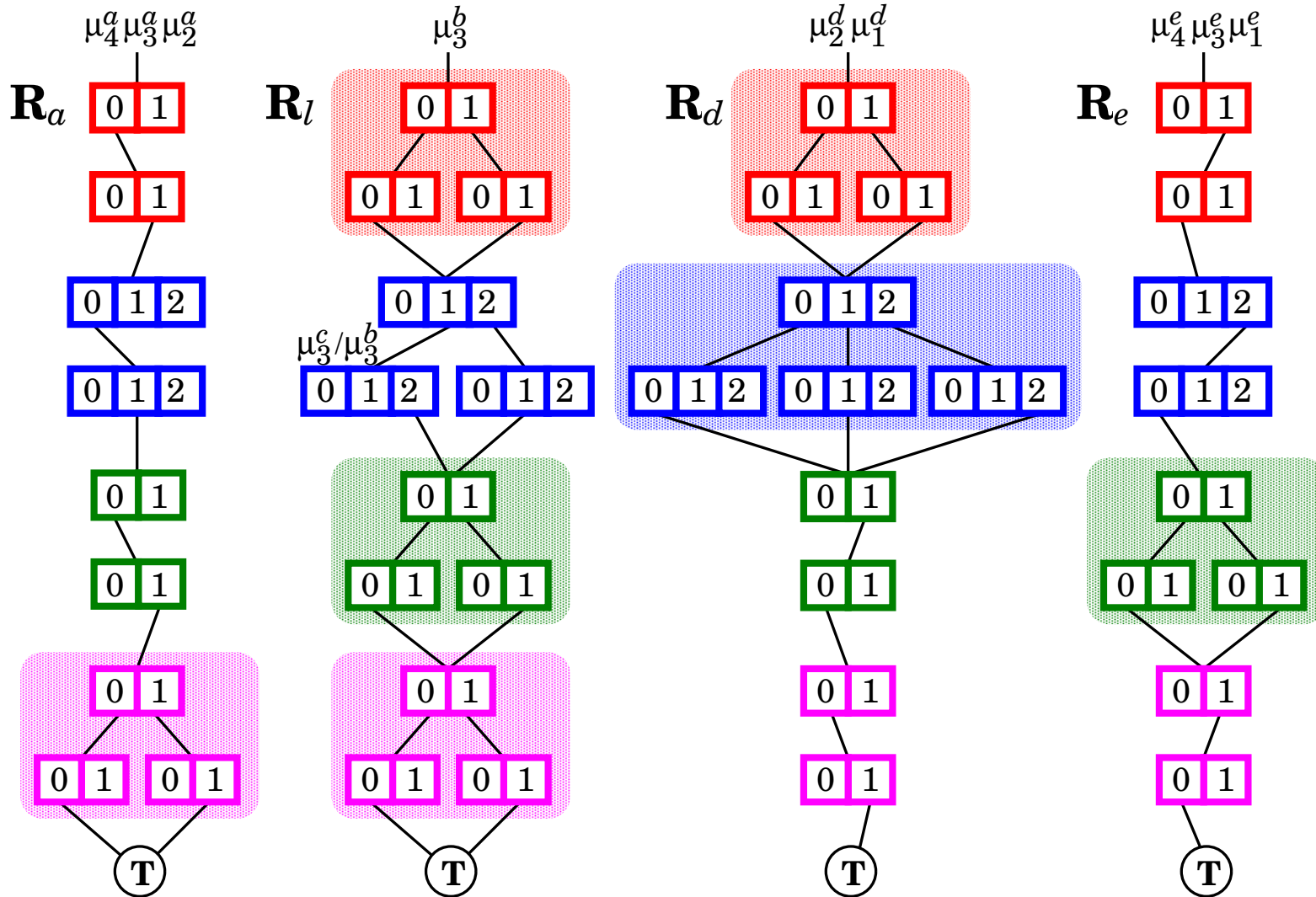
$$\mathbf{R} = \sum_{\alpha \in \{a,l,d,e\}} \mathbf{R}_\alpha = \sum_{\alpha \in \{a,l,d,e\}} \bigotimes_{4 \geq k \geq 1} \mathbf{R}_{k,\alpha}$$



note the shaded identity patterns!!!

