
BDDs and MDDs

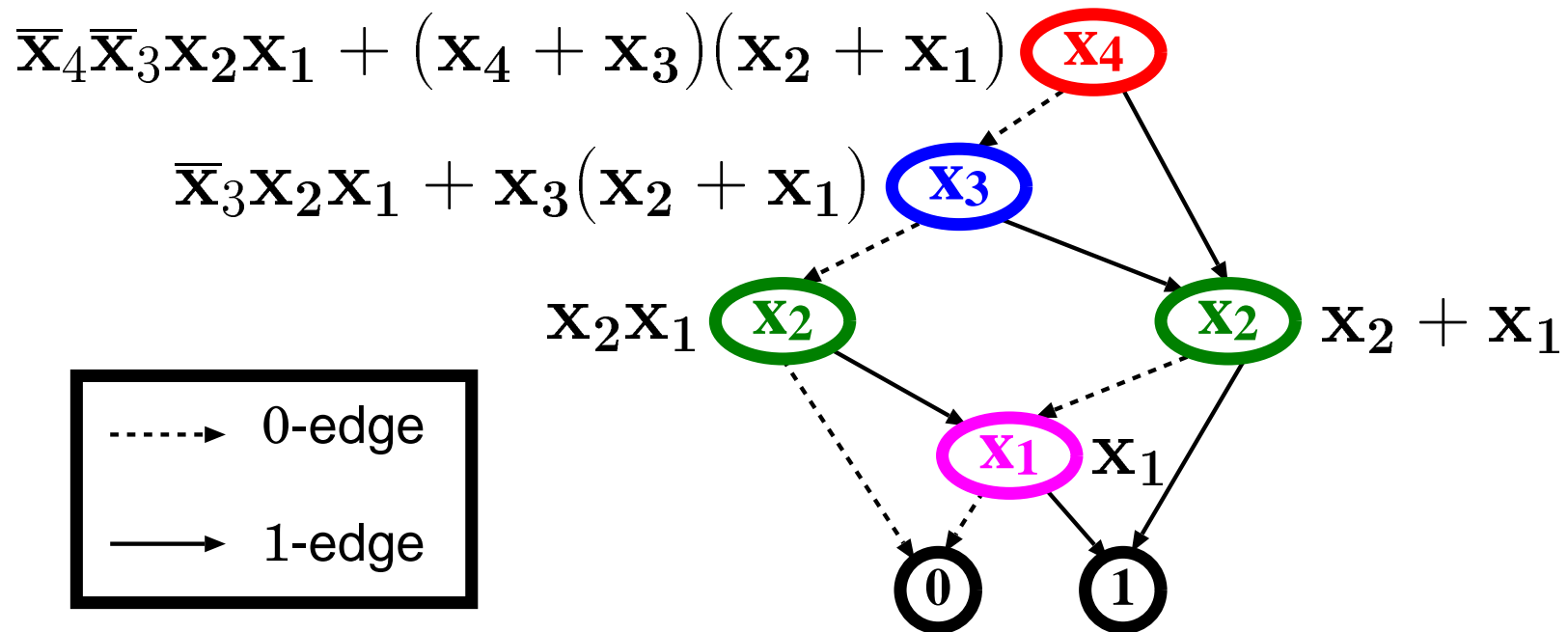
for state-space generation and symbolic model checking

“Graph-based algorithms for boolean function manipulation”

Randy Bryant (Carnegie Mellon University)

IEEE Transactions on Computers, 1986

CiteSeer most cited document!



BDDs are a **canonical** representation of boolean functions $f : \mathbb{B}^L \rightarrow \mathbb{B}$

For the **root** node, $f(x_4=0, x_3=1, x_2=1, x_1=0) = f(x_4=0, x_3=1, x_2=1, x_1=1) = 1$

A **BDD** is an acyclic directed edge-labeled graph where:

- The only **terminal** nodes can be **0** and **1**, and are at **level 0** $\mathbf{0.lvl} = \mathbf{1.lvl} = 0$
- A **nonterminal** node p is at a **level** k , with $L \geq k \geq 1$ $p.lvl = k$
- A nonterminal node p has two outgoing edges labelled 0 and 1, pointing to **children** $p[0]$ and $p[1]$
- The level of the children is lower than that of p ; $p[0].lvl < p.lvl, p[1].lvl < p.lvl$
- A node p at level k encodes the **function** $v_p : \mathbb{B}^L \rightarrow \mathbb{B}$ defined recursively by

$$v_p(x_L, \dots, x_1) = \begin{cases} p & \text{if } k = 0 \\ v_{p[x_k]}(x_L, \dots, x_1) & \text{if } k > 0 \end{cases}$$

Instead of levels, we can also talk of **variables**:

- The terminal nodes are associated with the **range variable** x_0
- A nonterminal node is associated with a **domain variable** x_k , with $L \geq k \geq 1$

For **canonical** BDDs, we further require that

- There are no **duplicates**: if $p.lvl = q.lvl$ and $p[0] = q[0]$ and $p[1] = q[1]$, then $p = q$

Then, if the BDD is **quasi-reduced**, there is **no level skipping**:

- The only **root** nodes with no incoming arcs are at level L
- The children $p[0]$ and $p[1]$ of a node p are at level $p.lvl - 1$

Or, if the BDD is **fully-reduced**, there is **maximum level skipping**:

- There are no **redundant** nodes p satisfying $p[0] = p[1]$

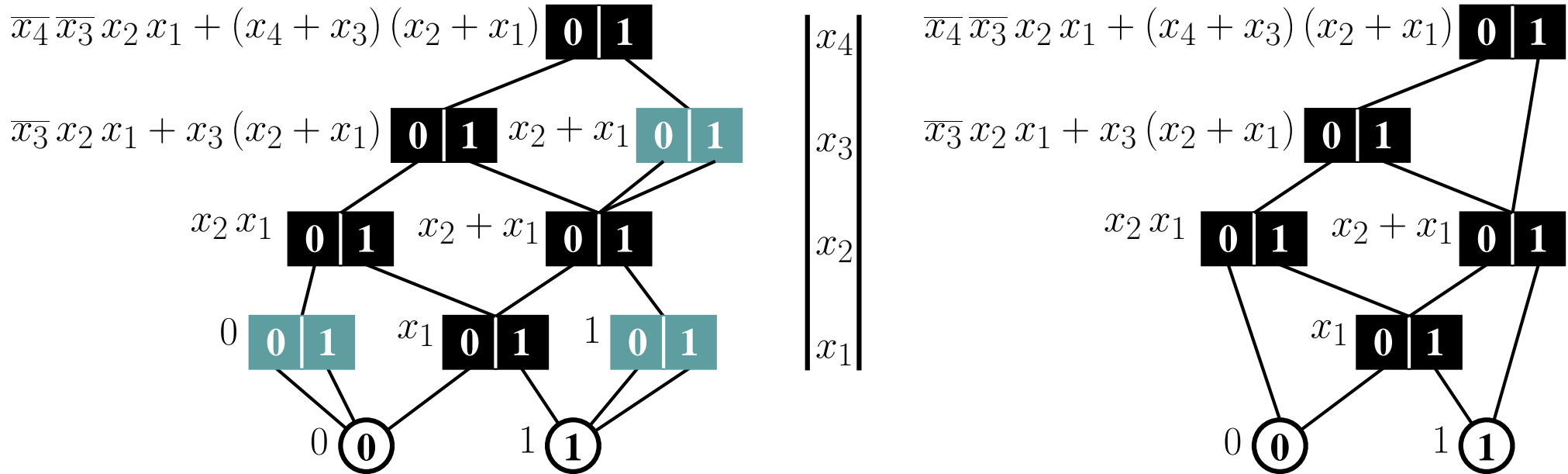
Both versions are **canonical**, if functions f and g are encoded using BDDs:

- Satisfiability, $f \neq 0$, or equivalence, $f = g$ $O(1)$
- Conjunction, $f \wedge g$, disjunction, $f \vee g$, relational product: $O(\|f\| \times \|g\|)$, if fully-reduced
 $\sum_{L \geq k \geq 1} O(\|f\|_k \times \|g\|_k)$, if quasi-reduced

$\|f\|$ = number of nodes in the BDD encoding f

$\|f\|_k$ = number of nodes at level k in the BDD encoding f

Quasi-reduced vs. fully-reduced BDDs



Fully-reduced BDDs: each node **in the BDD** encodes a different function

Quasi-reduced BDDs: each node **at a given level of the BDD** encodes a different function

- Given a boolean expression, or a function, $f : \mathbb{B}^L \rightarrow \mathbb{B}$, there is a unique BDD encoding it (for a fixed variable order x_L, \dots, x_1)
- Many functions have a **very compact BDD encoding**
- The constant functions 0 and 1 are represented by the nodes **0** and **1**, respectively
- Given the BDD encoding boolean expression f : test whether $f \equiv 0$ or $f \equiv 1$ in $O(1)$ time
- Given the BDDs encoding boolean expressions f and g : test whether $f \equiv g$ in $O(1)$ time

- The variable ordering affects the size of the BDD, consider $x_L \Leftrightarrow y_L \wedge \dots \wedge x_1 \Leftrightarrow y_1$
 - with the order $(x_L, y_L, \dots, x_1, y_1)$ $O(L)$ nodes
 - with the order $(x_L, \dots, x_1, y_L, \dots, y_1)$ $O(2^L)$ nodes
- The BDD encoding of some functions is exponentially large for any order
 - **the expression for bit 32 of the 64-bit result of the multiplication of two 32-bit integers**
- Finding the optimal ordering that minimizes the BDD size is an **NP-complete** problem

Any function $f : \mathbb{B}^L \rightarrow \mathbb{B}$ of L variables $\mathbf{x}_L, \dots, \mathbf{x}_1$ can be expressed as

- a **boolean vector** of size 2^L , or a **full binary tree** of height L
- an **expression** over $\mathbf{x}_L, \dots, \mathbf{x}_1$ using the operator set $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$, $\{\neg, \wedge\}$, or $\{\neg, \vee\}$
- a **disjunctive normal form (DNF)** expression over $\mathbf{x}_L, \dots, \mathbf{x}_1$
 $\bigvee_{d=1}^D \bigwedge_{c=1}^{C_d} t_{d,c}$ where each $t_{d,c}$ is either \mathbf{x}_k or its complement $\neg \mathbf{x}_k$, also written as $\overline{\mathbf{x}_k}$
- a **conjunctive normal form (CNF)** expression over $\mathbf{x}_L, \dots, \mathbf{x}_1$
 $\bigwedge_{c=1}^C \bigvee_{d=1}^{D_c} t_{c,d}$ where each $t_{c,d}$ is either \mathbf{x}_k or its complement $\neg \mathbf{x}_k$, also written as $\overline{\mathbf{x}_k}$
- an **expression** over $\mathbf{x}_L, \dots, \mathbf{x}_1$ using only the *ITE* operator and the constants **0** and **1**

$$ITE(f, g, h) = (f \wedge g) \vee (\neg f \wedge h)$$

- an **if-then-else normal form (INF)** expression over $\mathbf{x}_L, \dots, \mathbf{x}_1$ where variables appear only in tests, i.e., as the first parameter of *ITE*
 - $\neg f = ITE(f, 0, 1)$
 - $f \vee g = ITE(f, 1, ITE(g, 1, 0))$
 - $f \wedge g = ITE(f, ITE(g, 1, 0), 0)$
 - $f \Rightarrow g = ITE(f, ITE(g, 1, 0), 1)$
 - $f \Leftrightarrow g = ITE(f, ITE(g, 1, 0), ITE(g, 0, 1))$

Let $f[c/x]$ be the expression obtained from f by substituting variable x with the constant $c \in \mathbb{B}$

Then, $f = ITE(x, f[1/x], f[0/x])$ is the **Shannon expansion** of f with respect to x

Given an arbitrary boolean expression f over $\mathbf{x}_L, \dots, \mathbf{x}_1$, we can put it in INF as follows:

- let f_0 be $f[0/\mathbf{x}_L]$ f_0 does not contain \mathbf{x}_L
- let f_1 be $f[1/\mathbf{x}_L]$ f_1 does not contain \mathbf{x}_L
- let f_{00} be $f_0[0/\mathbf{x}_{L-1}]$ f_{00} contains neither \mathbf{x}_L nor \mathbf{x}_{L-1}
- let f_{01} be $f_0[1/\mathbf{x}_{L-1}]$ f_{01} contains neither \mathbf{x}_L nor \mathbf{x}_{L-1}
- let f_{10} be $f_1[0/\mathbf{x}_{L-1}]$ f_{10} contains neither \mathbf{x}_L nor \mathbf{x}_{L-1}
- let f_{11} be $f_1[1/\mathbf{x}_{L-1}]$ f_{11} contains neither \mathbf{x}_L nor \mathbf{x}_{L-1}
- ...

