A Systematic Approach to MDD-Based Constraint Programming

Samid Hoda Willem-Jan van Hoeve J.N. Hooker

Carnegie Mellon University

CP 2010

Problems with domain propagation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Domain store communicates **limited information** between constraints.
  - Other structural information learned by a constraint is lost.
  - It must be projected onto variable domains
- Domain store provides a very **coarse relaxation** of solution space.
  - Cartesian product of variable domains.
- **Imbalance** of search and inference.
  - Node processing to infer structure doesn't pay off.
  - Must process many nodes quickly.

Addressing the problems

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Replace domain store with a limited-width **multivalued decision diagram** (MDD).
  - Andersen, Hadzic, Hooker, Tiedemann (2007)
  - Domain store is a special case (width = 1).
- Propagate constraints through the MDD.
  - Use **node splitting** (refinement) and **edge domain filtering**.
- MDD provides a more refined relaxation.
  - Recognizes some variable interactions.
- Easily adapted to optimization.
  - Shortest path computation

Objectives

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Show that MDD-based propagation can follow a standard pattern.
  - And yet can be **specialized** to each type of constraint to exploit structure.
  - Implement a general **MDD-based constraint solver** based on this pattern.

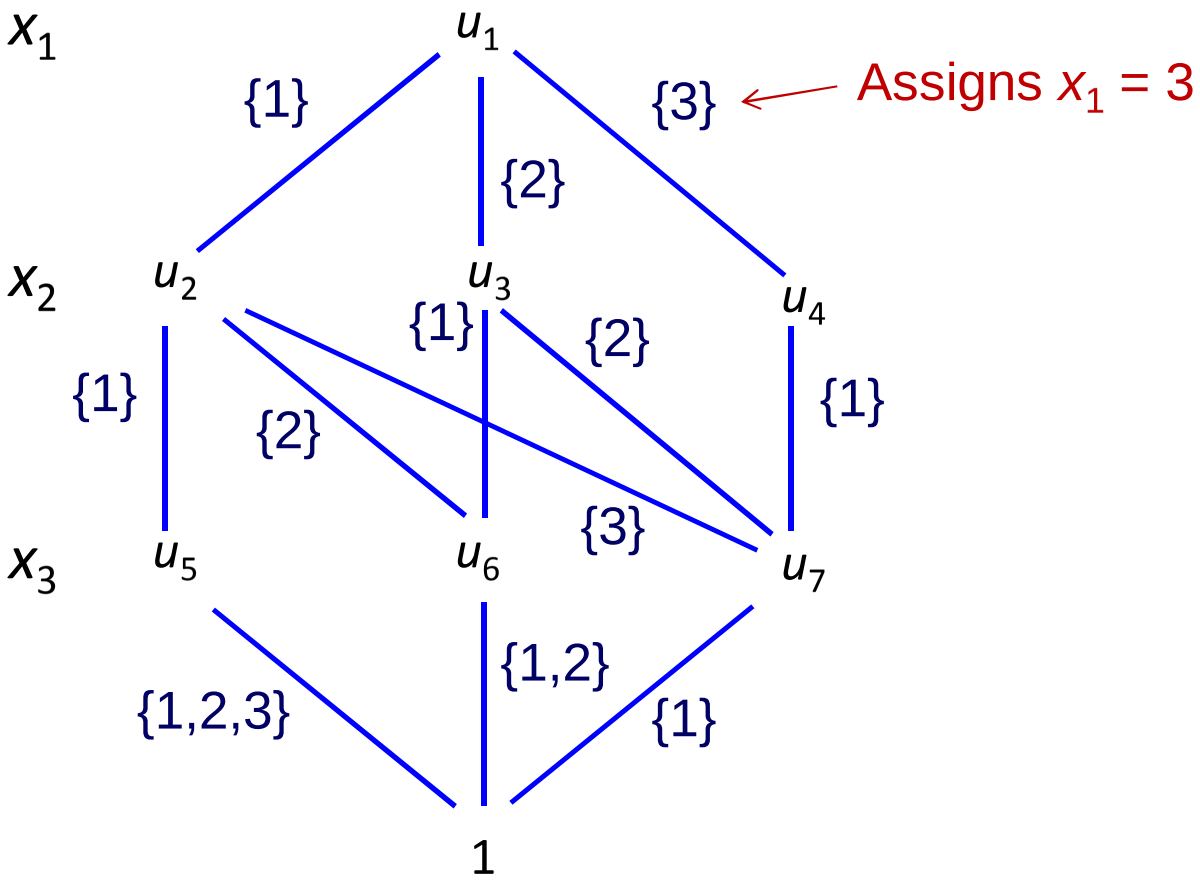
- Test the MDD-based solver on a class of constraints.
  - Multiple **among** constraints.