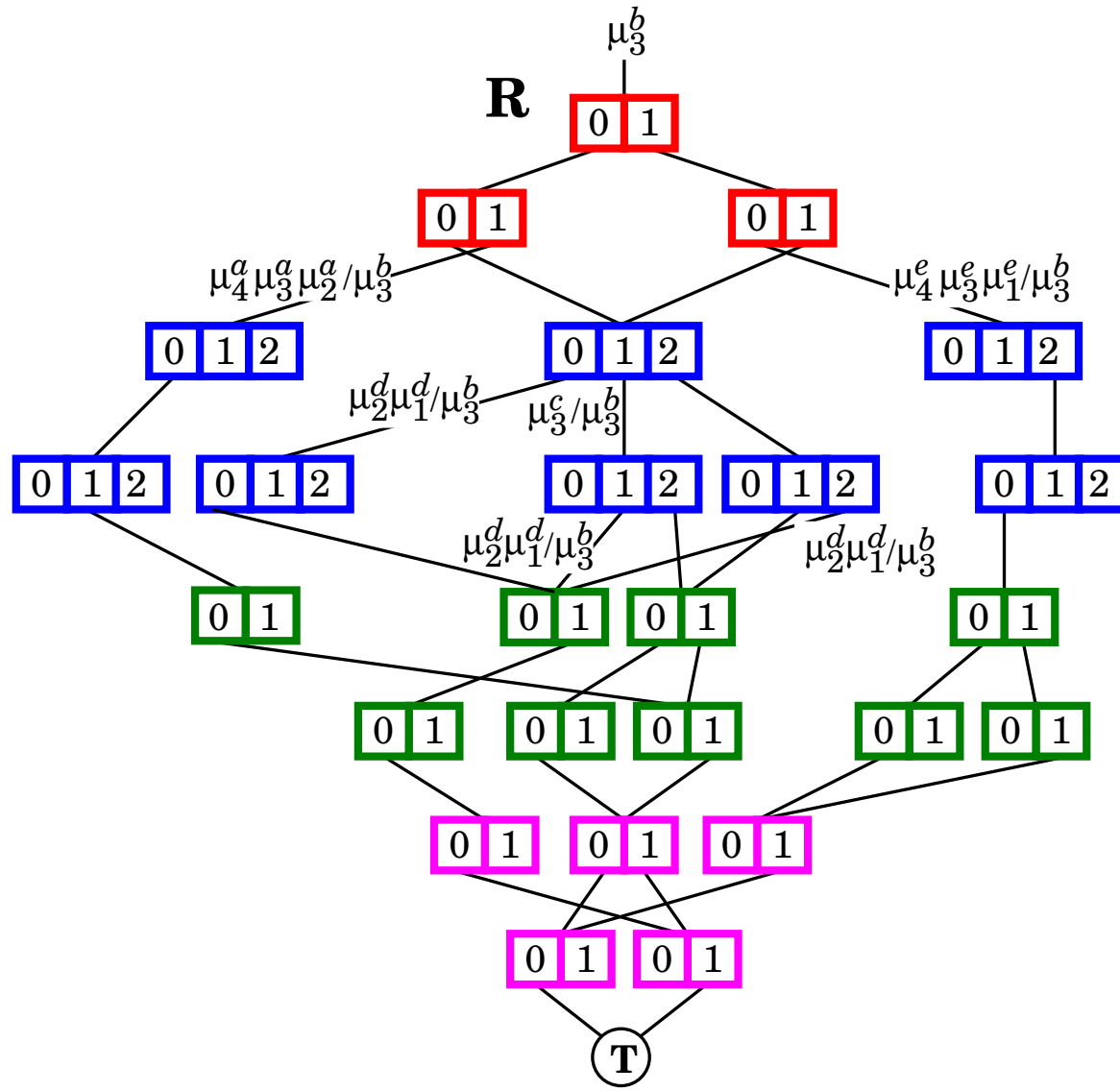
Encoding \mathbb{R} with an EV^* MDD: canonical EV^* MDDs

(assume that μ_3^b is the largest rate in \mathbb{R})



Encoding \mathbf{R} with an EV^*MDD : final EV^*MDD

Use a recursive algorithm to compute $\mathbf{R} = \sum_{\alpha \in \{a,l,d,e\}} \mathbf{R}_\alpha$



hidden identity patterns remain!!!

Empirical comparison

Memory consumption in bytes for:

S (MDD), \mathbf{R} (Sparse), $\hat{\mathbf{R}}$ (Kronecker), $\hat{\mathbf{R}}$ and \mathbf{R} (Pot/Act MXD), $\hat{\mathbf{R}}$ and \mathbf{R} (Pot/Act MTMDD)

Model	N	$ \hat{S} $	$ S $	MDD	Sparse	Kron	Pot MXD	Act MXD	Pot MTMDD	Act MTMDD
qn4	2	324	324	333	14256	772	586	722	22784	22784
	6	38416	38416	499	2524480	3092	2494	2870	36864	36864
	10	527076	527076	905	38524464	7076	5778	6522	62720	62720
qn8	2	6561	324	681	14256	1204	738	1688	43776	49152
	6	5764801	38416	1119	2524480	2404	1674	5872	55040	70912
	10	214358881	527076	1953	38524464	3604	2610	12040	66304	98560
mserv2	3	1485	495	705	23352	4124	3246	3952	34560	40704
	6	6345	2115	3176	111408	17468	13998	16432	111104	135168
	10	18495	6165	8846	342720	52228	42278	49032	306560	378460
mserv4	3	14256	495	1174	23352	5568	4098	4916	68864	79616
	6	106596	2115	8453	111408	22920	17502	20054	254360	298856
	10	488268	6165	33739	342720	67560	52342	58934	873896	998552
mserv6	3	32076	495	1333	23352	5724	4066	5316	86784	101376
	6	239841	2115	8614	111408	23076	17470	20238	298596	347956
	10	1098603	6165	33900	342720	67716	52310	59118	982396	1112684