$$\bullet f = ITE(\mathbf{x}_L, \underbrace{ITE(\mathbf{x}_{L-1}, \underbrace{\dots}_{f_{11}}, \underbrace{\dots}_{f_{10}})}_{f_1}, \underbrace{ITE(\mathbf{x}_{L-1}, \underbrace{\dots}_{f_{01}}, \underbrace{\dots}_{f_{00}})}_{f_0})$$

To derive the **ordered BDD (OBDD)** corresponding to f :

- identify 0 and 1 with **terminal nodes**
- identify each f_σ , for $\sigma \in \mathbb{B}^l$, $0 \leq k < L$, with a **non-terminal node**
- let a 0-edge go from node f_σ to node $f_{\sigma 0}$ and a 1-edge go from node f_σ to node $f_{\sigma 1}$

Two expressions f_σ and f_ρ , for $\sigma, \rho \in \mathbb{B}^l$, $0 \leq k < L$, may coincide:

- their 0-edges point to the same node and their 1-edges point to the same node

If we merge, or **share these duplicate nodes**, we have a **quasi-reduced OBDD**

In addition, an expression $f_{\mathbf{i}_L \dots \mathbf{i}_k}$ may be independent of \mathbf{x}_k , i.e., $f_{\mathbf{i}_L \dots \mathbf{i}_{k+1} 0} = f_{\mathbf{i}_L \dots \mathbf{i}_{k+1} 1}$:

- the 0-edge and the 1-edge of $f_{\mathbf{i}_L \dots \mathbf{i}_k}$ point to the same node

If we bypass and remove these **redundant nodes**, we have a **reduced OBDD** or ROBDD

To ensure canonicity, all decision diagram operations use a **Unique Table** (a hash table):

- **Search key:** the node's level and sequence of children's *node_ids* **Return value:** a *node_id*
- Alternative: one UT per level, no need to store the node's level, but more fragmentation
- All (non-dead) nodes are referenced by the UT
- Collisions must be **lossless**, multiple nodes with different *node_id* may have the same *hash_val*

With the UT, we avoid duplicate nodes

To achieve polynomial complexity, all operations use an **Operation Cache** (a hash table):

- **Search key:** *OpCODE* and sequence of operands' *node_ids* **Return value:** *node_id*
- Alternative: one OC per operation type, no need to store *OpCODE*, but more fragmentation
- Before computing $OpCODE(node_id_1, node_id_2, \dots)$, we search the OC
- If the search is successful, we avoid recomputing a result
- Collisions can be either **lossless** or **lossy**

With the OC, we consider every node combination instead of every path combination

The if-then-else, or ITE, ternary operator is defined as $ITE(f, g, h) = (f \wedge g) \vee (\neg f \wedge h)$

Let $f[c/x_k]$ be the function obtained from f by substituting variable x_k with the constant $c \in \mathbb{B}$

Then, $f = ITE(x_k, f[1/x_k], f[0/x_k])$ is the Shannon expansion of f with respect to variable x_k

For any binary boolean operator \odot : $ITE(x, u, v) \odot ITE(x, y, z) = ITE(x, u \odot y, v \odot z)$

This is the basis for the recursive BDD operator *Apply*

bdd Apply(operator \odot , bdd p , bdd q) is

fully-reduced version

```

local bdd r;
1 if  $p \in \{\mathbf{0}, \mathbf{1}\}$  and  $q \in \{\mathbf{0}, \mathbf{1}\}$  then return  $p \odot q$ ;
2 if Cache contains entry  $\langle \odot, p, q : r \rangle$  then return  $r$ ;
3 if  $p.lvl = q.lvl$  then
4    $r \leftarrow UniqueTableInsert(p.lvl, Apply(\odot, p[0], q[0]), Apply(\odot, p[1], q[1]))$ ;
5 else if  $p.lvl > q.lvl$  then
6    $r \leftarrow UniqueTableInsert(p.lvl, Apply(\odot, p[0], q), Apply(\odot, p[1], q))$ ;
7 else since  $p.lvl < q.lvl$  then
8    $r \leftarrow UniqueTableInsert(q.lvl, Apply(\odot, p, q[0]), Apply(\odot, p, q[1]))$ ;
9 enter  $\langle \odot, p, q : r \rangle$  in Cache;
10 return  $r$ ;

```

bdd $Union(bdd\ p, bdd\ q)$ is

fully-reduced version

```

local bdd r;
1  if  $p = \mathbf{0}$  or  $q = \mathbf{1}$  then return  $q$ ;
2  if  $q = \mathbf{0}$  or  $p = \mathbf{1}$  then return  $p$ ;
3  if  $p = q$  then return  $p$ ;
4  if Cache contains entry  $\langle UnionCODE, \{p, q\} : r \rangle$  then return  $r$ ;
5  if  $p.lvl = q.lvl$  then
6     $r \leftarrow UniqueTableInsert(p.lvl, Union(p[0], q[0]), Union(p[1], q[1]));$ 
7  else if  $p.lvl > q.lvl$  then
8     $r \leftarrow UniqueTableInsert(p.lvl, Union(p[0], q), Union(p[1], q));$ 
9  else since  $p.lvl < q.lvl$  then
10    $r \leftarrow UniqueTableInsert(q.lvl, Union(p, q[0]), Union(p, q[1]));$ 
11  enter  $\langle UnionCODE, \{p, q\} : r \rangle$  in Cache;
12  return  $r$ ;

```

$Intersection(p, q)$ differs from $Union(p, q)$ only in the terminal cases:

<i>Union:</i>	if $p = \mathbf{0}$ or $q = \mathbf{1}$ then return q ;	<i>Intersection:</i>	if $p = \mathbf{1}$ or $q = \mathbf{0}$ then return q ;
	if $q = \mathbf{0}$ or $p = \mathbf{1}$ then return p ;		if $q = \mathbf{1}$ or $p = \mathbf{0}$ then return p ;

complexity $O(\text{product of the number of nodes in } p \text{ and } q)$

Computing the relational product symbolically

Given an L -level BDD on (x_L, \dots, x_1) rooted at p_* encoding a set $\mathcal{Y} \subseteq \hat{\mathcal{X}}$

Given a $2L$ -level BDD on $(x_L, x'_L, \dots, x_1, x'_1)$ rooted at r_* encoding a function $\mathcal{N} : \hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$

RelationalProduct (p_*, r_*) returns the root of the BDD encoding $\{\mathbf{j} : \exists \mathbf{i} \in \mathcal{Y} \wedge \mathbf{j} \in \mathcal{N}(\mathbf{i})\}$

bdd RelationalProduct(*bdd p*, *bdd2 r*) is

quasi-reduced version

local *bdd q*, q_1 , q_2 ;

1 if $p = \mathbf{0}$ or $r = \mathbf{0}$ then return $\mathbf{0}$;

2 if $p = \mathbf{1}$ and $r = \mathbf{1}$ then return $\mathbf{1}$;

3 if *Cache* contains entry $\langle \textit{RelationalProductCODE}, p, r : q \rangle$ then return q ;

4 $q_0 \leftarrow \textit{Union}(\textit{RelationalProduct}(p[0], r[0][0]), \textit{RelationalProduct}(p[1], r[1][0]))$;

5 $q_1 \leftarrow \textit{Union}(\textit{RelationalProduct}(p[0], r[0][1]), \textit{RelationalProduct}(p[1], r[1][1]))$;

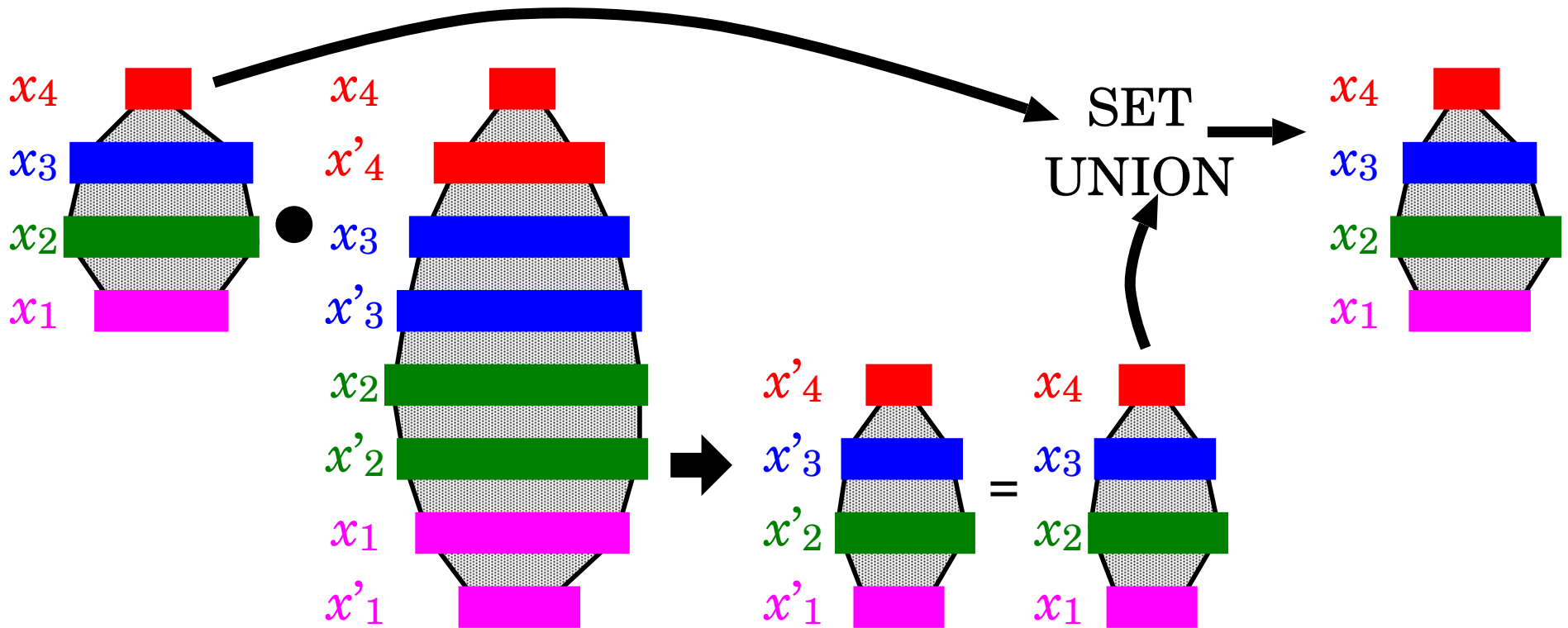
6 $q \leftarrow \textit{UniqueTableInsert}(p.lvl, q_0, q_1)$;

7 enter $\langle \textit{RelationalProductCODE}, p, r : q \rangle$ in *Cache*;

8 return q ;

BDD encoding of the next-state function

- Given a current set of states \mathcal{S} encoded as a BDD in L variables (x_L, \dots, x_1)
- Given the next-state function \mathcal{N} encoded as a BDD in $2L$ variables $(x_L, x'_L, \dots, x_1, x'_1)$
- We compute the set of states $\mathcal{N}(\mathcal{S})$ reachable from \mathcal{S} in one step



Iterations to generate the state space \mathcal{X}_{reach} : max distance d of any state from initial states + 1

Peak BDD size usually occurs well before reaching the final BDD for \mathcal{X}_{reach}

An L -level BDD encodes a set of states \mathcal{Y} as a subset of the potential state space $\hat{\mathcal{X}} = \mathbb{B}^L$

$\mathbf{i} \equiv (i_L, \dots, i_1) \in \mathcal{Y} \Leftrightarrow$ the corresponding path from the root leads to terminal 1

A $2L$ -level BDD encodes the next-state function $\mathcal{N} : \hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$

$\mathbf{j} \in \mathcal{N}(\mathbf{i}) \Leftrightarrow$ the system can go from \mathbf{i} to \mathbf{j} in one step

The state space \mathcal{X}_{reach} is the fixpoint of the iteration

$$\mathcal{X}_{init} \cup \mathcal{N}(\mathcal{X}_{init}) \cup \mathcal{N}(\mathcal{N}(\mathcal{X}_{init})) \cup \mathcal{N}(\mathcal{N}(\mathcal{N}(\mathcal{X}_{init}))) \cup \dots$$

Standard method

ExploreBdd($\mathcal{X}_{init}, \mathcal{N}$) is

```

1  $\mathcal{Y} \leftarrow \mathcal{X}_{init};$            known states
2  $\mathcal{U} \leftarrow \mathcal{X}_{init};$        unexplored states
3 repeat
4    $\mathcal{W} \leftarrow \mathcal{N}(\mathcal{U});$    potentially new states
5    $\mathcal{U} \leftarrow \mathcal{W} \setminus \mathcal{Y};$  truly new states
6    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{U};$ 
7 until  $\mathcal{U} = \emptyset;$ 
8 return  $\mathcal{Y};$ 