MDD basics

MDD representing

$x_1 + x_2 + x_3 \leq 5$

$x_j \in \{1,2,3\}$

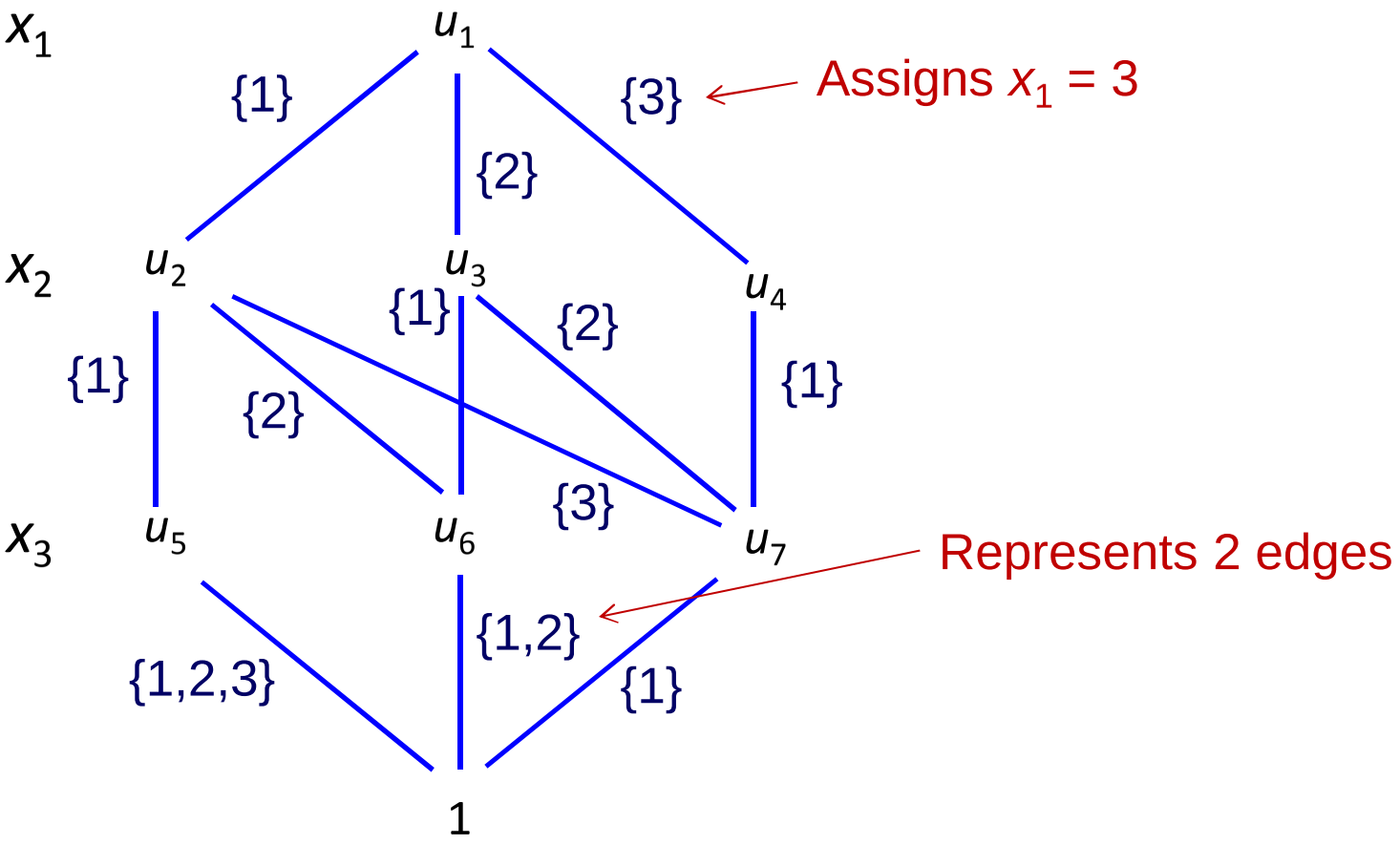


MDD basics

MDD representing

$x_1 + x_2 + x_3 \leq 5$

$x_j \in \{1,2,3\}$

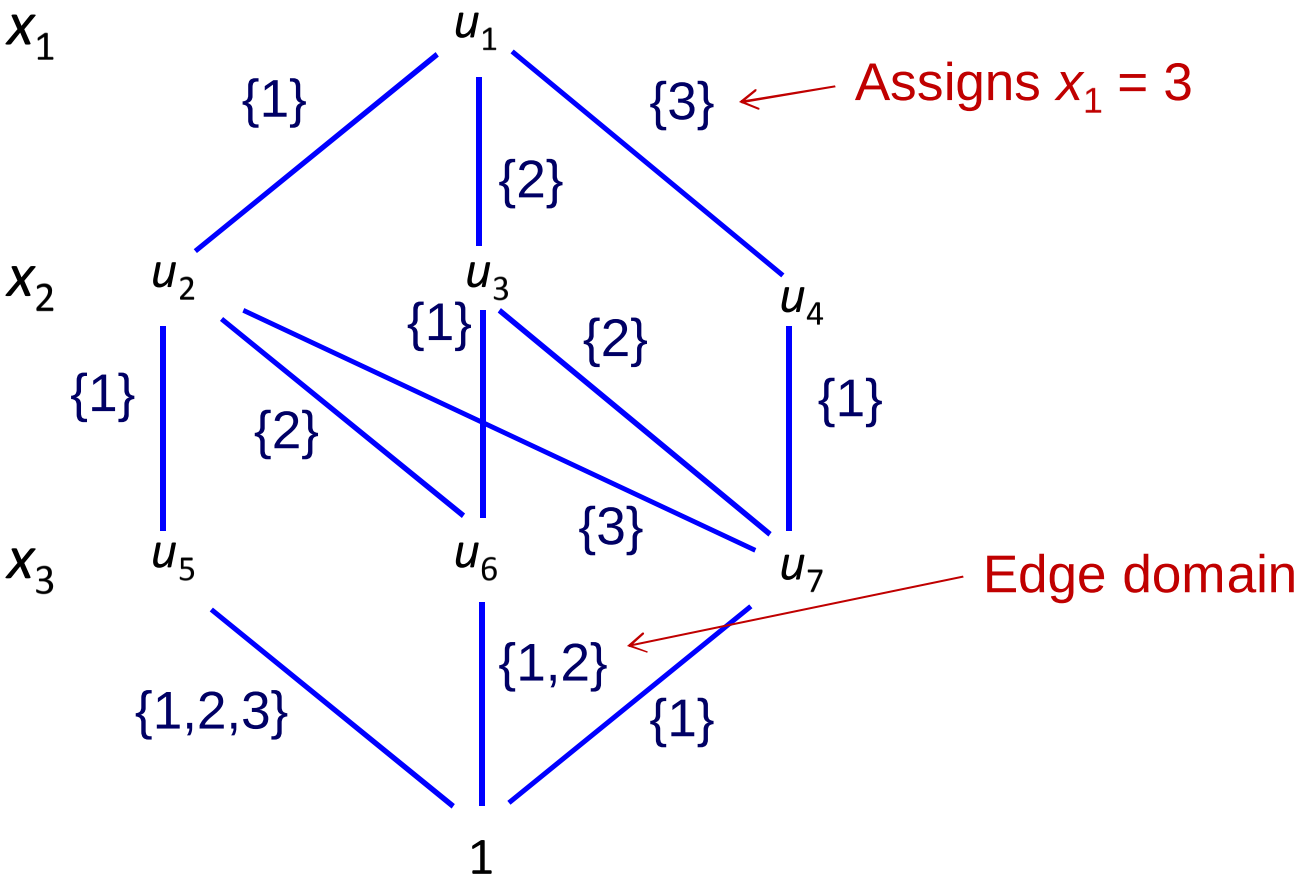


MDD basics

MDD representing

$x_1 + x_2 + x_3 \leq 5$

$x_j \in \{1,2,3\}$

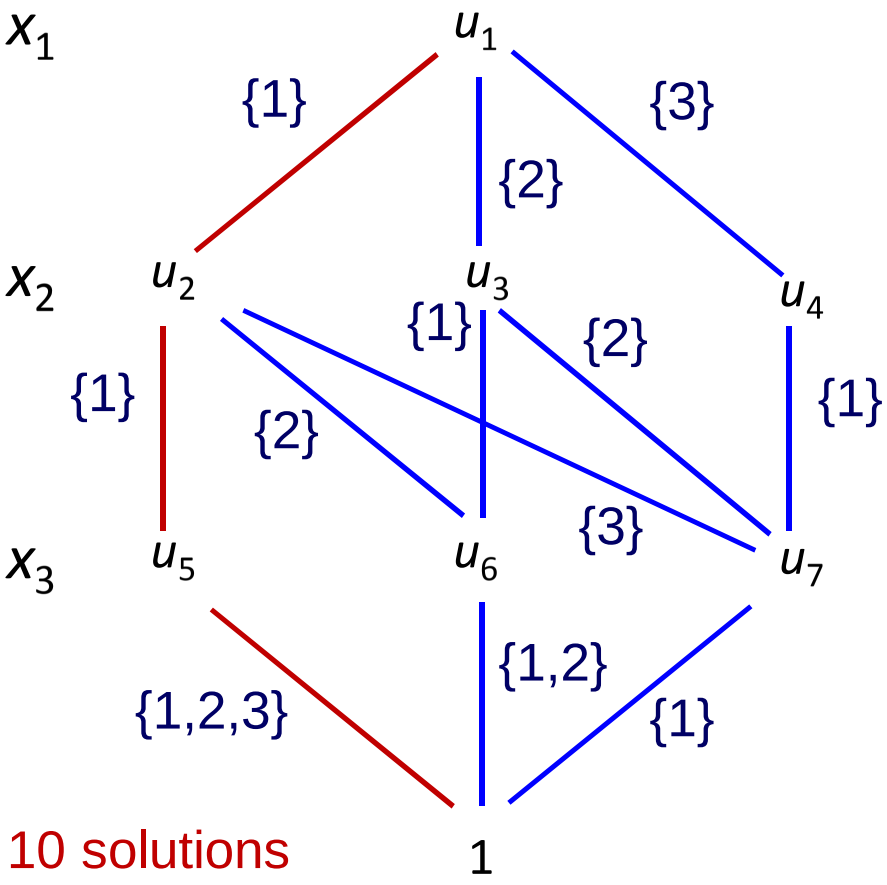


MDD basics

MDD representing

$x_1 + x_2 + x_3 \leq 5$

$x_j \in \{1,2,3\}$



10 solutions total

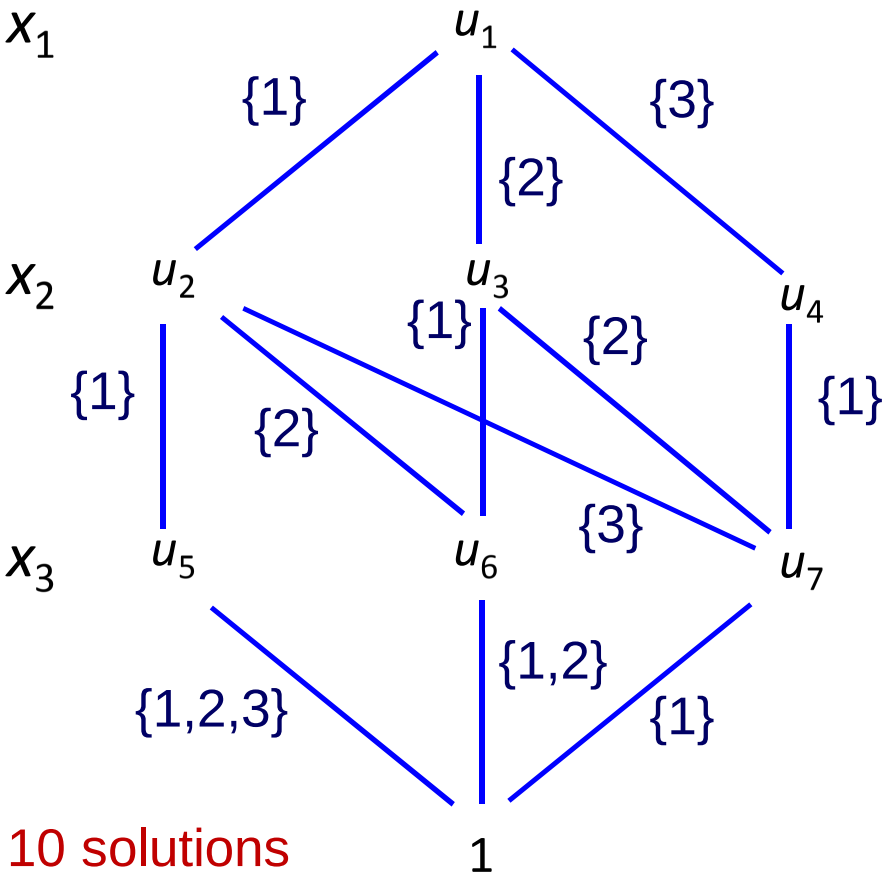
Each path from top to bottom represents a set of solutions