Model	N	$ \hat{S} $	$ S $	MDD	Sparse	Kron	Pot MXD	Act MXD	Pot MTMDD	Act MTMDD
molloy4	5	4536	91	660	4204	1316	1148	2534	23552	28160
	8	32805	285	1215	14676	2528	2300	5216	27648	38656
	10	87846	506	1766	27104	3556	3288	7504	31232	47360
molloy5	5	7776	91	846	4204	1100	792	4298	28416	37120
	8	59049	285	1545	14676	1592	1188	9356	31232	50944
	10	161051	506	2223	27104	1920	1452	13778	33280	61952
kanban3	1	160	160	264	8032	500	412	544	18432	18432
	3	58400	58400	937	5590400	7572	6786	8134	66816	67072
	5	2546432	2546432	5646	303705920	45660	41816	48780	303776	303776
kanban4	1	256	160	332	8032	420	354	602	23552	24576
	3	160000	58400	628	5590400	2500	2216	3284	44032	50176
	5	9834496	2546432	1532	303705920	7940	7118	9950	92928	110592
kanban16	1	65536	160	1275	8032	2148	866	3000	95232	107520
	3	Overflow	58400	1902	5590400	3236	1746	10566	115456	151808
	5	Overflow	2546432	3149	303705920	4324	2626	24106	135168	216320
fms5	1	2100	84	535	3228	1456	604	1808	36096	40960
	3	9432500	20600	3294	1554080	8304	5224	24320	151296	247040
	5	2016379008	852012	30490	82727748	34484	24664	138244	654892	1255108
fms21	1	4194304	84	2050	3228	3132	1132	7396	126976	148224
	3	Overflow	20600	6777	1554080	5028	2328	68762	176896	437760
	5	Overflow	852012	22038	82727748	6924	3524	255988	235008	1393932

Conclusions

MTMDDs work best when many nonzero entries in \mathbf{R} have the same value

Extensive presence of state-dependent rates can make MTMDDs inefficient

Kronecker, MXDs, and EV*MDD remain instead efficient if *the model is Kronecker-consistent*

Size of Kronecker is not affected by variable ordering

MTMDDs, MXDs, EV*MDDs, and MDDs are instead affected by variable ordering

Kronecker and MXDs are restricted to contiguous “ $(\mathbf{i}_k, \mathbf{j}_k)$ ” or “ $(\mathbf{j}_k, \mathbf{i}_k)$ ” variable ordering

MTMDDs and EV*MDDs can instead have any ordering (but is this generality useful in practice?)

Kronecker is quite efficient but more restrictive

MXDs and EV*MDDs are fully general and their size is similar to Kronecker, if it exists

MTMDDs are also fully general, their size tends to be larger than Kronecker, MXDs, and EV*MDDs

MTMDDs, MXDs, and EV*MDDs can encode \mathbf{R} instead of $\hat{\mathbf{R}}$, but memory requirements increase

Kronecker must rely on an external (MDD) representation of \mathcal{S} to zero unreachable rows

Fundamental advantage of Kronecker, MXDs over MTMDDs:

they exploit the presence of numerous identity matrices in the description of \mathbf{R} (also true when encoding just \mathcal{N} , e.g., in symbolic model-checking)

As presented, EV*MDDs do not exploit identities, but we are working on that

Solution algorithms

First algorithm to be proposed for Kronecker solution [Plateau SIGMETRICS 1985]

PSh computes $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{x}} \cdot \bigotimes_{K \geq k \geq 1} \mathbf{A}_k$

PSh⁺ computes $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{x}} \cdot \mathbf{I}_{n_K \cdots n_{k+1}} \otimes \mathbf{A}_k \otimes \mathbf{I}_{n_{k-1} \cdots n_1}$

Based on the equality [Davio 1981]

$$\bigotimes_{K \geq k \geq 1} \mathbf{A}_k = \prod_{K \geq k \geq 1} \mathbf{S}_{(n_K \cdots n_{k+1}, n_k \cdots n_1)}^T \cdot (\mathbf{I}_{|\hat{\mathcal{S}}|/n_k} \otimes \mathbf{A}_k) \cdot \mathbf{S}_{(n_K \cdots n_{k+1}, n_k \cdots n_1)}$$

where $\mathbf{S}_{(a,b)} \in \{0, 1\}^{a \cdot b \times a \cdot b}$ is an (a, b) -perfect shuffle permutation:

$$\mathbf{S}_{(a,b)}[i, j] = \begin{cases} 1 & \text{if } j = (i \bmod a) \cdot b + (i \operatorname{div} a) \\ 0 & \text{otherwise} \end{cases}$$

Requires

- K vector permutations and
- K multiplications $\mathbf{x} \cdot (\mathbf{I}_{|\hat{\mathcal{S}}|/n_k} \otimes \mathbf{A}_k)$.

Complexity of the k -th multiplication: $O(|\hat{\mathcal{S}}|/n_k \cdot \eta[\mathbf{A}_k])$.