```

Alternative *All* method

AllExploreBdd($\mathcal{X}_{init}, \mathcal{N}$) is

```

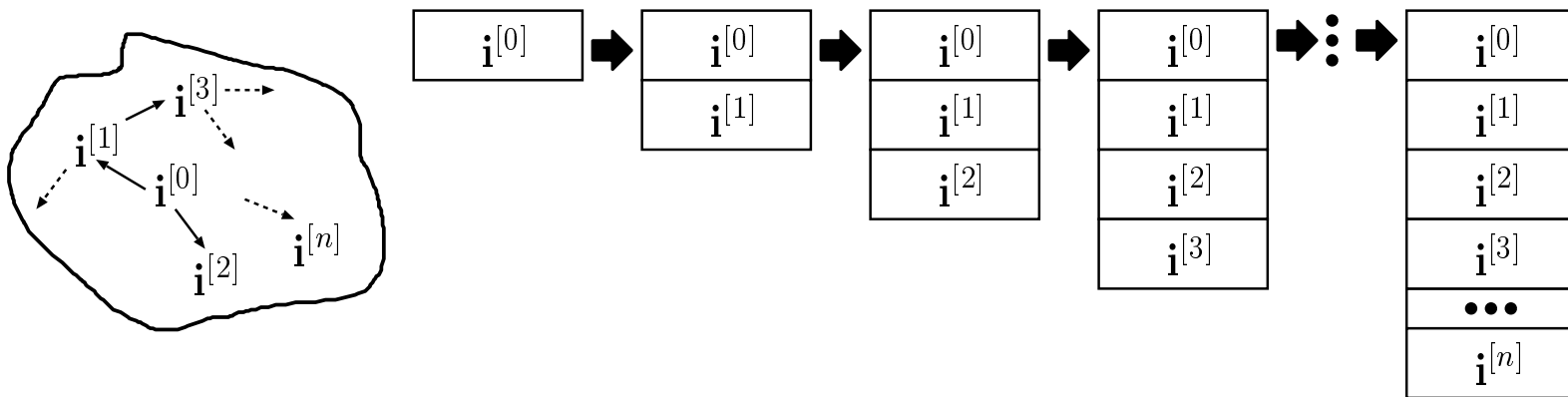
1  $\mathcal{Y} \leftarrow \mathcal{X}_{init};$ 
2 repeat
3    $\mathcal{O} \leftarrow \mathcal{Y};$            old states
4    $\mathcal{Y} \leftarrow \mathcal{O} \cup \mathcal{N}(\mathcal{O});$  new states
5 until  $\mathcal{O} = \mathcal{Y};$ 
6 return  $\mathcal{Y};$ 

```

Explicit vs. symbolic state space generation

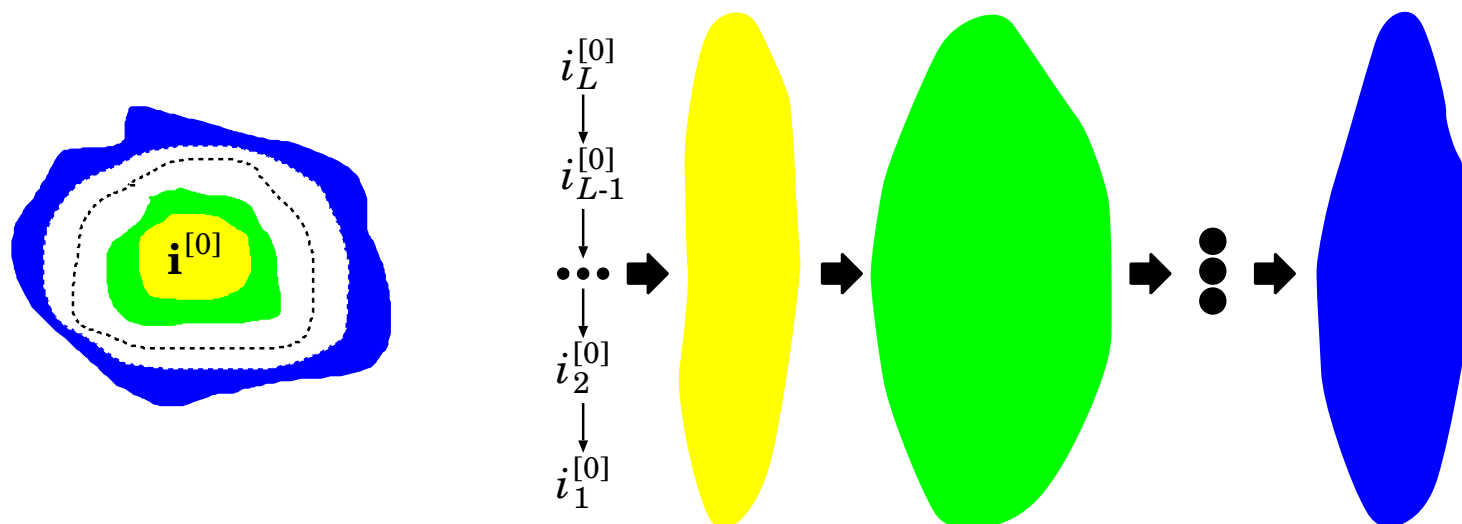
Explicit generation of the state space \mathcal{X}_{reach} adds **one state** at a time

- memory $O(\text{states})$, increases linearly, peaks at the end



Symbolic generation of the state space \mathcal{X}_{reach} with decision diagrams adds **sets of states** instead

- memory $O(\text{decision diagram nodes})$, grows and shrinks, usually peaks well before the end



Assume a **domain** $\hat{\mathcal{X}} = \mathcal{X}_L \times \cdots \times \mathcal{X}_1$, where $\mathcal{X}_k = \{0, 1, \dots, n_k - 1\}$, for some $n_k \in \mathbb{N}$

Assume the **range** $\mathcal{X}_0 = \mathbb{B}$

An MDD is an acyclic directed edge-labeled graph where:

- The only **terminal** nodes can be **0** and **1**, and are at **level 0** $\mathbf{0.lvl} = \mathbf{1.lvl} = 0$
- A **nonterminal** node p is at a **level** k , with $L \geq k \geq 1$ $p.lvl = k$
- For each $i_k \in \mathcal{X}_k$, a nonterminal node p at level k has an outgoing edge pointing to **child** $p[i_k]$
- The level of a child is lower than that of p $p[i_k].lvl < p.lvl$
- A node p at level k encodes the **function** $v_p : \hat{\mathcal{X}} \rightarrow \mathbb{B}$ defined recursively by

$$v_p(x_L, \dots, x_1) = \begin{cases} p & \text{if } k = 0 \\ v_{p[x_k]}(x_L, \dots, x_1) & \text{if } k > 0 \end{cases}$$

Instead of levels, we can also talk of **variables**:

- The terminal nodes are associated with the **range variable** x_0
- A nonterminal node is associated with a **domain variable** x_k , with $L \geq k \geq 1$

For **canonical** MDDs, we further require that

- There are no **duplicates**: if $p.lvl = q.lvl = k$ and $p[i_k] = q[i_k]$ for all $i_k \in \mathcal{X}_k$, then $p = q$

Then, if the MDD is **quasi-reduced**, there is **no level skipping**:

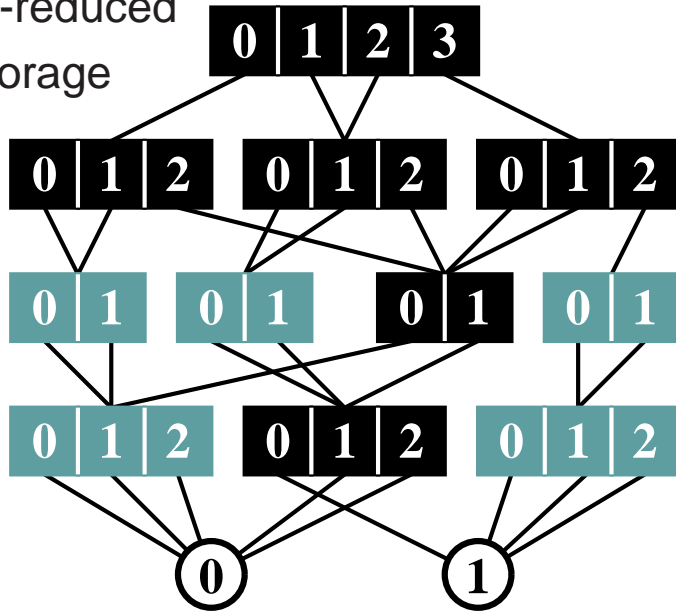
- The only **root** nodes with no incoming arcs are at level L
- If a node p is at level k , each child $p[i_k]$ is at level $k - 1$

Or, if the MDD is **fully-reduced**, there is **maximum level skipping**:

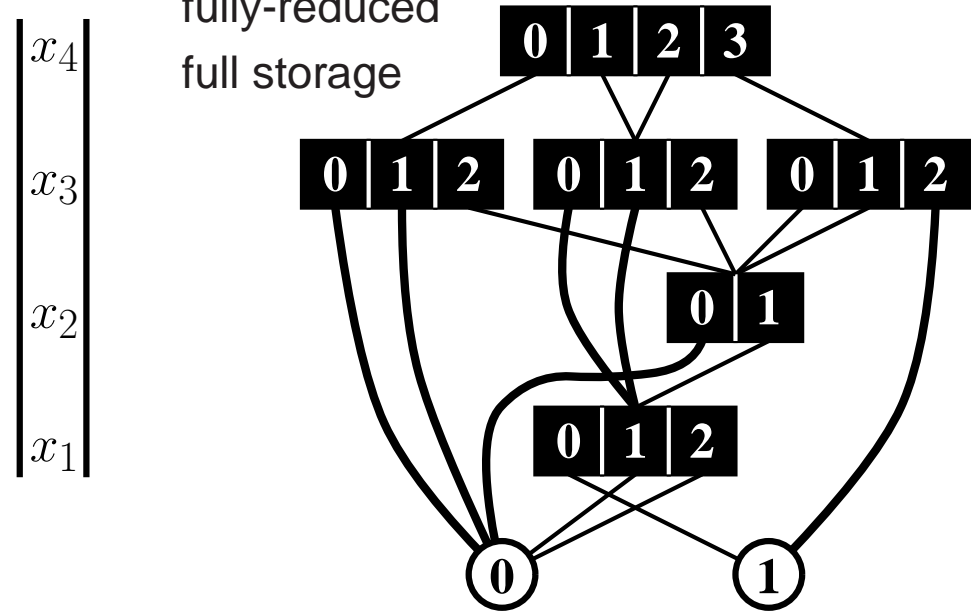
- There are no **redundant** nodes p at level k satisfying $p[i_k] = q$ for all $i_k \in \mathcal{X}_k$

Quasi-reduced vs. fully-reduced MDDs

quasi-reduced
full storage



fully-reduced
full storage



What if we don't know the range of each \mathcal{X}_k ?