{1}×{1}×{1,2,3} =

{(1,1,1), (1,1,2), (1,1,3)}

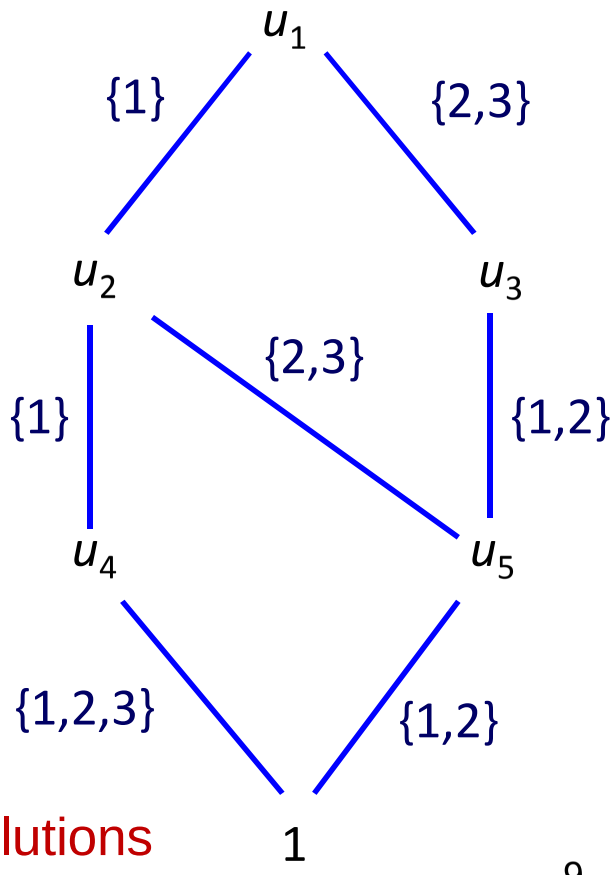
MDD relaxation

MDD representing
$x_1 + x_2 + x_3 \leq 5$
$x_j \in \{1,2,3\}$



10 solutions

A **relaxed** MDD (width 2) representing a superset of the solutions



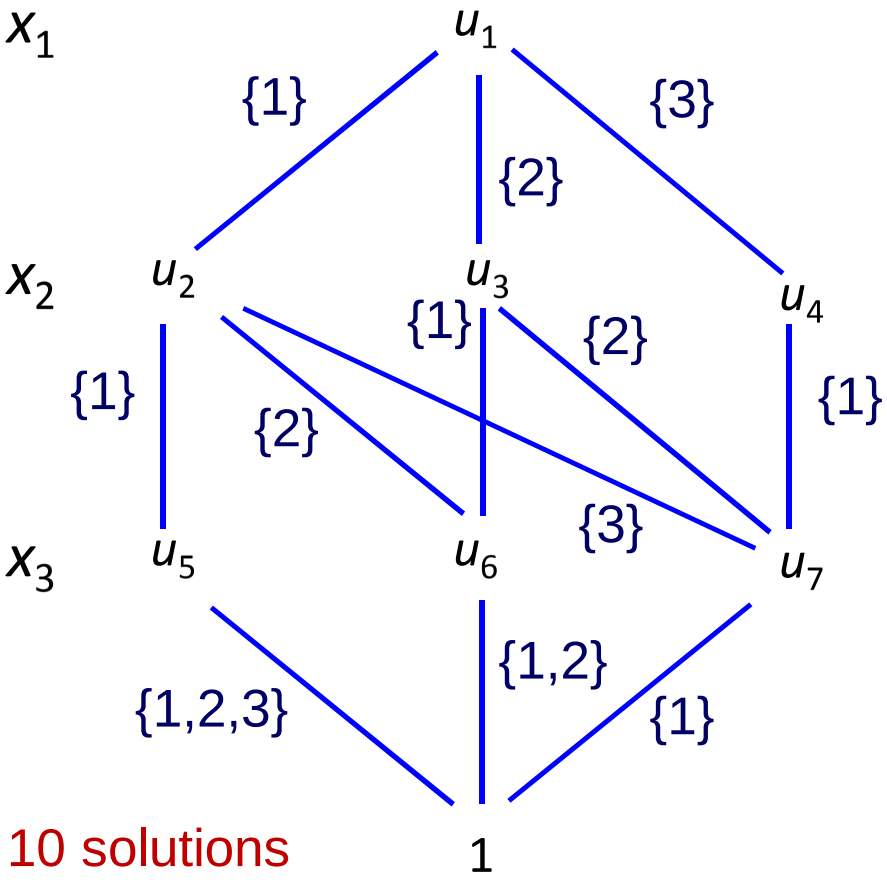
15 solutions

MDD relaxation

MDD representing

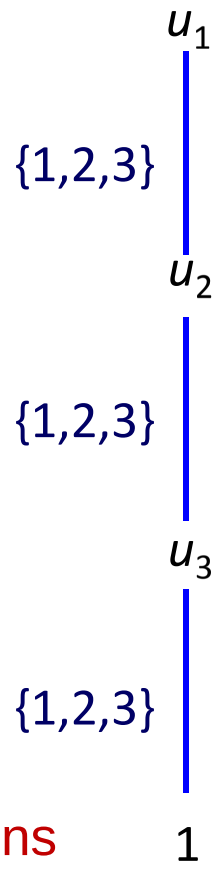
$x_1 + x_2 + x_3 \leq 5$

$x_j \in \{1,2,3\}$



10 solutions

Conventional **domain store**
(width 1 relaxation)



27 solutions

Among constraint

- We will use among constraint as a running example.
- At least *L* and at most *U* variables in set *X* take a value in set *S:*

among(*X*, *S*, *L*, *U*)

- Applications in sequencing and scheduling, etc.

Among constraint

- Example: Employee scheduling

x_t = shift assigned on day t.
S = {M,A,N} = {shifts}

Between 2 and 4 afternoon/night shifts in a week:

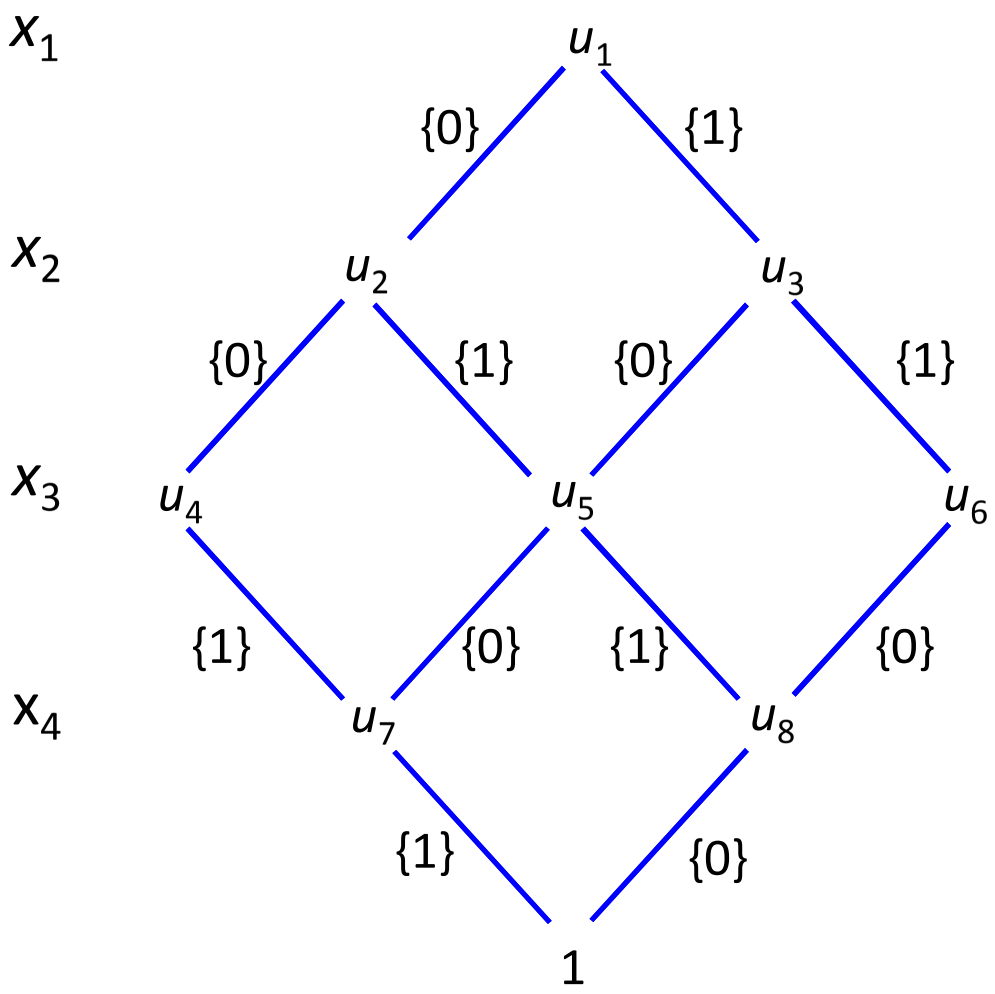
Among($\{x_1, \ldots, x_7\}$, {A,N}, 2, 4)

t	1	2	3	4	5	6	7
x_t	M	**A**	**N**	M	M	**A**	M

WOLOG, suppose $x_t \in \{0,1\}$, S = {1}

t	1	2	3	4	5	6	7
x_t	0	**1**	**1**	0	0	**1**	0

Among constraint



Exact MDD for among($\{x_1,x_2,x_3,x_4\}$, {1}, 2, 2)