Potential Kronecker: the shuffle algorithm PSh

```

PSh(in:  $n_K, \dots, n_1, \mathbf{A}_K, \dots, \mathbf{A}_1$ ; inout:  $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ );
1.   $n_{left} \leftarrow 1$ ;
2.   $n_{right} \leftarrow n_{K-1} \cdots n_1$ ;
3.  for  $k = K$  down to 1
4.     $base \leftarrow 0$ ;
5.     $jump \leftarrow n_k \cdot n_{right}$ ;
6.    if  $\mathbf{A}_k \neq \mathbf{I}$  then
7.      for  $block = 0$  to  $n_{left} - 1$ 
8.        for  $offset = 0$  to  $n_{right} - 1$ 
9.           $index \leftarrow base + offset$ ;
10.         for  $h = 0$  to  $n_k - 1$ 
11.            $\mathbf{z}_h \leftarrow \hat{\mathbf{x}}_{index}$ ;
12.            $index \leftarrow index + n_{right}$ ;
13.            $\mathbf{z}' \leftarrow \mathbf{z} \cdot \mathbf{A}_k$ ;
14.            $index \leftarrow base + offset$ ;
15.           for  $h = 0$  to  $n_k - 1$ 
16.              $\hat{\mathbf{y}}_{index} \leftarrow \mathbf{z}'_h$ ;
17.              $index \leftarrow index + n_{right}$ ;
18.            $base \leftarrow base + jump$ ;
19.          $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{y}}$ ;
20.          $n_{left} \leftarrow n_{left} \cdot n_k$ ;
21.          $n_{right} \leftarrow n_{right} / n_{k-1}$ ;

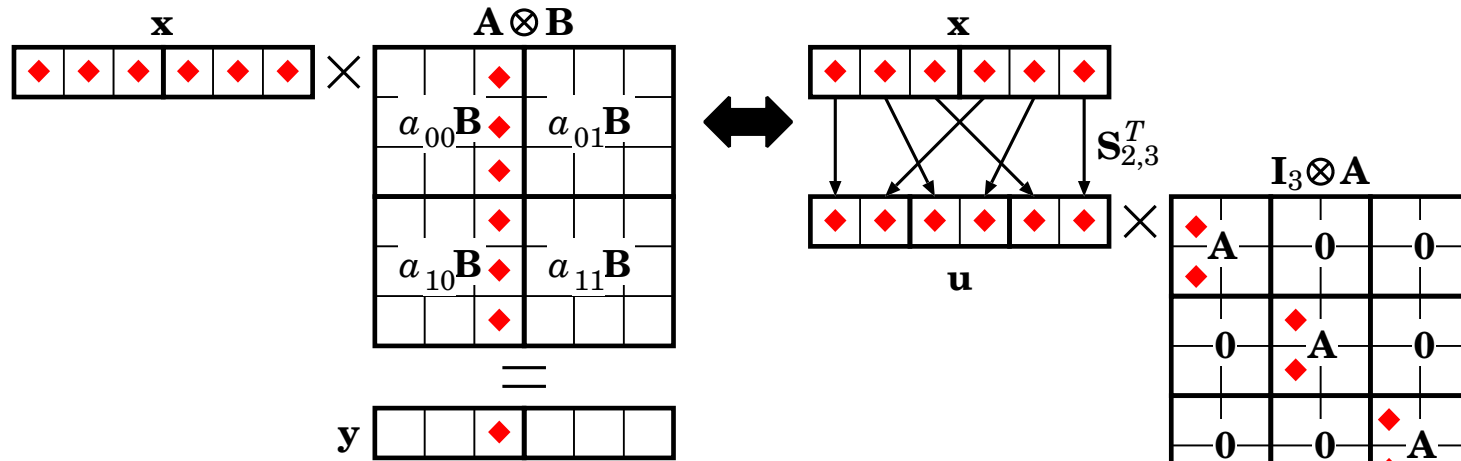
```

Let n_0 be 1

Example of shuffle computation

$$y \leftarrow x \cdot (A \otimes B)$$

Follow the entries marked with a **diamond** to obtain y_2



$$y \leftarrow \underbrace{x \cdot S_{2,3}^T}_{u} \cdot \underbrace{(I_3 \otimes A)}_v \cdot \underbrace{S_{2,3} \cdot S_{6,1}^T \cdot (I_2 \otimes B) \cdot S_{6,1}}_w = y$$

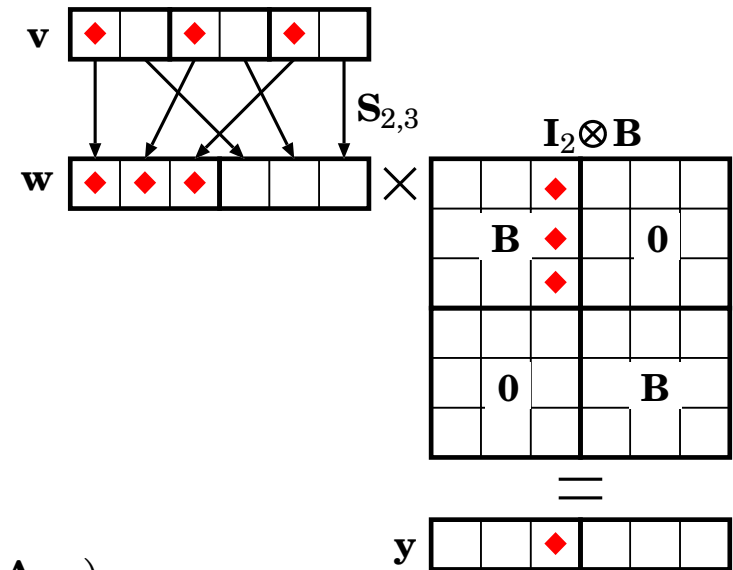
$$y_2 \leftarrow B_{02}w_0 + B_{12}w_1 + B_{22}w_2 =$$

$$B_{02}v_0 + B_{12}v_2 + B_{22}v_4 =$$

$$B_{02}(u_0A_{00} + u_1A_{10}) + B_{12}(u_2A_{00} + u_3A_{10}) + B_{22}(u_4A_{00} + u_5A_{10}) =$$

$$B_{02}(x_0A_{00} + x_3A_{10}) + B_{12}(x_1A_{00} + x_4A_{10}) + B_{22}(x_2A_{00} + x_5A_{10}) =$$

$$A_{00}B_{02}x_0 + A_{00}B_{12}x_1 + A_{00}B_{22}x_2 + A_{10}B_{02}x_3 + A_{10}B_{12}x_4 + A_{10}B_{22}x_5$$