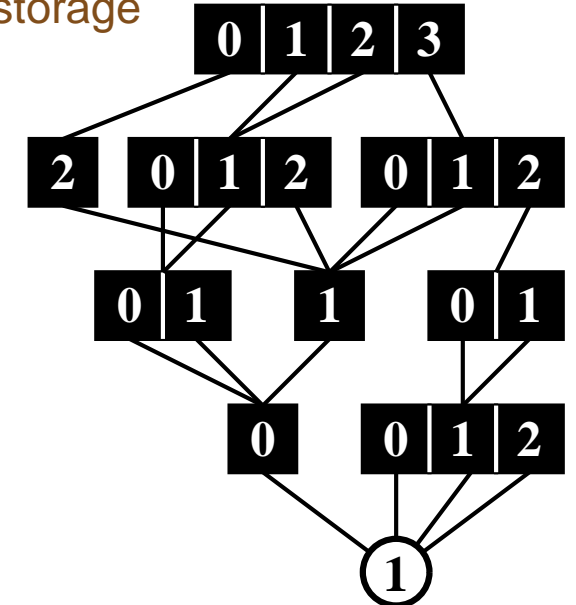
We can simply assume $\hat{\mathcal{X}} = \mathbb{N}^L$

All but a finite number of edges point to **0**

Only nodes encoding \emptyset are redundant

x_4
 x_3
 x_2
 x_1

sparse storage



Fully-reduced MDDs: each node in the MDD encodes a different function

Quasi-reduced MDDs: each node at a given level of the MDD encodes a different function

An important application of decision diagrams is to encode large sets to be manipulated **symbolically**

To encode a set $\mathcal{Y} \subseteq \hat{\mathcal{X}}$, we store its **indicator function** $f_{\mathcal{Y}}$ in a decision diagram rooted at node p :

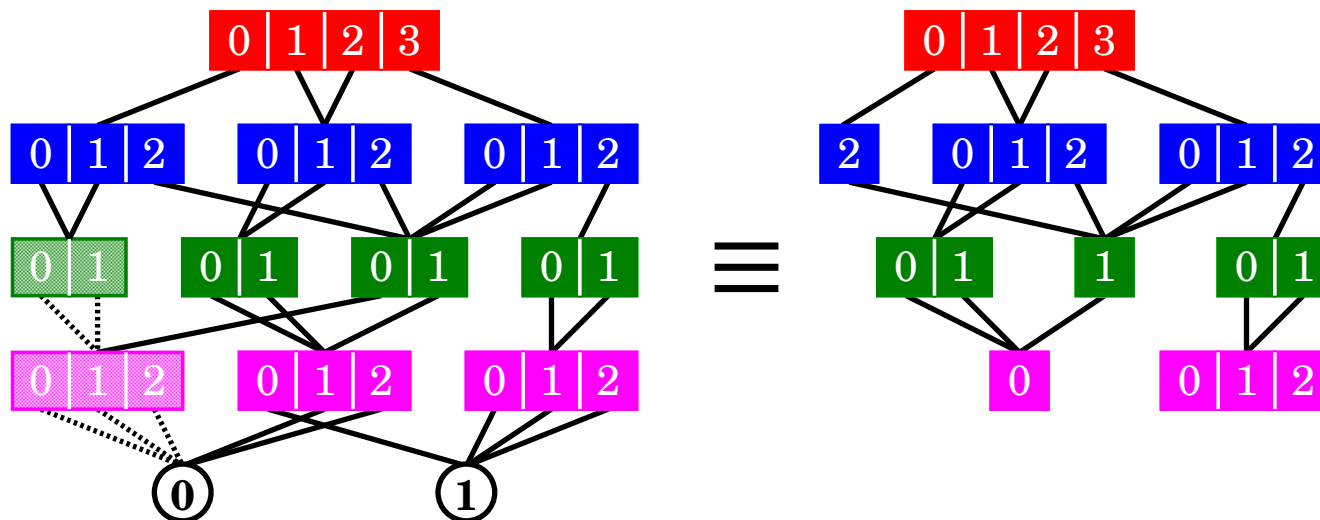
$$(i_L, \dots, i_1) \in \mathcal{Y} \Leftrightarrow f_{\mathcal{Y}}(i_L, \dots, i_1) = 1 \Leftrightarrow v_p(i_L, \dots, i_1) = 1$$

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

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

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

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



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

mdd Union(lvl k , *mdd* p , *mdd* q) is

```

local mdd  $r, r_0, \dots, r_{n_k-1}$ ;
1 if  $p = \mathbf{0}$  then return  $q$ ;
2 if  $q = \mathbf{0}$  then return  $p$ ;
3 if  $p = q$  then return  $p$ ;
4 if Cache contains entry  $\langle \text{UnionCODE}, \{p, q\} : r \rangle$  then return  $r$ ;
5 foreach  $i_k \in \mathcal{X}_k$  do
6    $r_{i_k} \leftarrow \text{Union}(k-1, p[i_k], q[i_k])$ ;
7 end for
8  $r \leftarrow \text{UniqueTableInsert}(k, r_0, \dots, r_{n_k-1})$ ;
9 enter  $\langle \text{UnionCODE}, \{p, q\} : r \rangle$  in Cache;
10 return  $r$ ;

```

Intersection(k, p, q) differs from *Union*(k, p, q) only in the terminal cases:

Union: if $p = \mathbf{0}$ then return q ;
 if $q = \mathbf{0}$ then return p ;

Intersection: if $p = \mathbf{1}$ then return q ;
 if $q = \mathbf{1}$ then return p ;

complexity:

$$O\left(\sum_{L \geq k \geq 1} \#(\text{nodes at level } k \text{ in } p) \times \#(\text{nodes at level } k \text{ in } q)\right)$$

We can store

- any set of markings $\mathcal{Y} \subseteq \hat{\mathcal{X}} = \mathbb{B}^{|\mathcal{P}|}$ of a **safe** PN with a $|\mathcal{P}|$ -level BDD
- any relation over $\hat{\mathcal{X}}$, or function $\hat{\mathcal{X}} \rightarrow 2^{\hat{\mathcal{X}}}$, such as \mathcal{N} , with a $2|\mathcal{P}|$ -level BDD

We can encode \mathcal{N} using $4 \cdot |\mathcal{E}|$ boolean functions, each corresponding to a very simple BDD

- $APM_\alpha = \prod_{p:\mathbf{F}^-p,\alpha=1} (x_p = 1)$ (all predecessor places of α are marked)
- $NPM_\alpha = \prod_{p:\mathbf{F}^-p,\alpha=1} (x_p = 0)$ (no predecessor place of α is marked)
- $ASM_\alpha = \prod_{p:\mathbf{F}^+p,\alpha=1} (x_p = 1)$ (all successor places of α are marked)
- $NSM_\alpha = \prod_{p:\mathbf{F}^+p,\alpha=1} (x_p = 0)$ (no successor place of α is marked)

The **topological image computation** for a transition α on a set of states \mathcal{U} can be expressed as

$$\mathcal{N}_\alpha(\mathcal{U}) = (((\mathcal{U} \div APM_\alpha) \cdot NPM_\alpha) \div NSM_\alpha) \cdot ASM_\alpha$$

where “ \div ” indicates the **cofactor** operator and “ \cdot ” indicates boolean conjunction

Given

- a boolean function f over (x_L, \dots, x_1)
- a literal $x_k = i_k$, with $L \geq k \geq 1$ and $i_k \in \mathbb{B}$

the cofactor $f \div (x_k = i_k)$ is defined as

- $f(x_L, \dots, x_{k+1}, i_k, x_{k-1}, \dots, x_1)$

The extension to multiple literals, $f \div (x_{k_c} = i_{k_c}, \dots, x_{k_1} = i_{k_1})$, is recursively defined as

- $f(x_L, \dots, x_{k_c+1}, i_{k_c}, x_{k_c-1}, \dots, x_1) \div (x_{k_{c-1}} = i_{k_{c-1}}, \dots, x_{k_1} = i_{k_1})$

Thus, \mathcal{N} is stored in a **disjunctively partition form** as
$$\mathcal{N} = \bigcup_{\alpha \in \mathcal{E}} \mathcal{N}_\alpha$$

If \mathcal{N} is stored in a disjunctively partitioned form as $\mathcal{N} = \bigcup_{\alpha \in \mathcal{E}} \mathcal{N}_\alpha$, using $|\mathcal{E}|$ MDDs, the effect of

$\mathcal{W} \leftarrow \mathcal{N}(\mathcal{U});$ *potentially new states*

$\mathcal{U} \leftarrow \mathcal{W} \setminus \mathcal{Y};$ *truly new states*

is exactly achieved with the statements

$\mathcal{W} \leftarrow \emptyset;$

for each $\alpha \in \mathcal{E}$ do

$\mathcal{W} \leftarrow \mathcal{W} \cup \mathcal{N}_\alpha(\mathcal{U});$

$\mathcal{U} \leftarrow \mathcal{W} \setminus \mathcal{Y};$

However, if we do not require strict breadth-first order, we can use **chaining** and do

for each $\alpha \in \mathcal{E}$ do

$\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{N}_\alpha(\mathcal{U});$

$\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{Y};$

$BfSsGen(\mathcal{X}_{init}, \{\mathcal{N}_\alpha : \alpha \in \mathcal{E}\})$

```

1  $\mathcal{Y} \leftarrow \mathcal{X}_{init};$  known states
2  $\mathcal{U} \leftarrow \mathcal{X}_{init};$  unexplored known states
3 repeat
4    $\mathcal{W} \leftarrow \emptyset;$ 
5   for each  $\alpha \in \mathcal{E}$  do
6      $\mathcal{W} \leftarrow \mathcal{W} \cup \mathcal{N}_\alpha(\mathcal{U});$ 
7    $\mathcal{U} \leftarrow \mathcal{W} \setminus \mathcal{Y};$  truly new states
8    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{U};$ 
9 until  $\mathcal{U} = \emptyset;$ 
10 return  $\mathcal{Y};$ 

```

 $ChSsGen(\mathcal{X}_{init}, \{\mathcal{N}_\alpha : \alpha \in \mathcal{E}\})$

```

1  $\mathcal{Y} \leftarrow \mathcal{X}_{init};$  known states
2  $\mathcal{U} \leftarrow \mathcal{X}_{init};$  unexplored known states
3 repeat
4   for each  $\alpha \in \mathcal{E}$  do
5      $\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{N}_\alpha(\mathcal{U});$ 
6      $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{Y};$  truly new states
7    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{U};$ 
8 until  $\mathcal{U} = \emptyset;$ 
9 return  $\mathcal{Y};$ 

```

 $AllBfSsGen(\mathcal{X}_{init}, \{\mathcal{N}_\alpha : \alpha \in \mathcal{E}\})$

```

1  $\mathcal{Y} \leftarrow \mathcal{X}_{init};$  known states
2 repeat
3    $\mathcal{O} \leftarrow \mathcal{Y};$  save old state space
4    $\mathcal{W} \leftarrow \emptyset;$ 
5   for each  $\alpha \in \mathcal{E}$  do
6      $\mathcal{W} \leftarrow \mathcal{W} \cup \mathcal{N}_\alpha(\mathcal{O});$ 
7    $\mathcal{Y} \leftarrow \mathcal{O} \cup \mathcal{W};$ 
8 until  $\mathcal{O} = \mathcal{Y};$ 
9 return  $\mathcal{Y};$ 

```

 $AllChSsGen(\mathcal{X}_{init}, \{\mathcal{N}_\alpha : \alpha \in \mathcal{E}\})$

```

1  $\mathcal{Y} \leftarrow \mathcal{X}_{init};$  known states
2 repeat
3    $\mathcal{O} \leftarrow \mathcal{Y};$  save old state space
4   for each  $\alpha \in \mathcal{E}$  do
5      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{N}_\alpha(\mathcal{Y});$ 
6 until  $\mathcal{O} = \mathcal{Y};$ 
7 return  $\mathcal{Y};$ 

```

Comparing the four approaches

N	$ \mathcal{X}_{reach} $	Time (sec)				Memory (MB)				final
		Bf	$AllBf$	Ch	$AllCh$	Bf	$AllBf$	Ch	$AllCh$	

Dining Philosophers: $L = N/2$, $|\mathcal{X}_k| = 34$ for all k

50	2.2×10^{31}	37.6	36.8	1.3	1.3	146.8	131.6	2.2	2.2	<0.1
100	5.0×10^{62}	644.1	630.4	5.4	5.3	>999.9	>999.9	8.9	8.9	<0.1
1000	9.2×10^{626}	—	—	895.4	915.5	—	—	895.2	895.0	0.3

Slotted Ring Network: $L = N$, $|\mathcal{X}_k| = 15$ for all k

5	5.3×10^4	0.2	0.3	0.1	0.1	0.8	1.1	0.3	0.2	<0.1
10	8.3×10^9	21.5	24.1	2.1	1.2	39.0	45.0	5.7	3.3	<0.1
15	1.5×10^{15}	745.4	771.5	18.5	8.9	344.3	375.4	35.1	20.2	<0.1

Round Robin Mutual Exclusion: $L = N + 1$, $|\mathcal{X}_k| = 10$ for all k except $|\mathcal{X}_1| = N + 1$

10	2.3×10^4	0.2	0.3	0.1	0.1	0.6	1.2	0.1	0.1	<0.1
20	4.7×10^7	2.7	4.4	0.3	0.3	5.9	12.8	0.5	0.5	<0.1
50	1.3×10^{17}	263.2	427.6	2.9	2.8	126.7	257.7	4.3	3.8	0.1

FMS: $L = 19$, $|\mathcal{X}_k| = N + 1$ for all k except $|\mathcal{X}_{17}| = 4$, $|\mathcal{X}_{12}| = 3$, $|\mathcal{X}_7| = 2$

5	2.9×10^6	0.7	0.7	0.1	0.1	2.6	2.2	0.4	0.2	<0.1
10	2.5×10^9	7.0	5.8	0.5	0.3	18.2	14.7	2.3	1.3	<0.1
25	8.5×10^{13}	677.2	437.9	12.9	5.1	319.7	245.3	42.7	21.2	0.1

All sets of states and relations over sets of states are encoded using DDs

An algorithm to build the DD encoding the set of states that satisfy EXp

Assume that the DD encoding the set \mathcal{P} of states satisfying p has been built already

BuildEXsymbolic(\mathcal{P}) is

1 return *RelationalProduct*($\mathcal{P}, \mathcal{N}^{-1}$); *perform one backward step in the transition relation*