MDD-based constraint programming

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (domain store)
  - Update MDD by propagating constraints sequentially
- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges
  - Node refinement: Split nodes to separate edge information
- Search
  - As in classical CP, but may now be guided by MDD
- Easily extended to optimization
  - Shortest path in MDD

Example

- We will propagate

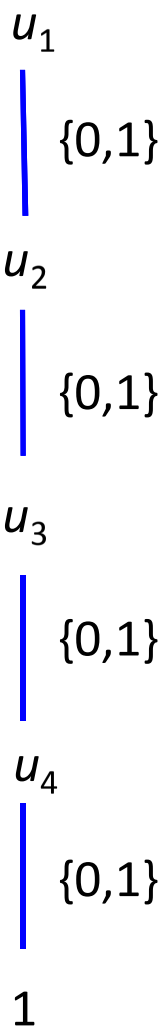
$$\text{among}(\{x_1,x_2,x_3,x_4\}, \{1\}, 2, 2)$$

through a BDD of maximum width 3

  - All path lengths must be = 2.

Example

Initially, the MDD has width 1
(traditional domain store)



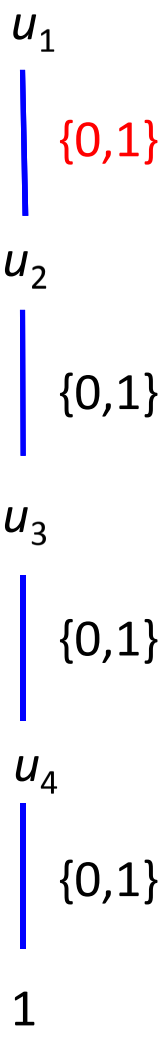
among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example

Try to filter edge domain

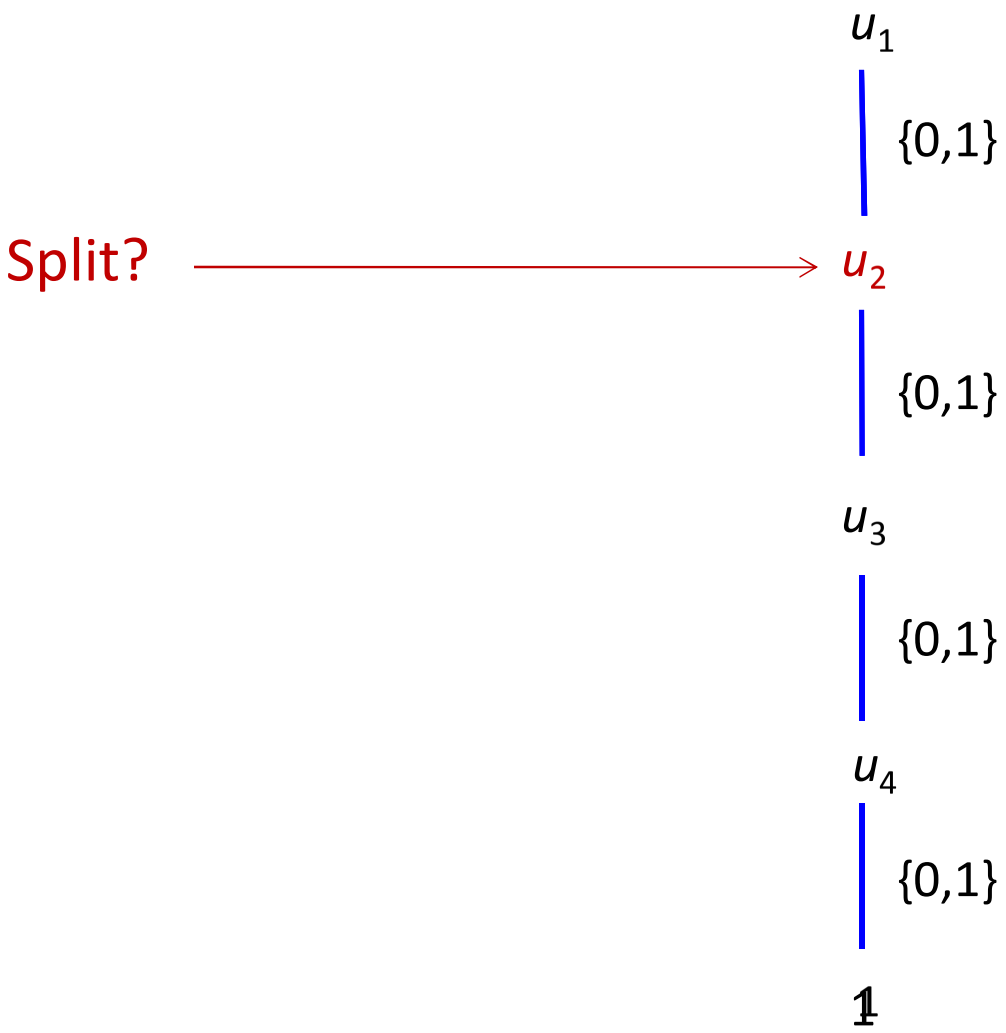
No filtering possible.

Initially, the MDD has width 1
(traditional domain store)



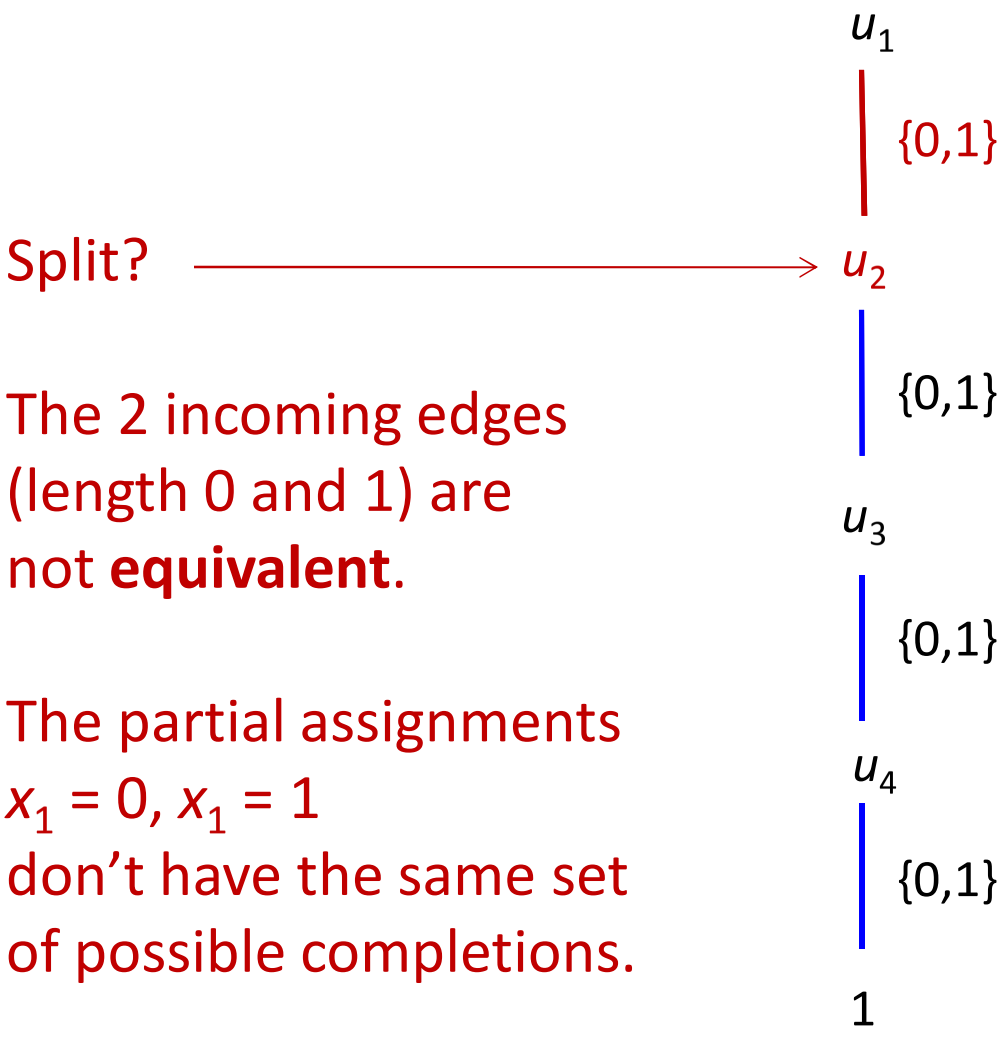
among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example