Complexity of PSh and PSh^+

$$PSh \text{ has complexity } O \left(\sum_{K \geq k \geq 1} |\hat{\mathcal{S}}|/n_k \cdot \eta[\mathbf{A}_k] \right) = O \left(|\hat{\mathcal{S}}| \cdot K \cdot \alpha \right)$$

Even when $\hat{\mathcal{S}} = \mathcal{S}$, PSh is faster than *Ordinary* explicit multiplication only if

$$|\hat{\mathcal{S}}| \cdot K \cdot \alpha < |\hat{\mathcal{S}}| \cdot \alpha^K \quad \Leftrightarrow \quad \alpha > K^{\frac{1}{K-1}}$$

$$PSh^+ \text{ has complexity } O \left(|\hat{\mathcal{S}}|/n_k \cdot \eta[\mathbf{A}_k] \right) = O \left(|\hat{\mathcal{S}}| \cdot \alpha \right)$$

Complexity of computing $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \hat{\mathbf{x}} \cdot \bigoplus_{K \geq k \geq 1} \mathbf{A}_k$:

$$O \left(\sum_{K \geq k \geq 1} |\hat{\mathcal{S}}|/n_k \cdot \eta[\mathbf{A}_k] \right) = O \left(|\hat{\mathcal{S}}| \sum_{K \geq k \geq 1} \frac{\eta[\mathbf{A}_k]}{n_k} \right) = O \left(|\hat{\mathcal{S}}| \cdot K \cdot \alpha \right)$$

Ordinary is faster than PSh if $\alpha \leq 1$

PSh^+ saves space, but not time, w.r.t. *Ordinary*

Potential Kronecker: PRw and PRw^+

$PRwEl(\text{in: } \mathbf{i}, x, n_K, \dots, n_1, \mathbf{A}_K, \dots, \mathbf{A}_1; \text{inout: } \widehat{\mathbf{y}})$

1. for each \mathbf{j}_K s.t. $\mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K] > 0$
2. $j'_K \leftarrow \mathbf{j}_K; a_K \leftarrow \mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K];$
3. for each \mathbf{j}_{K-1} s.t. $\mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}] > 0$
4. $j'_{K-1} \leftarrow j'_K \cdot n_{K-1} + \mathbf{j}_{K-1}; a_{K-1} \leftarrow a_K \cdot \mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}];$
- ...
5. for each \mathbf{j}_1 s.t. $\mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1] > 0$
6. $j'_1 \leftarrow j'_2 \cdot n_1 + \mathbf{j}_1; a_1 \leftarrow a_2 \cdot \mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1];$
7. $\widehat{\mathbf{y}}_{j'_1} \leftarrow \widehat{\mathbf{y}}_{j'_1} + x \cdot a_1;$

$PRw(\text{in: } \widehat{\mathbf{x}}, n_K, \dots, n_1, \mathbf{A}_K, \dots, \mathbf{A}_1; \text{inout: } \widehat{\mathbf{y}})$

1. for $\mathbf{i} = 0$ to $|\widehat{\mathcal{S}}| - 1$
2. $PRwEl(i, \widehat{\mathbf{x}}_i, n_K, \dots, n_1, \mathbf{A}_K, \dots, \mathbf{A}_1, \widehat{\mathbf{y}});$

$PRwEl^+(\text{in: } n_k, n_{k-1} \cdots n_1, i_k^-, \mathbf{i}_k, i_k^+, x, \mathbf{A}_k; \text{inout: } \widehat{\mathbf{y}})$

1. for each \mathbf{j}_k s.t. $\mathbf{A}_k[\mathbf{i}_k, \mathbf{j}_k] > 0$
2. $j' \leftarrow (i_k^- \cdot n_k + \mathbf{j}_k) \cdot n_{k-1} \cdots n_1 + i_k^+;$
3. $\widehat{\mathbf{y}}_{j'} \leftarrow \widehat{\mathbf{y}}_{j'} + x \cdot \mathbf{A}_k[\mathbf{i}_k, \mathbf{j}_k];$

$PRw^+(\text{in: } \widehat{\mathbf{x}}, n_K \cdots n_{k+1}, n_k, n_{k-1} \cdots n_1, \mathbf{A}_k; \text{inout: } \widehat{\mathbf{y}})$

1. for $i \equiv (i_k^-, \mathbf{i}_k, i_k^+) = 0$ to $n_K \cdots n_{k+1} \cdot n_k \cdot n_{k-1} \cdots n_1 - 1$
2. $PRwEl^+(n_k, n_{k-1} \cdots n_1, i_k^-, \mathbf{i}_k, i_k^+, \widehat{\mathbf{x}}_i, \mathbf{A}_k, \widehat{\mathbf{y}});$