Where

- \mathcal{N}^{-1} is the **inverse** or **backwards** transition relation:

$$\mathbf{i} \in \mathcal{N}^{-1}(\mathbf{j}) \iff \mathbf{j} \in \mathcal{N}(\mathbf{i})$$

- given a relation $\mathcal{R} : \mathcal{A} \rightarrow 2^{\mathcal{B}}$ and a set $\mathcal{A}' \subseteq \mathcal{A}$:

$$\textit{RelationalProduct}(\mathcal{A}', \mathcal{R}) = \mathcal{R}(\mathcal{A}') = \bigcup_{i \in \mathcal{A}'} \mathcal{R}(i) \subseteq \mathcal{B}$$

Two algorithms to build the DD encoding the set of states that satisfy $EpUq$

Assume that the DDs encoding the sets \mathcal{P} and \mathcal{Q} of states satisfying p and q have been built already

BuildEUsymbolic(\mathcal{P}, \mathcal{Q}) is

```

1  $\mathcal{Y} \leftarrow \emptyset;$ 
2  $\mathcal{U} \leftarrow \mathcal{Q};$            initialize the unexplored set  $\mathcal{U}$  with the states satisfying  $q$ 
3 repeat
4    $\mathcal{Y} \leftarrow \text{Union}(\mathcal{Y}, \mathcal{U});$            currently known states satisfying  $EpUq$ 
5    $\mathcal{W} \leftarrow \text{RelationalProduct}(\mathcal{U}, \mathcal{N}^{-1});$    perform one backward step in the transition relation
6    $\mathcal{Z} \leftarrow \text{Intersection}(\mathcal{W}, \mathcal{P});$            discard the states that do not satisfy  $p$ 
7    $\mathcal{U} \leftarrow \text{Difference}(\mathcal{Z}, \mathcal{Y});$            discard the states that are not new
8 until  $\mathcal{U} = \emptyset;$ 
9 return  $\mathcal{Y};$ 

```

BuildEUsymbolicAll(\mathcal{P}, \mathcal{Q}) is

```

1  $\mathcal{Y} \leftarrow \mathcal{Q};$            initialize the currently known result with the states satisfying  $q$ 
2 repeat
3    $\mathcal{O} \leftarrow \mathcal{Y};$            save the old set of states
4    $\mathcal{W} \leftarrow \text{RelationalProduct}(\mathcal{Y}, \mathcal{N}^{-1});$    perform one backward step in the transition relation
5    $\mathcal{Z} \leftarrow \text{Intersection}(\mathcal{W}, \mathcal{P});$            discard the states that do not satisfy  $p$ 
6    $\mathcal{Y} \leftarrow \text{Union}(\mathcal{Z}, \mathcal{Y});$            add to the currently known result
7 until  $\mathcal{O} = \mathcal{Y};$ 
8 return  $\mathcal{Y};$ 

```

An algorithm to build the DD encoding the set of states that satisfy EGp

Assume that the DDs encoding the set \mathcal{P} of states satisfying p has been built already

BuildEGsymbolic(\mathcal{P}) is

1	$\mathcal{Y} \leftarrow \mathcal{P};$	<i>initialize \mathcal{Y} with the states satisfying p</i>
2	repeat	
3	$\mathcal{O} \leftarrow \mathcal{Y};$	<i>save the old set of states</i>
4	$\mathcal{W} \leftarrow \text{RelationalProduct}(\mathcal{Y}, \mathcal{N}^{-1});$	<i>perform one backward step in the transition relation</i>
5	$\mathcal{Y} \leftarrow \text{Intersection}(\mathcal{Y}, \mathcal{W});$	
6	until $\mathcal{O} = \mathcal{Y};$	
7	return $\mathcal{Y};$	

This algorithm starts with a larger set of states and **reduces** it

This algorithm is not based on finding the strongly connected components of \mathcal{N}

Decision diagrams to encode numeric functions

Assume a **domain** $\hat{\mathcal{X}} = \mathcal{X}_L \times \cdots \times \mathcal{X}_1$, where $\mathcal{X}_k = \{0, 1, \dots, n_k - 1\}$, for some $n_k \in \mathbb{N}$

Assume a **range** $\mathcal{X}_0 = \{0, 1, \dots, n_0 - 1\}$, for some $n_0 \in \mathbb{N}$ (or an arbitrary \mathcal{X}_0 ...)

An **MTMDD** is an acyclic directed edge-labeled graph where:

- The only **terminal** nodes are values from \mathcal{X}_0 and are at **level 0** $\forall i_0 \in \mathcal{X}_0, i_0.lvl = 0$
- A **nonterminal** node p is at a **level** k , with $L \geq k \geq 1$ $p.lvl = k$
- A nonterminal node p at level k has n_k outgoing edges pointing to **children** $p[i_k]$, for $i_k \in \mathcal{X}_k$
- The level of the children is lower than that of p ; $p[0].lvl < p.lvl, p[1].lvl < p.lvl$
- A node p at level k encodes the **function** $v_p : \hat{\mathcal{X}} \rightarrow \mathcal{X}_0$ defined recursively by

$$v_p(x_L, \dots, x_1) = \begin{cases} p & \text{if } k = 0 \\ v_{p[x_k]}(x_L, \dots, x_1) & \text{if } k > 0 \end{cases}$$

Instead of levels, we can also talk of **variables**:

- The terminal nodes are associated with the **range variable** x_0
- A nonterminal node is associated with a **domain variable** x_k , with $L \geq k \geq 1$

For **canonical** MTMDDs, we further require that

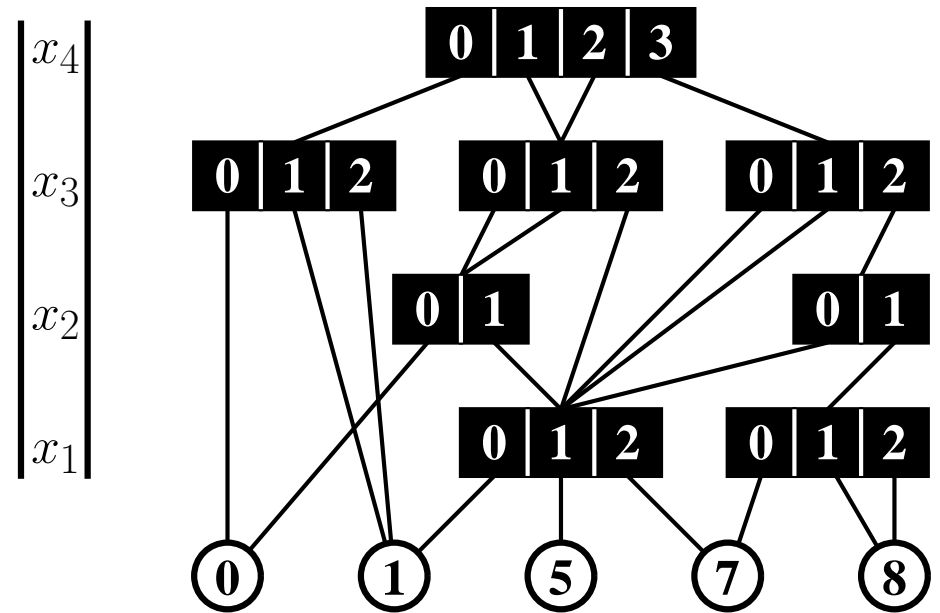
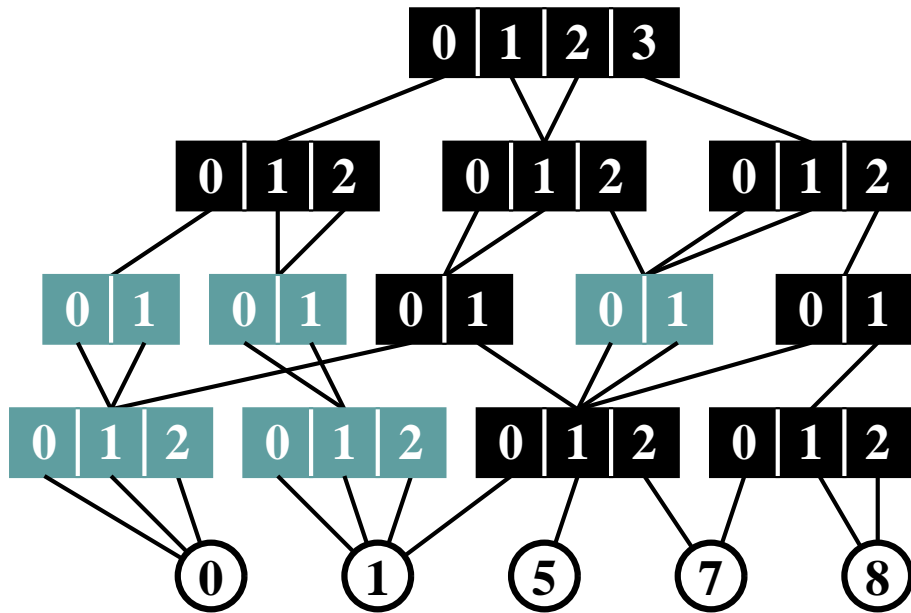
- There are no **duplicates**: if $p.lvl = q.lvl = k$ and $p[i_k] = q[i_k]$ for all $i_k \in \mathcal{X}_k$, then $p = q$

Then, if the MTMDD is **quasi-reduced**, there is **no level skipping**:

- The only **root** nodes with no incoming arcs are at level L
- Each child $p[i_k]$ of a node p is at level $p.lvl - 1$

Or, if the MTMDD is **fully-reduced**, there is **maximum level skipping**:

- There are no **redundant** nodes p satisfying $p[i_k] = q$ for all $i_k \in \mathcal{X}_k$



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

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

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

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

These MTMDDs encode a function $\hat{\mathcal{X}} = \mathcal{X}_4 \times \cdots \times \mathcal{X}_1 \rightarrow \mathbb{N}$

Assume a **domain** $\hat{\mathcal{X}} = \mathcal{X}_L \times \cdots \times \mathcal{X}_1$, where $\mathcal{X}_k = \{0, 1, \dots, n_k - 1\}$, for some $n_k \in \mathbb{N}$

Assume the **range** \mathbb{Z} and the **combinator** “+” (addition over the integers)

An EVMDD is an acyclic directed edge-labeled graph where:

- The only **terminal** node is Ω and is at **level 0** $\Omega.lvl = 0$
- A **nonterminal** node p is at a **level** k , with $L \geq k \geq 1$ $p.lvl = k$
- A nonterminal node p at level k has n_k outgoing **edges**
- For $i_k \in \mathcal{X}_k$, edge $p[i_k]$ points to **child** $p[i_k].child$, and has **value** $p[i_k].val \in \mathbb{Z}$
- The level of the children is lower than that of p $p[i_k].child.lvl < p.lvl$
- An edge $\langle \sigma, p \rangle$, with $p.lvl = k$ encodes the **function** $v_{\langle \sigma, p \rangle} : \hat{\mathcal{X}} \rightarrow \mathbb{Z}$ defined recursively by

$$v_{\langle \sigma, p \rangle}(x_L, \dots, x_1) = \begin{cases} \sigma & \text{if } k = 0, \text{ i.e., } p = \Omega \\ \sigma + v_{p[x_k]}(x_L, \dots, x_1) & \text{if } k > 0, \text{ i.e., } p \neq \Omega \end{cases}$$

For **canonical** EVMDDs, we first **normalize** each node p at level $k \geq 1$ in one of two ways:

- $p[0].val = 0$, or EVMDDs
- $p[i_k].val \geq 0$ for all $i_k \in \mathcal{X}_k$, and $p[j_k] = 0$ for at least one $j_k \in \mathcal{X}_k$ EV⁺MDDs

Then, the usual reduction requirements apply:

- There are no **duplicates**: if $p.lvl = q.lvl = k$ and $p[i_k] = q[i_k]$ for all $i_k \in \mathcal{X}_k$, then $p = q$

And, if the MDD is **quasi-reduced**, there is **no level skipping**:

- The only **root** nodes with no incoming arcs are at level L , and have **root edge values** in \mathbb{Z}
- Each child $p[i_k].child$ of a node p is at level $p.lvl - 1$