among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example

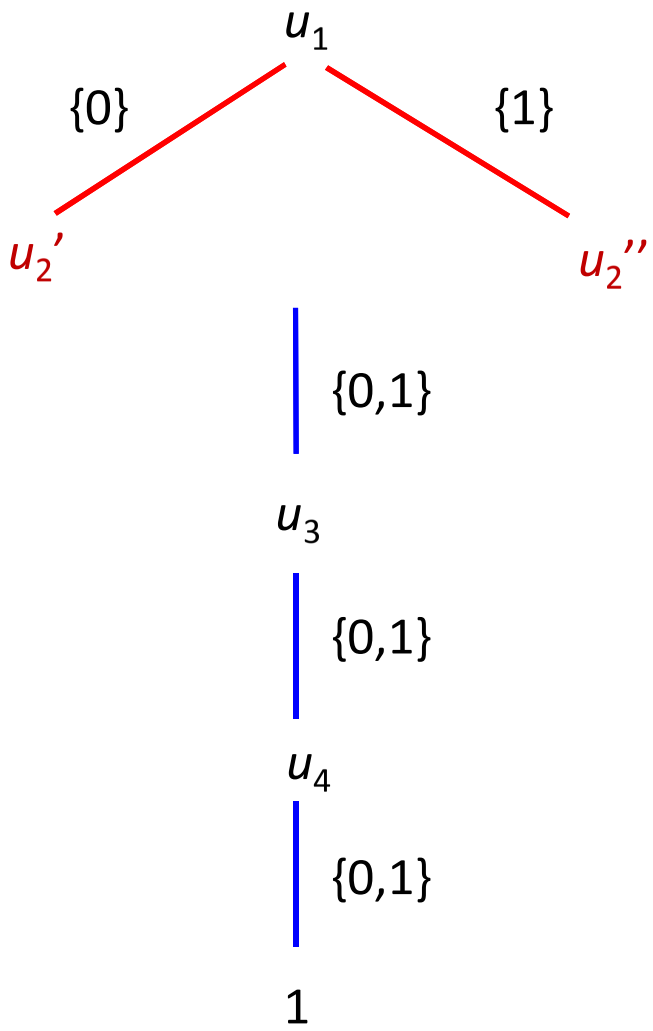


The 2 incoming edges (length 0 and 1) are not **equivalent**.

The partial assignments $x_1 = 0$, $x_1 = 1$ don't have the same set of possible completions.

among($\{x_1,x_2,x_3,x_4\}$, {1}, 2 ,2)

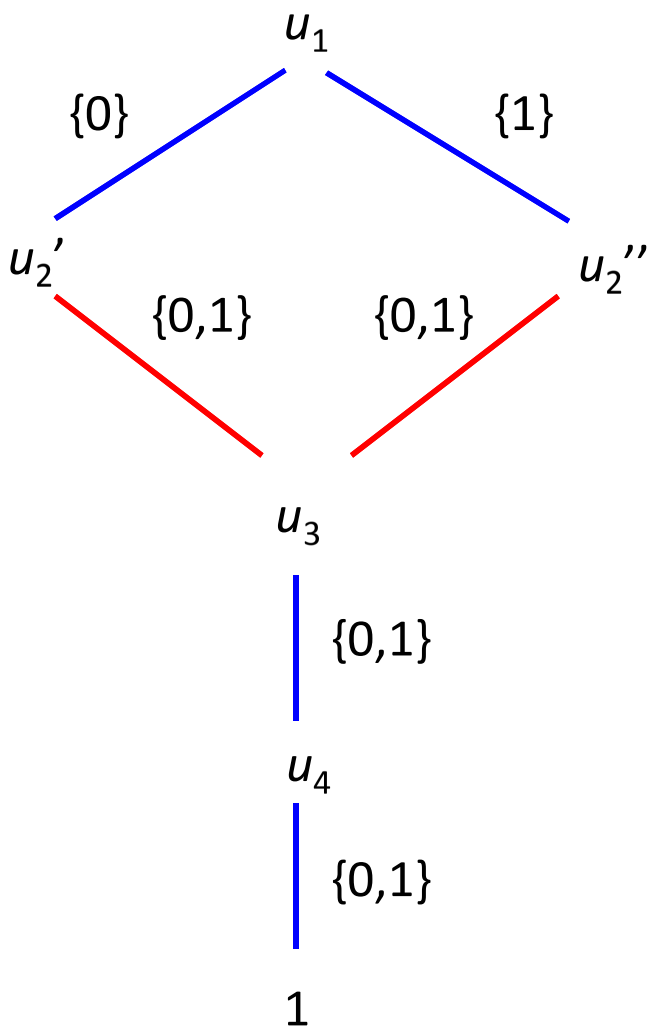
Example



So we split.
This doesn't exceed max width of 3.

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example

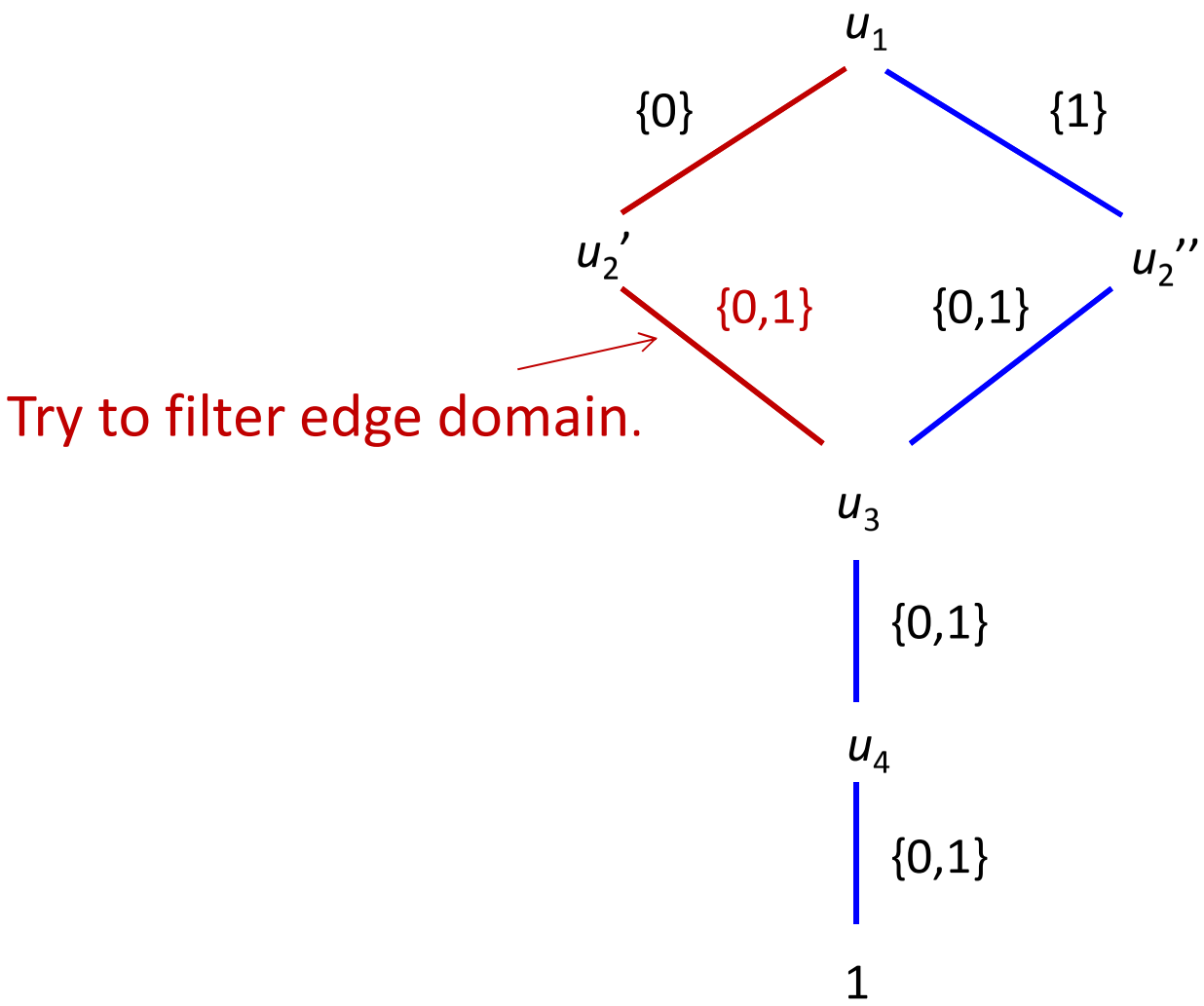


So we split.
This doesn't exceed max width of 3.