Complexity of PRw and PRw^+

PRw computes $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \hat{\mathbf{x}} \cdot \mathbf{A}$, according to the definition of Kronecker product
Requires sparse row-wise format for each \mathbf{A}_k

$PRwEl$ computes the contribution of $\hat{\mathbf{x}}_i$ to each entry of $\hat{\mathbf{y}}$ as

$$\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \hat{\mathbf{x}}_i \cdot \mathbf{A}_{i, \hat{\mathcal{S}}}$$

$PRwEl$ reaches statement $a_k \leftarrow a_{k-1} \cdot \mathbf{A}_k[\mathbf{i}_k, \mathbf{j}_k]$ $O(\alpha^k)$ times.

PRw makes $|\hat{\mathcal{S}}|$ calls to $PRwEl$, hence has complexity

$$O\left(|\hat{\mathcal{S}}| \cdot \sum_{K \geq k \geq 1} \alpha^k\right) = \begin{cases} O(|\hat{\mathcal{S}}| \cdot K) = O(K \cdot \eta[\mathbf{A}]) & \text{if } \alpha \leq 1 \\ O(|\hat{\mathcal{S}}| \cdot \alpha^K) = O(\eta[\mathbf{A}]) & \text{if } \alpha > 1 \end{cases}$$

PRw^+ has complexity $O\left(|\hat{\mathcal{S}}| \cdot \frac{\eta[\mathbf{A}_k]}{n_k}\right) = O(|\hat{\mathcal{S}}| \cdot \alpha)$

Complexity of computing $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \hat{\mathbf{x}} \cdot \bigoplus_{K \geq k \geq 1} \mathbf{A}_k$ using PRw^+ : $O(|\hat{\mathcal{S}}| \cdot K \cdot \alpha)$

PRw amortizes the multiplications for a_{K-1}, \dots, a_2 only if $\alpha \gg 1$
 PRw^+ saves space, but not time, w.r.t. *Ordinary*

Potential Kronecker: $PRwCl$ and $PRwCl^+$

$PRwCl(\text{in: } \widehat{\mathbf{x}}, n_K, \dots, n_1, \mathbf{A}_K, \dots, \mathbf{A}_1; \text{inout: } \widehat{\mathbf{y}})$

1. for $\mathbf{i}_K = 0$ to $n_K - 1$
2. for each \mathbf{j}_K s.t. $\mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K] > 0$
3. $\mathbf{j}'_K \leftarrow \mathbf{j}_K; a_K \leftarrow \mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K];$
4. for $\mathbf{i}_{K-1} = 0$ to $n_{K-1} - 1$
5. for each \mathbf{j}_{K-1} s.t. $\mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}] > 0$
6. ... $\mathbf{j}'_{K-1} \leftarrow \mathbf{j}'_K \cdot n_{K-1} + \mathbf{j}_{K-1}; a_{K-1} \leftarrow a_K \cdot \mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}];$
7. for $\mathbf{i}_1 = 0$ to $n_1 - 1$
8. for each \mathbf{j}_1 s.t. $\mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1] > 0$
9. $\mathbf{j}'_1 \leftarrow \mathbf{j}'_2 \cdot n_1 + \mathbf{j}_1; a_1 \leftarrow a_2 \cdot \mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1];$
10. $\widehat{\mathbf{y}}_{\mathbf{j}'_1} \leftarrow \widehat{\mathbf{y}}_{\mathbf{j}'_1} + \widehat{\mathbf{x}}_{\mathbf{i}_1} \cdot a_1;$

The overall complexity is $O(|\widehat{\mathcal{S}}| \cdot \alpha^K)$

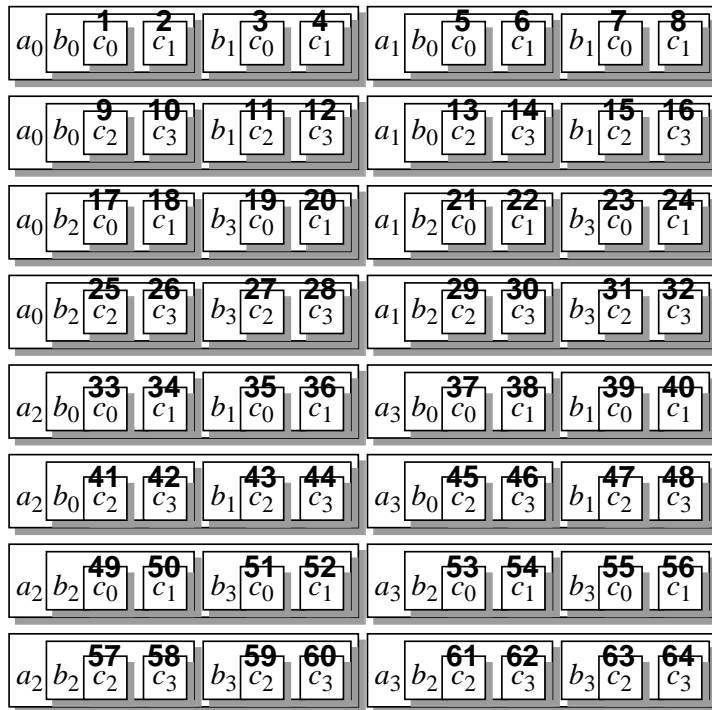
$PRwCl^+(\text{in: } \widehat{\mathbf{x}}, n_K \cdots n_{k+1}, n_k, n_{k-1} \cdots n_1, \mathbf{A}_k; \text{inout: } \widehat{\mathbf{y}})$