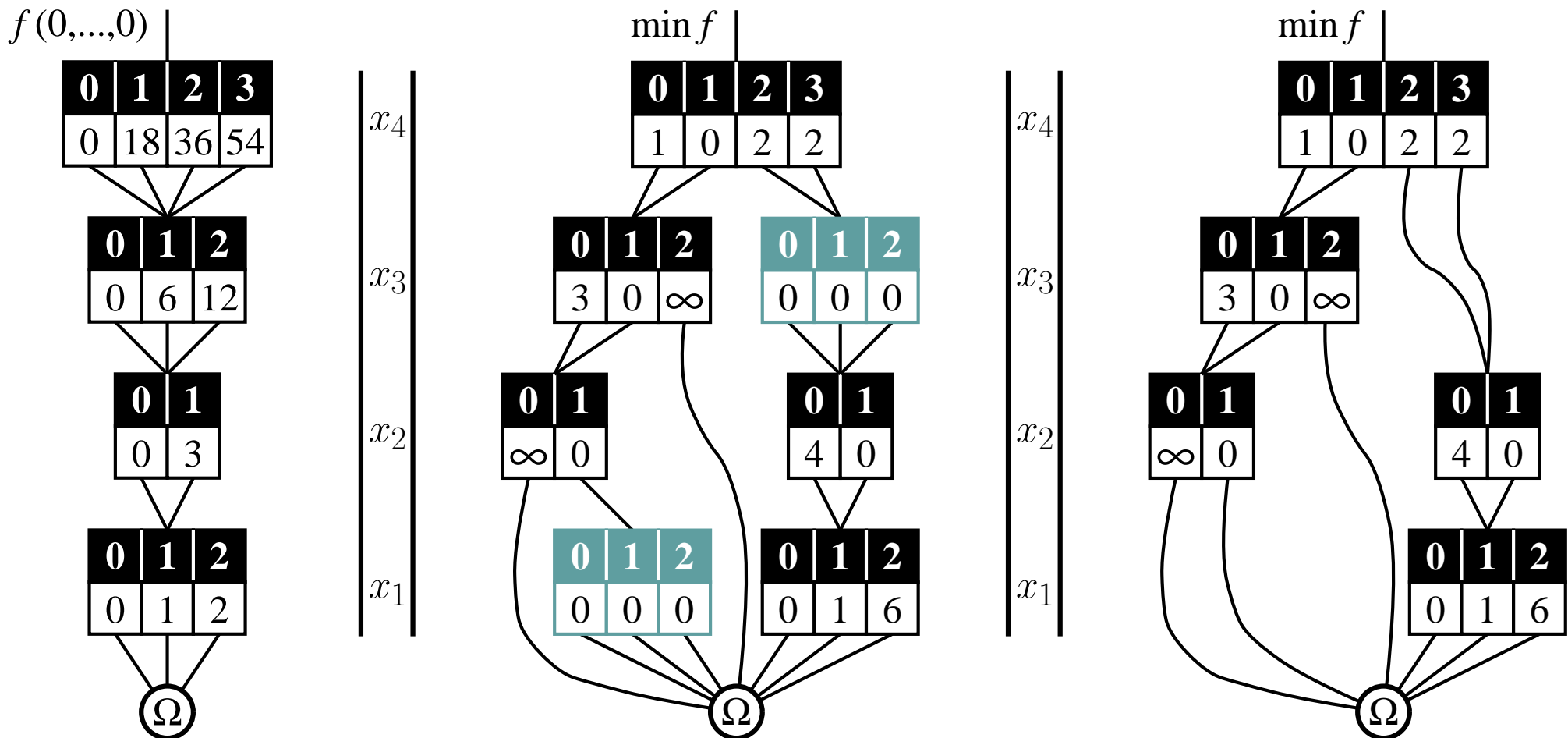
Or, if the MDD is **fully-reduced**, there is **maximum level skipping**:

- There are no **redundant** nodes p satisfying $p[i_k].child = q$ and $p[i_k].val = 0$ for all $i_k \in \mathcal{X}_k$

For EVMDDs, the value of the incoming root edge is $f(0, \dots, 0)$

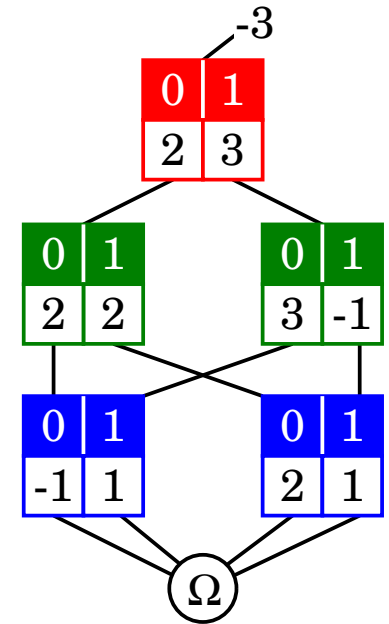
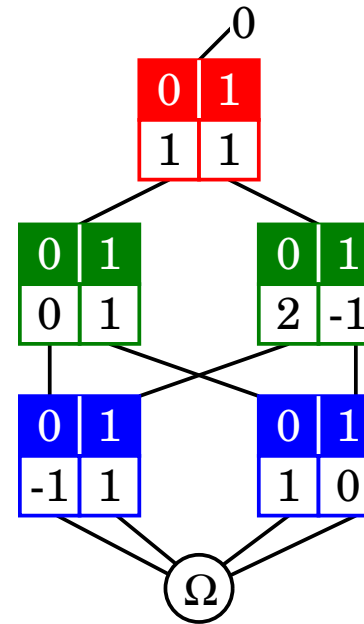
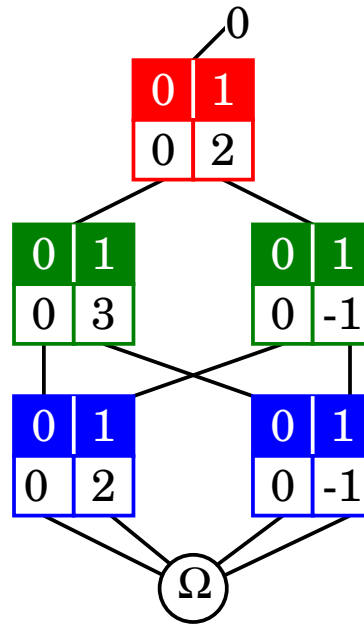
For EV^+ MDDs, the value of the incoming root edge is $\min f$

The EV^+ MDDs normalization allows to store partial functions $\hat{\mathcal{X}} \rightarrow \mathbb{Z} \cup \{\infty\}$



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

x_3	0 0 0 0 1 1 1 1
x_2	0 0 1 1 0 0 1 1
x_1	0 1 0 1 0 1 0 1
f	0 2 3 2 2 4 1 0



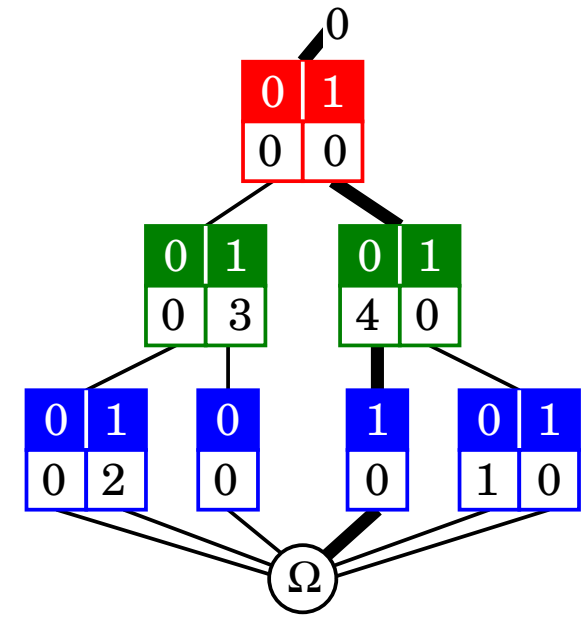
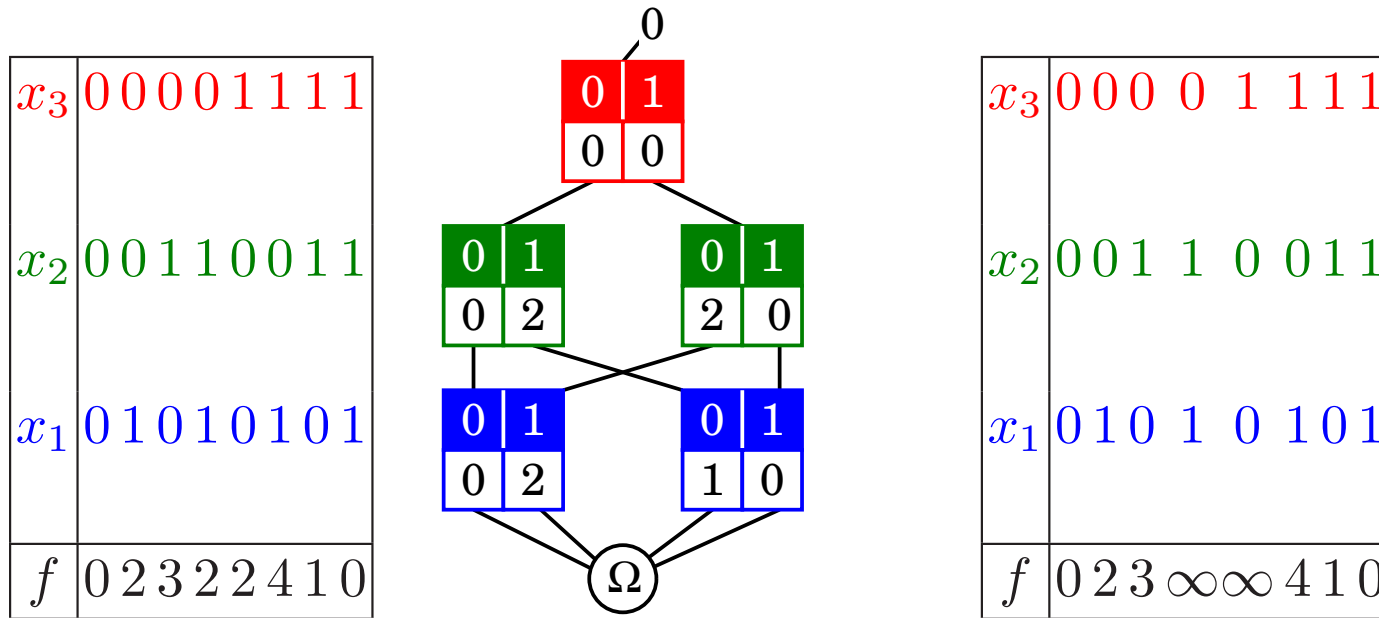
Canonicity: all nodes have a value 0 associated to the 0-arc (only the EVBDD on the left is canonical)

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

From BDD to MDD: the usual extension

∞ -edge values: can store partial arithmetic functions

Canonization rule different from that of EVBDDs: essential to encode partial arithmetic functions



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

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

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

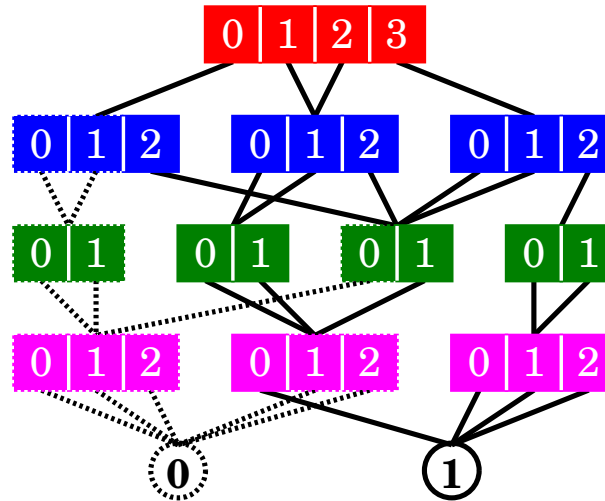
EV⁺MDDs to store the lexicographic state indexing function ψ

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

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

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

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



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

To compute the index of a state, use **edge values**:

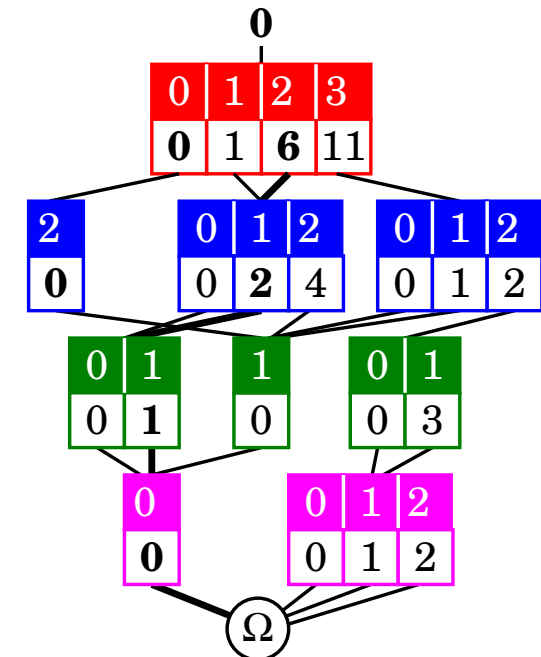
- Sum the values found on the corresponding path:

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

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

$$\psi(0, 2, 0, 0) = 0 + 0 + 0 + \infty = \infty$$

(a missing edge has the default value of ∞)



edge *Minimum*(level k , edge $\langle \alpha, p \rangle$, edge $\langle \beta, q \rangle$)

edge is a pair $\langle int, node \rangle$

local node p', q', r ;

local int μ, α', β' ;

local local i_k ;

1 if $\alpha = \infty$ then return $\langle \beta, q \rangle$;

2 if $\beta = \infty$ then return $\langle \alpha, p \rangle$;

3 $\mu \leftarrow \min\{\alpha, \beta\}$;

4 if $k = 0$ then return $\langle \mu, \Omega \rangle$;

the only node at level 0 is Ω

5 if *Cache* contains entry $\langle \textit{MinimumCODE}, k, p, q, \alpha - \beta : \gamma, r \rangle$ then return $\langle \gamma + \mu, r \rangle$;

6 $r \leftarrow \textit{NewNode}(k)$;

create new node at level k with edges set to $\langle \infty, \Omega \rangle$

7 foreach $i_k \in \mathcal{X}_k$ do

8 $p' \leftarrow p.\textit{child}[i_k]$;

9 $\alpha' \leftarrow \alpha - \mu + p.\textit{val}[i_k]$;

10 $q' \leftarrow q.\textit{child}[i_k]$;

11 $\beta' \leftarrow \beta - \mu + q.\textit{val}[i_k]$;

12 $r[i_k] \leftarrow \textit{Minimum}(k-1, \langle \alpha', p' \rangle, \langle \beta', q' \rangle)$;

continue downstream

13 *UniqueTableInsert*(k, r);

14 enter $\langle \textit{MinimumCODE}, k, p, q, \alpha - \beta : \mu, r \rangle$ in *Cache*;

15 return $\langle \mu, r \rangle$;

Decision diagrams for matrices

Assume a **domain** $\hat{\mathcal{X}} = \mathcal{X}_L \times \cdots \times \mathcal{X}_1$, where $\mathcal{X}_k = \{0, 1, \dots, n_k - 1\}$, for some $n_k \in \mathbb{N}$

Assume the **range** $\mathbb{R}^{\geq 0} = [0, +\infty)$ and the **combinator** “.” (multiplication over the reals)

An (edge-valued) MxD is an acyclic directed edge-labeled graph where:

- The only **terminal** node is Ω and is at **level 0** $\Omega.lvl = 0$
- A **nonterminal** node p is at a **level** k , with $L \geq k \geq 1$ $p.lvl = k$
- A nonterminal node p at level k has $n_k \times n_k$ outgoing edges
- For $i_k, i'_k \in \mathcal{X}_k$, edge $p[i_k, i'_k]$ points to **child** $p[i_k, jk].child$, and has **value** $p[i_k, i'_k].val \geq 0$
- The level of the children is lower than that of p $p[i_k, i'_k].child.lvl < p.lvl$
- An edge $\langle \sigma, p \rangle$, with $p.lvl = k$ encodes the **function** $v_{\langle \sigma, p \rangle} : \hat{\mathcal{X}} \rightarrow \mathbb{Z}$ defined recursively by

$$v_{\langle \sigma, p \rangle}(x_L, x'_L, \dots, x_1, x'_1) = \begin{cases} \sigma & \text{if } k = 0, \text{ i.e., } p = \Omega \\ \sigma \cdot v_{p[x_k, x'_k]}(x_L, x'_L, \dots, x_1, x'_1) & \text{if } k > 0, \text{ i.e., } p \neq \Omega \end{cases}$$

This definition of $v_{\langle \sigma, p \rangle}$ applies when no edge skips a level, otherwise we have more choices...

For **canonical** MxDs, we first **normalize** each node p in one of two ways:

- $\max\{p[i_k, i'_k].val : i_k, i'_k \in \mathcal{X}_k\} = 1$, or
- $\min\{p[i_k, i'_k].val : i_k, i'_k \in \mathcal{X}_k, p[i_k, i'_k].val \neq 0\} = 1$

Then, the usual reduction requirements apply, there are no **duplicates**:

- If $p.lvl = q.lvl = k$ and $p[i_k] = q[i_k]$ for all $i_k \in \mathcal{X}_k$, then $p = q$

And, if the MxD is **quasi-reduced**, there is **no level skipping**:

- The only **root** nodes with no incoming arcs are at level L , and have **root edge values** in \mathbb{Z}
- Each child $p[i_k, i'_k].child$ of a node p is at level $p.lvl - 1$

Or, if the MxD is **fully-reduced**, there is no **redundant node** p satisfying:

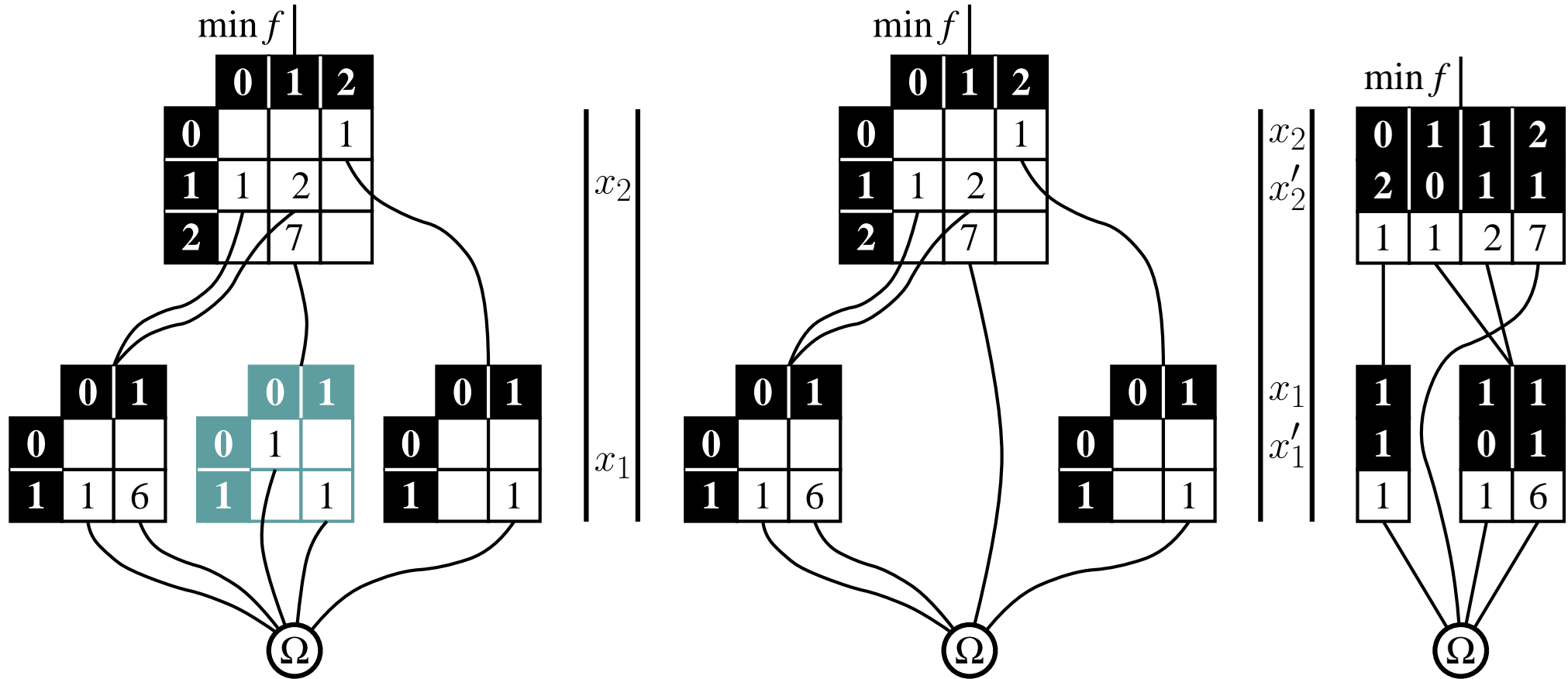
- $p[i_k, i'_k].child = q$ and $p[i_k, i'_k].val = 1$ for all $i_k, i'_k \in \mathcal{X}_k$

Or, if the MxD is **identity-reduced**, there are no **identity nodes** p satisfying:

- $p[i_k, i_k].child = q$ and $p[i_k, i_k].val = 1$ for all $i_k \in \mathcal{X}_k$
- $p[i_k, i'_k].val = 0$ for all $i_k \neq i'_k$

NEW!!!

Quasi-reduced vs. identity-reduced MxDs; sparse storage



A function $f : \widehat{\mathcal{X}} \rightarrow \mathcal{X}_0$ can be thought of as an \mathcal{X}_0 -valued one-dimensional **vector** of size $|\widehat{\mathcal{X}}|$

We often need to store functions $\widehat{\mathcal{X}} \times \widehat{\mathcal{X}} \rightarrow \mathcal{X}_0$, or **two-dimensional matrices**

We can use a decision diagram with $2L$ nonterminal levels:

- **Unprimed** x_k for the rows, or **from**, variables
- **Primed** x'_k for columns, or **to** variables
- Levels can be **interleaved**, $(x_L, x'_L, \dots, x_1, x'_1)$, or **non-interleaved**, $(x_L, \dots, x_1, x'_L, \dots, x'_1)$

We can use a (**terminal-valued**) matrix diagram (MxD)

- A non-terminal node p at level k , for $L \geq k \geq 1$, has $n_k \times n_k$ edges
- $p[i_k, i'_k]$ points to the child corresponding to the choices $x_k = i_k$ and $x'_k = i'_k$

In the matrices that we need to encode, it is often the case that the entry is 0 if $x_k \neq x'_k$

An **identity pattern** in an interleaved $2L$ -level MDD is

- a node p at level k
- with $p[i_k] = p'_{i_k}$
- such that $p'_{i_k}[i'_k] = 0$ for $i'_k \neq i_k$
- and there is a node q such that $p'_{i_k}[i_k] = q \neq 0$

In an **identity-reduced** primed level k , we skip the nodes p'_{i_k}

An **identity node** in an MxD is

- a node p
- such that $p[i_k, i'_k] = 0$ for all $i_k, i'_k \in \mathcal{X}_k, i_k \neq i'_k$
- and $p[i_k, i_k] = q$ for all $i_k \in \mathcal{X}_k$