Duplicate outgoing Edges.

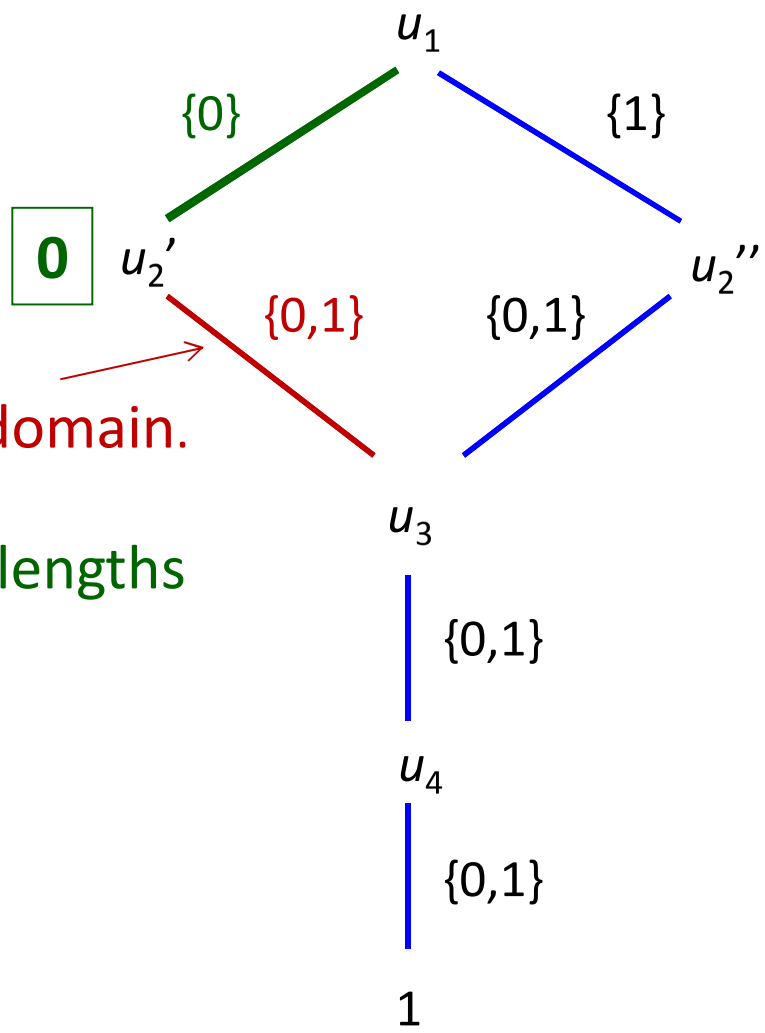
among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example



among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example

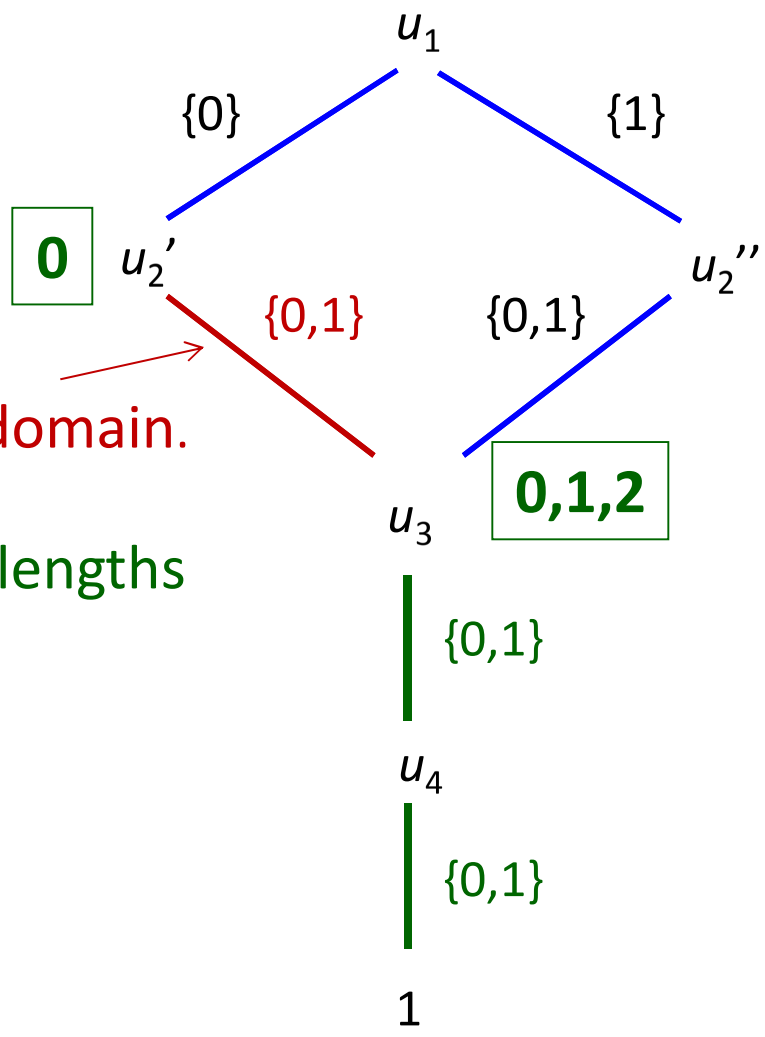


Try to filter edge domain.

Compute all path lengths from top…

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example

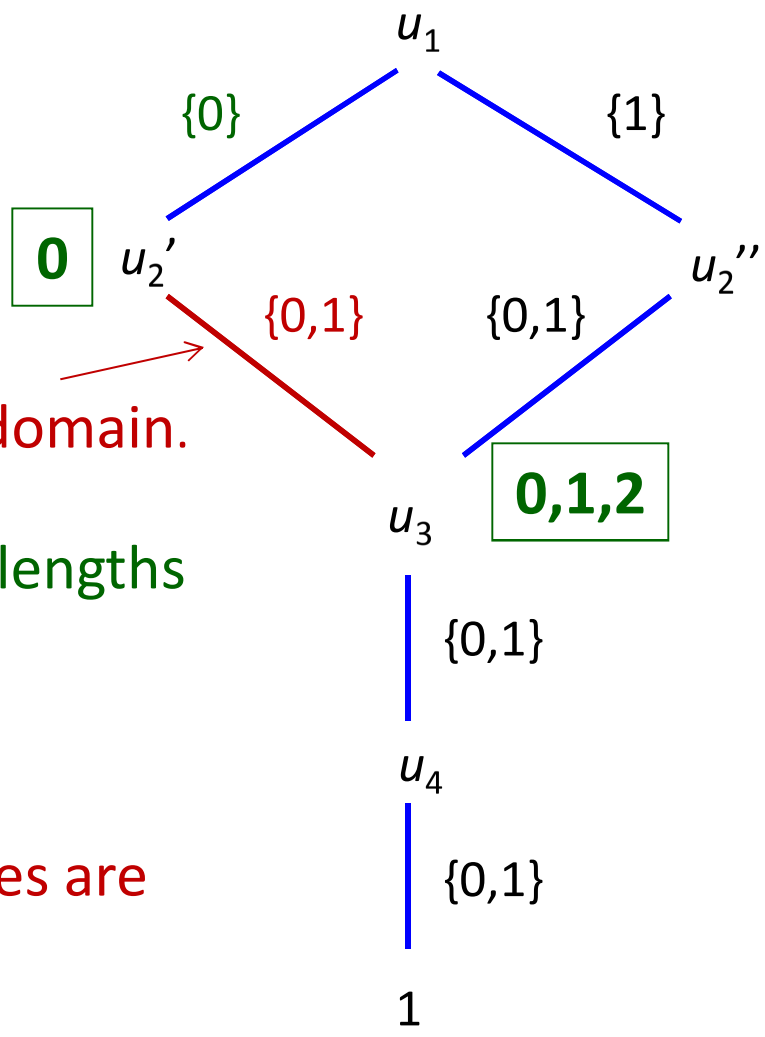


Try to filter edge domain.

Compute all path lengths
from top…
and from bottom.