1. for $i_k^- = 0$ to $n_K \cdots n_{k+1} - 1$
2. for $\mathbf{i}_k = 0$ to $n_k - 1$
3. for each \mathbf{j}_k s.t. $\mathbf{A}_k[\mathbf{i}_k, \mathbf{j}_k] > 0$
4. $\mathbf{j}'_k \leftarrow i_k^- \cdot n_k + \mathbf{j}_k;$
5. for $i_k^+ = 0$ to $n_{k-1} \cdots n_1 - 1$
6. $\mathbf{j}'_K \leftarrow \mathbf{j}'_k \cdot n_{k-1} \cdots n_1 + i_k^+;$
7. $\widehat{\mathbf{y}}_{\mathbf{j}'_K} \leftarrow \widehat{\mathbf{y}}_{\mathbf{j}'_K} + \widehat{\mathbf{x}}_{(i_k^-, \mathbf{i}_k, i_k^+)} \cdot \mathbf{A}_k[\mathbf{i}_k, \mathbf{j}_k];$

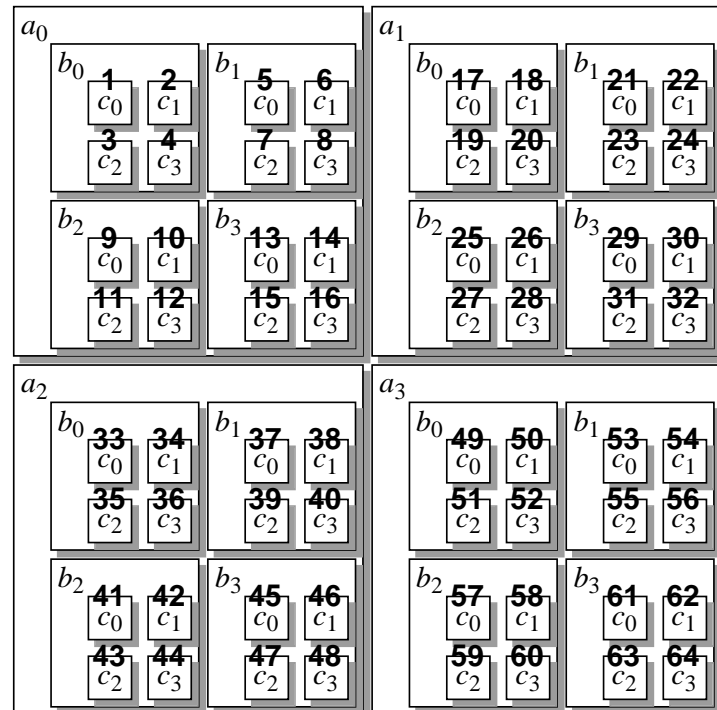
Potential Kronecker: $PRwCl$

$$\hat{\mathbf{x}} \cdot \mathbf{A} = \hat{\mathbf{x}} \cdot \left(\begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix} \otimes \begin{bmatrix} b_0 & b_1 \\ b_2 & b_3 \end{bmatrix} \otimes \begin{bmatrix} c_0 & c_1 \\ c_2 & c_3 \end{bmatrix} \right)$$

PRw



$PRwCl$



Each “ b ” and “ c ” box corresponds to one multiplication: $8 \times 8 = 64$ entries of the form $a_i b_j c_l$

- Computing each entry from scratch: $64 \times 2 = 128$ multiplications
- Using PRw : $64 + 32 = 96$ multiplications
- Using $PRwCl$: $64 + 16 = 80$ multiplications: *interleaving helps!*

same complexity as *Ordinary* regardless the sparsity level

but the entries of \mathbf{A} are not generated in column order

Actual Kronecker: ARw

$ARw(\text{in: } \mathbf{x}, \mathbf{A}_K, \dots, \mathbf{A}_1, \mathcal{S}; \text{inout: } \mathbf{y})$

1. for each $\mathbf{i} \in \mathcal{S}$
2. $I \leftarrow \psi(\mathbf{i});$
3. for each \mathbf{j}_K s.t. $\mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K] > 0$
4. $a_K \leftarrow \mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K];$
5. for each \mathbf{j}_{K-1} s.t. $\mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}] > 0$
6. $a_{K-1} \leftarrow a_K \cdot \mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}];$
- ...
7. for each \mathbf{j}_1 s.t. $\mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1] > 0$
8. $a_1 \leftarrow a_2 \cdot \mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1];$
9. $J \leftarrow \psi(\mathbf{j});$
10. $\mathbf{y}_J \leftarrow \mathbf{y}_J + \mathbf{x}_I \cdot a_1;$

Statement 9 computes the index $J = \psi(\mathbf{j})$ of state \mathbf{j} in the array \mathbf{y} .