In an **identity-reduced MxD**, we skip these identity nodes

2L-level MDDs vs. MxDs: encoding a $(3 \cdot 2) \times (3 \cdot 2)$ matrix

$$0 \equiv (x_2 = 0, x_1 = 0)$$

$$1 \equiv (x_2 = 0, x_1 = 1)$$

$$2 \equiv (x_2 = 1, x_1 = 0)$$

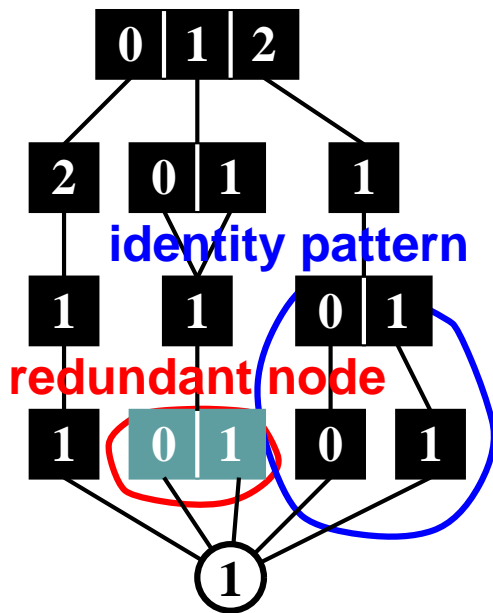
$$3 \equiv (x_2 = 1, x_1 = 1)$$

$$4 \equiv (x_2 = 2, x_1 = 0)$$

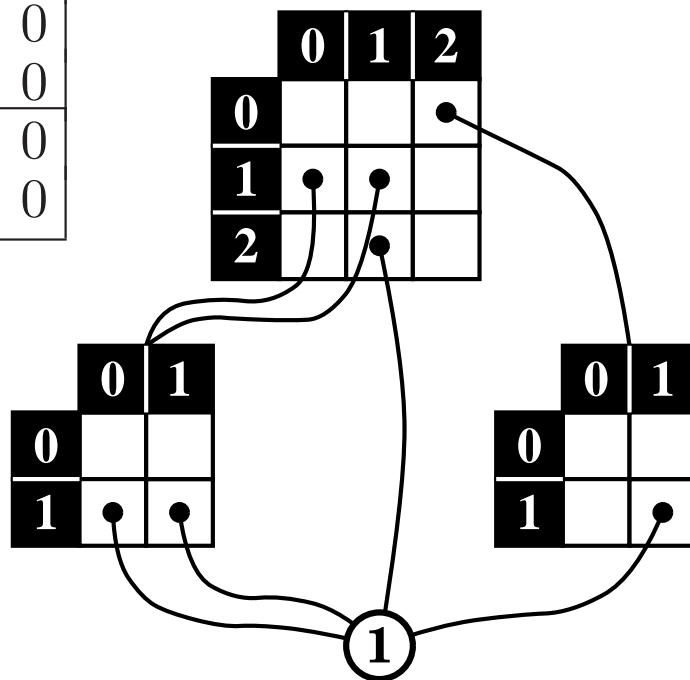
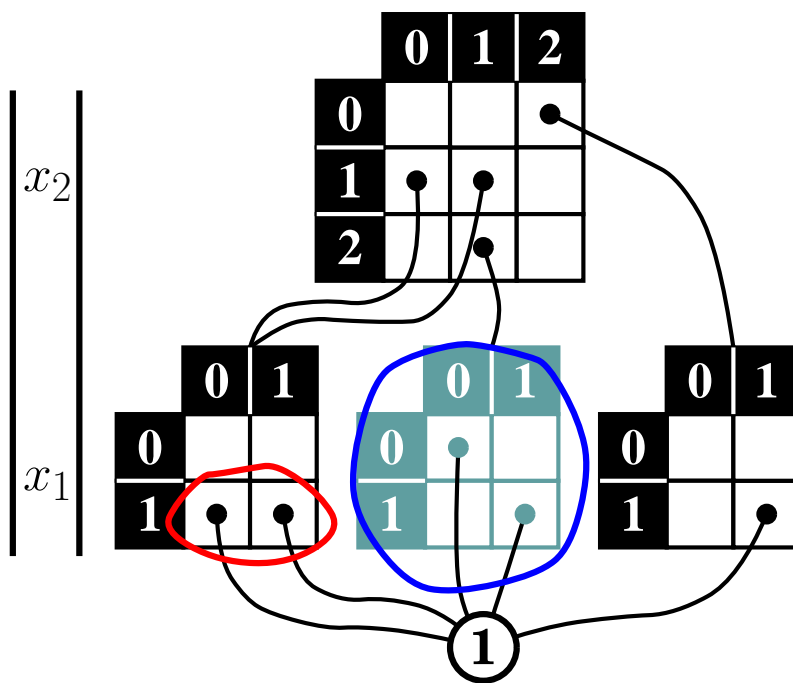
$$5 \equiv (x_2 = 2, x_1 = 1)$$

0 1 2 3 4 5

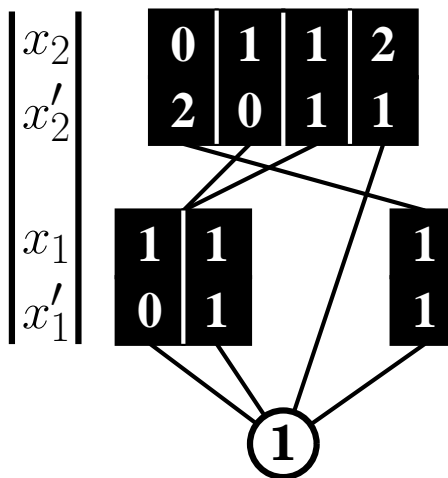
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	0	0	0
3	1	1	1	0	0
4	0	0	1	0	0
5	0	0	0	1	0



x_2
 x_2'
 x_1
 x_1'



x_2
 x_1



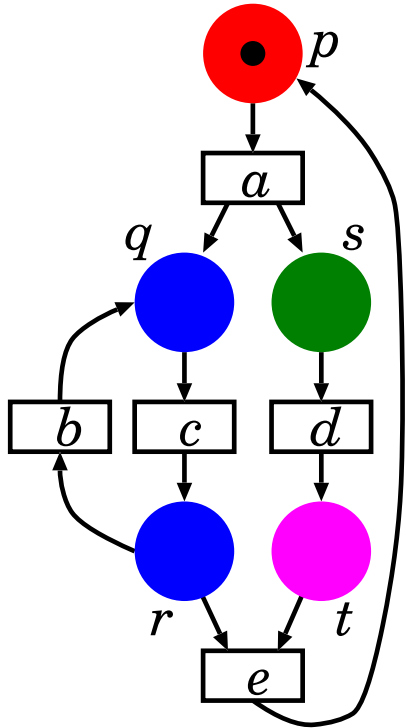
An example of MTMDD: the transition rate matrix of an SPN

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

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

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

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



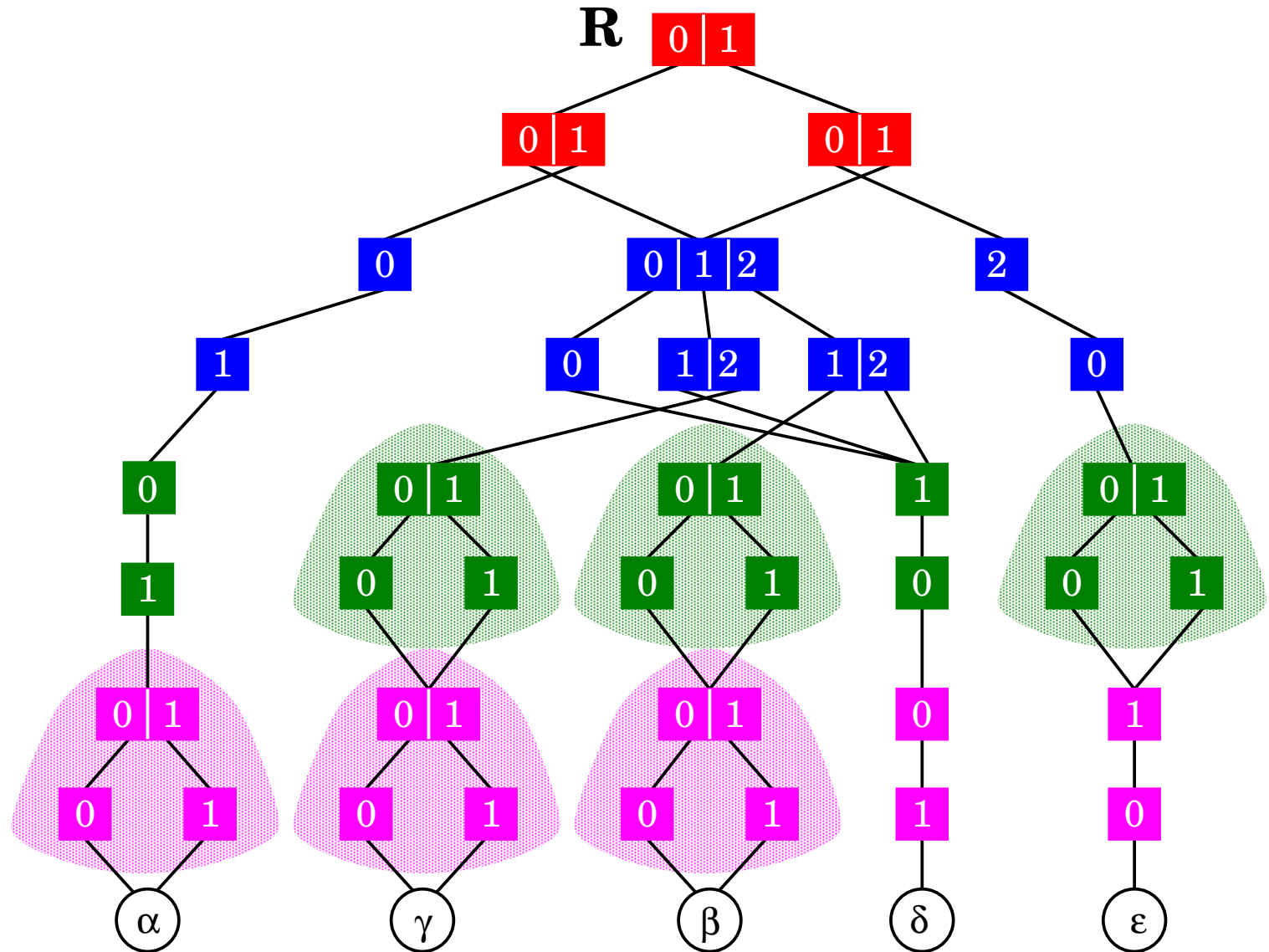
α = rate of a

β = rate of b

γ = rate of c

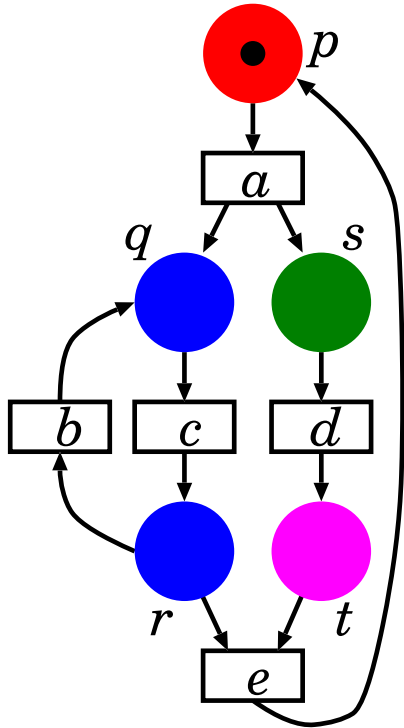
δ = rate of d

ϵ = rate of e



note the shaded identity patterns!!!

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



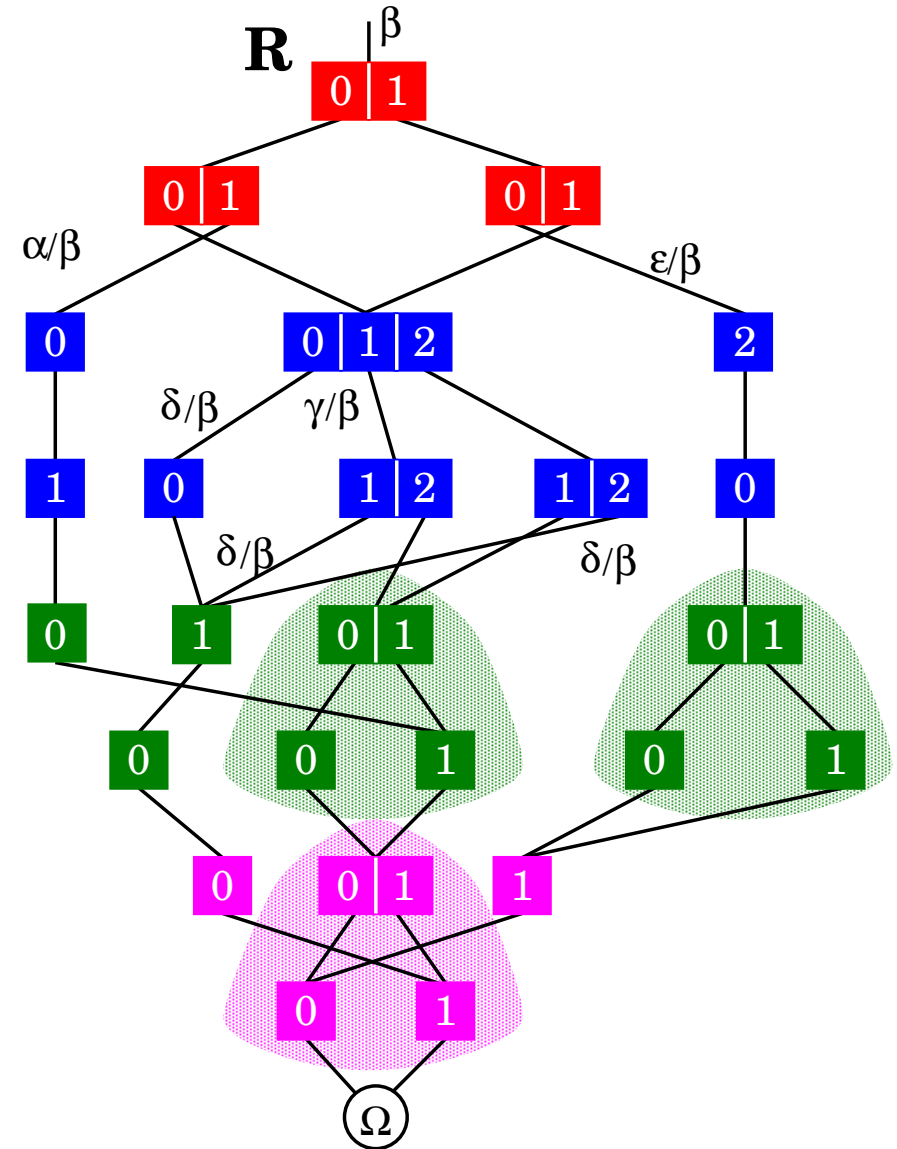
α = rate of a

β = rate of b

γ = rate of c

δ = rate of d

ϵ = rate of e



identity patterns remain!!!