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example



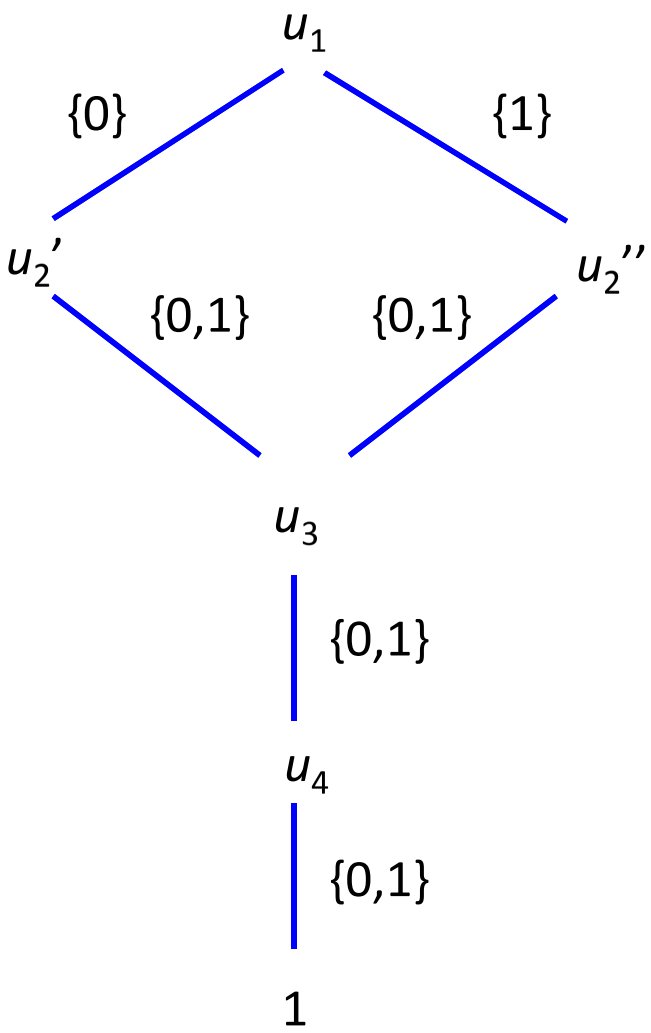
Try to filter edge domain.

Compute all path lengths from top…
and from bottom.

Both domain values are consistent.

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

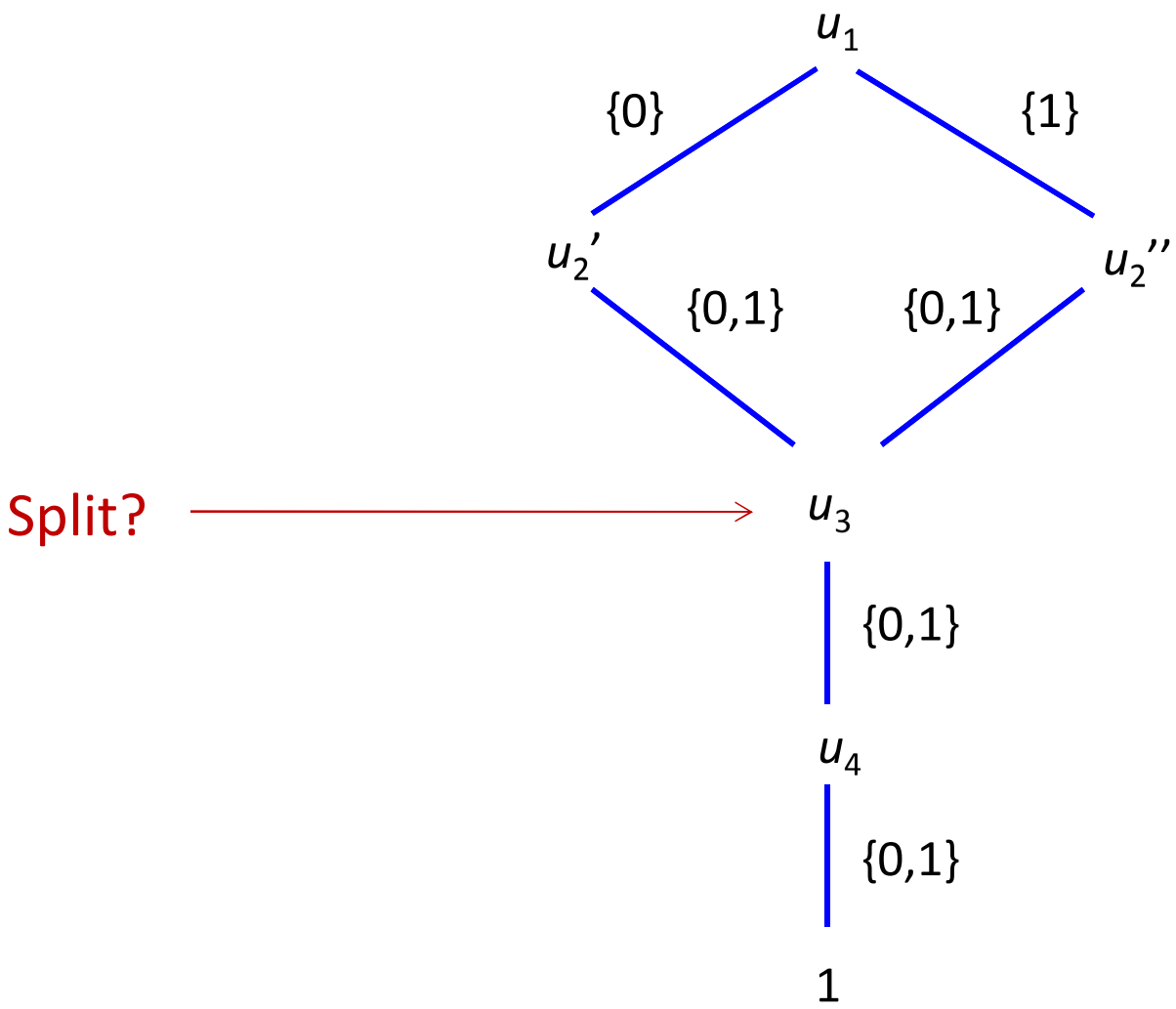
Example



No filtering possible for x_2.

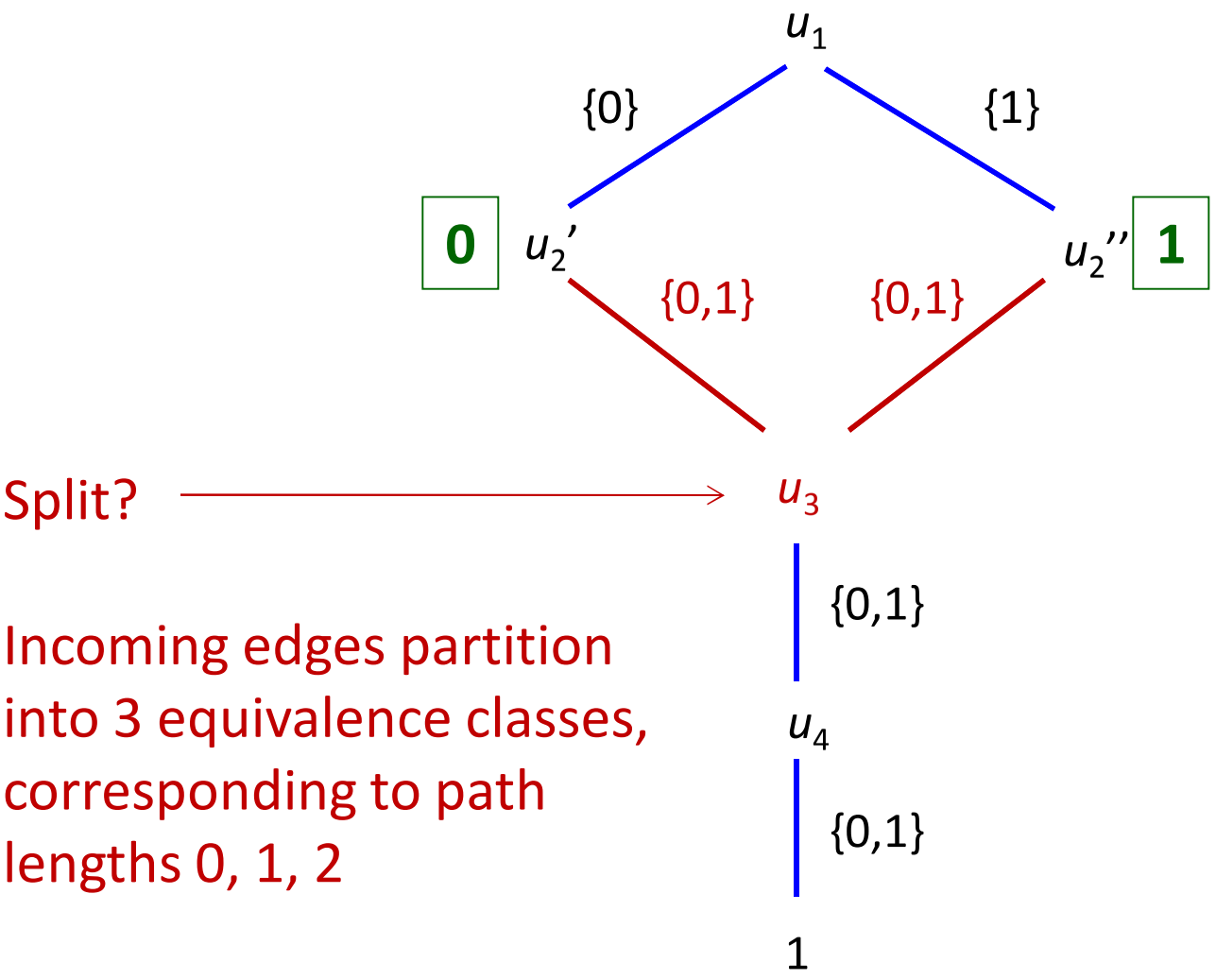
among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example