$$O \left(|\mathcal{S}| \cdot \left(\sum_{K \geq k \geq 1} \alpha^k + \alpha^K \cdot \log |\mathcal{S}| \right) \right) = \begin{cases} O(|\mathcal{S}| \cdot (K + \log |\mathcal{S}|)) & \text{if } \alpha \leq 1 \\ O(|\mathcal{S}| \cdot \alpha^K \cdot \log |\mathcal{S}|) & \text{if } \alpha > 1 \end{cases}$$

if $K < \log |\mathcal{S}|$: ARw has a $\log |\mathcal{S}|$ overhead w.r.t. *Ordinary*

Actual Kronecker: $ARwCl$ and $ARwCl^+$

$ARwCl(\text{in: } \mathbf{x}, \mathbf{A}_K, \dots, \mathbf{A}_1, \mathcal{S}; \text{inout: } \mathbf{y})$

1. for each $\mathbf{i}_K \in \mathcal{S}_K$ *all local states \mathbf{i}_K*
2. $I_K \leftarrow \psi_K(\mathbf{i}_K);$
3. for each \mathbf{j}_K s.t. $\mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K] > 0$
4. $J_K \leftarrow \psi_K(\mathbf{j}_K);$
5. if $J_K \neq \text{null}$ then
6. $a_K \leftarrow \mathbf{A}_K[\mathbf{i}_K, \mathbf{j}_K];$
7. for each $\mathbf{i}_{K-1} \in \mathcal{S}_{K-1}(\mathbf{i}_K)$ *all \mathbf{i}_{K-1} compatible with \mathbf{i}_K*
8. $I_{K-1} \leftarrow \psi_{K-1}(I_K, \mathbf{i}_{K-1});$
9. for each \mathbf{j}_{K-1} s.t. $\mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}] > 0$
10. $J_{K-1} \leftarrow \psi_{K-1}(J_K, \mathbf{j}_{K-1});$
11. if $J_{K-1} \neq \text{null}$ then
12. $a_{K-1} \leftarrow a_K \cdot \mathbf{A}_{K-1}[\mathbf{i}_{K-1}, \mathbf{j}_{K-1}];$
- ...
 13. for each $\mathbf{i}_1 \in \mathcal{S}_1(\mathbf{i}_K, \dots, \mathbf{i}_2)$ *all \mathbf{i}_1 compatible with $\mathbf{i}_K, \dots, \mathbf{i}_2$*
 14. $I_1 \leftarrow \psi_1(I_2, \mathbf{i}_1);$
 15. for each \mathbf{j}_1 s.t. $\mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1] > 0$
 16. $J_1 \leftarrow \psi_1(J_2, \mathbf{j}_1);$
 17. if $J_1 \neq \text{null}$ then
 18. $a_1 \leftarrow a_2 \cdot \mathbf{A}_1[\mathbf{i}_1, \mathbf{j}_1];$
 19. $\mathbf{y}_{J_1} \leftarrow \mathbf{y}_{J_1} + \mathbf{x}_{I_1} \cdot a_1;$

note the need for EV^+ MDDs to index the state space

Actual Kronecker: $ARwCl$

Complexity of $ARwCl$: $O\left(\sum_{K \geq k \geq 1} |\mathcal{S}_1| \cdots |\mathcal{S}_k| \cdot \alpha^k \cdot \log n_k\right) = O(|\mathcal{S}| \cdot \alpha^K \cdot \log n_K)$

(assuming that $|\mathcal{S}_1| \cdots |\mathcal{S}_{K-1}| \ll |\mathcal{S}|$)

Complexity of $ARwCl^+$: $O(|\mathcal{S}| \cdot \alpha \cdot \log n_K)$

regardless of k , and the resulting complexity of computing

$$\mathbf{y} \leftarrow \mathbf{y} + \mathbf{x} \cdot \left(\bigoplus_{K \geq k \geq 1} \mathbf{A}_k \right)_{\mathcal{S}, \mathcal{S}}$$

using $ARwCl^+$ is $O(K \cdot |\mathcal{S}| \cdot \alpha \cdot \log n_K)$

only $\log n_K$ overhead w.r.t. *Ordinary* for any sparsity level
 but it cannot be used in a Gauss-Seidel iteration

Results for the numerical solution

Matrix diagrams achieve the same efficiency as *multiplication by blocks* ($ARwCl$)...

... but can provide access by columns as required by Gauss–Seidel

Time requirements for the Kanban model (450MHz Pentium workstation with 384 Mbytes RAM)

N	$ \mathcal{S} $	number of arcs in \mathbf{R}	MXDs		Kronecker				Explicit	
			Gauss–Seidel Iters	sec/iter	Gauss–Seidel Iters	sec/iter	Jacobi Iters	sec/iter	Gauss–Seidel Iters	sec/iter
2	4,600	28,120	40	0.11	55	0.17	134	0.09	55	0.02
3	58,400	446,400	67	1.46	97	2.56	240	1.34	97	0.34
4	454,475	3,979,850	99	12.33	149	23.69	370	11.99	149	3.04
5	2,546,432	24,460,016	139	73.09	214	147.70	527	74.09	214	18.51
6	11,261,376	115,708,992	185	336.21	289	723.30	713	359.15	—	—
7	41,644,800	450,455,040	238	1,289.91	374	2,922.80	—	—	—	—