among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

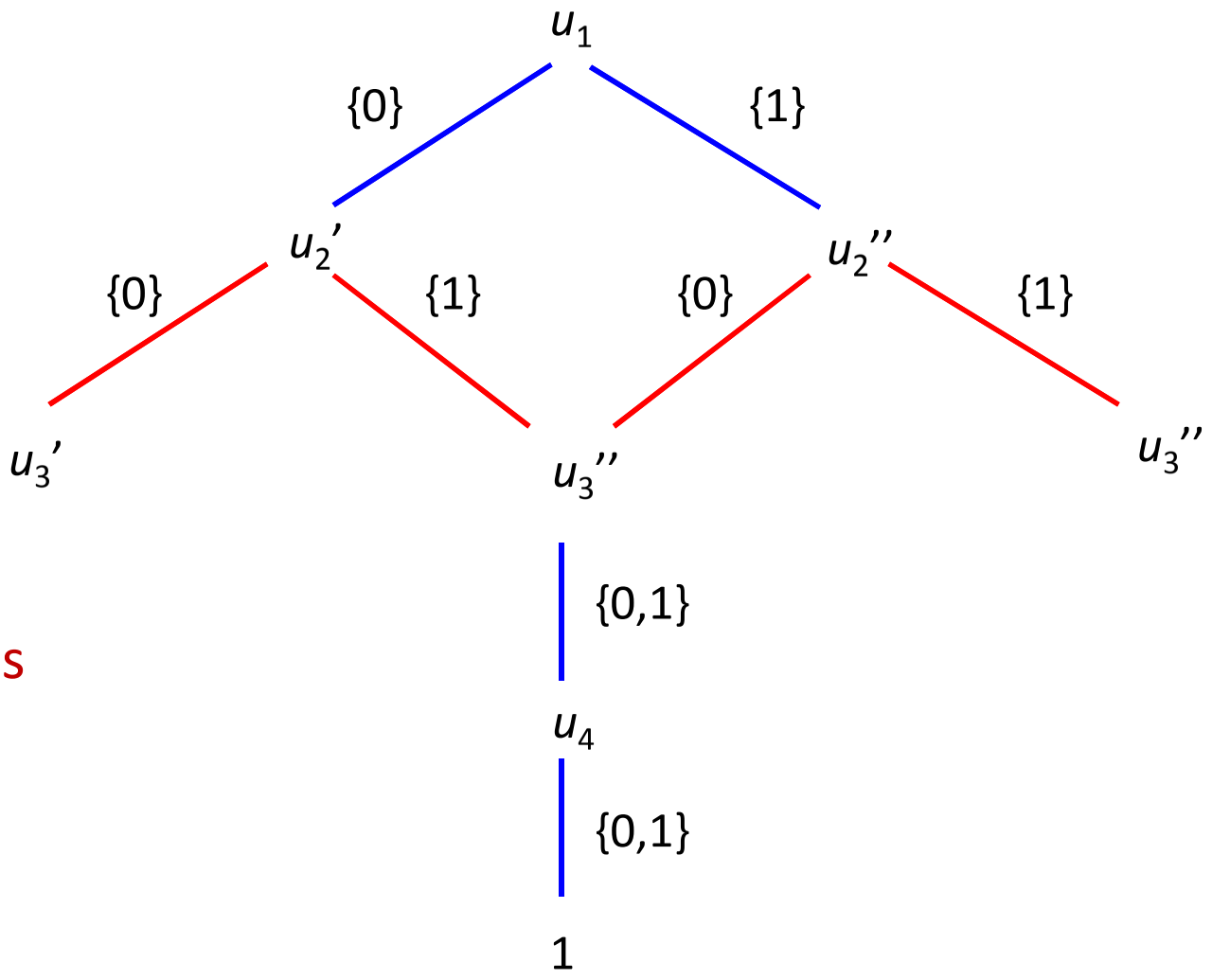
Example



Incoming edges partition into 3 equivalence classes, corresponding to path lengths 0, 1, 2

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

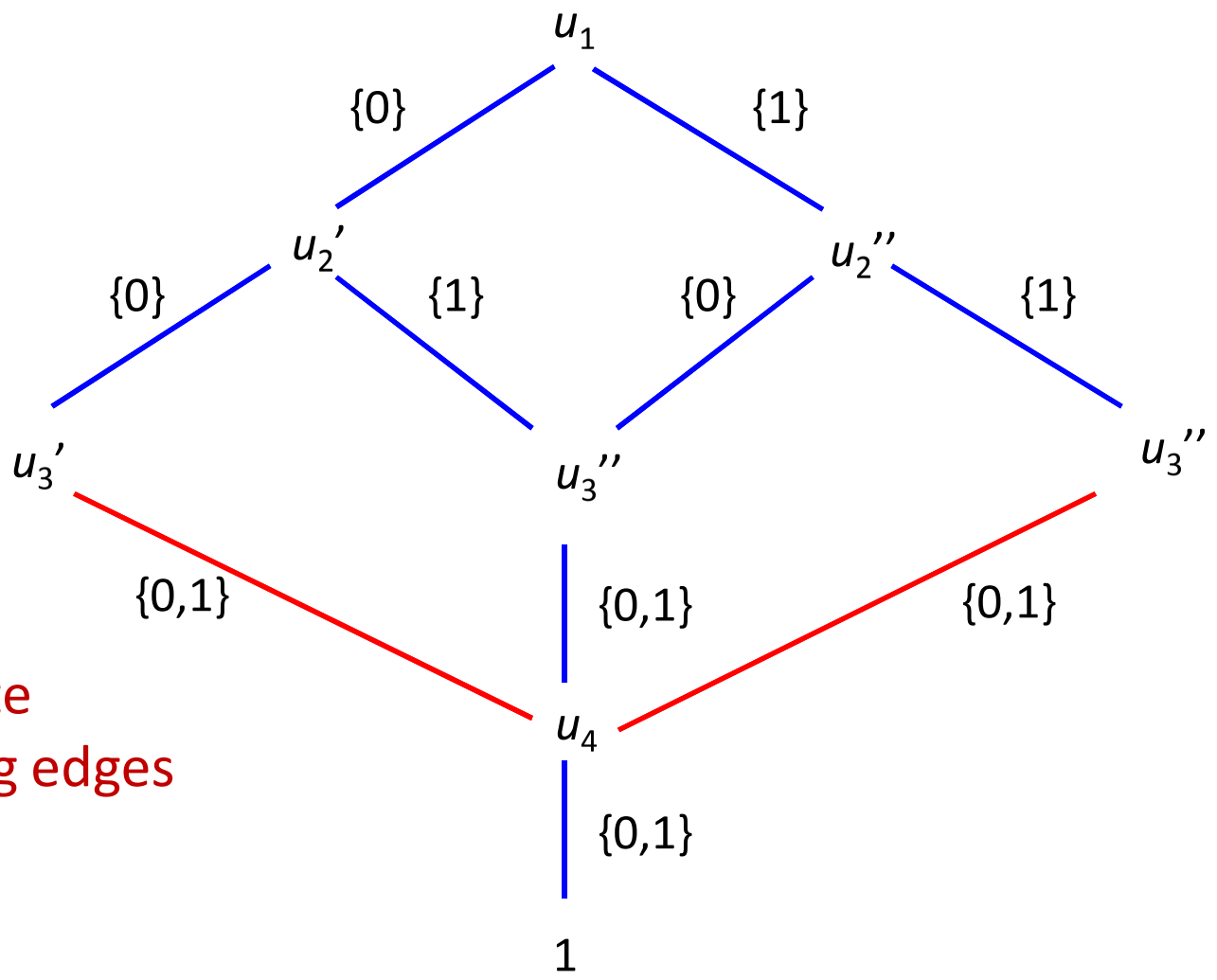
Example



So we split into 3 nodes

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

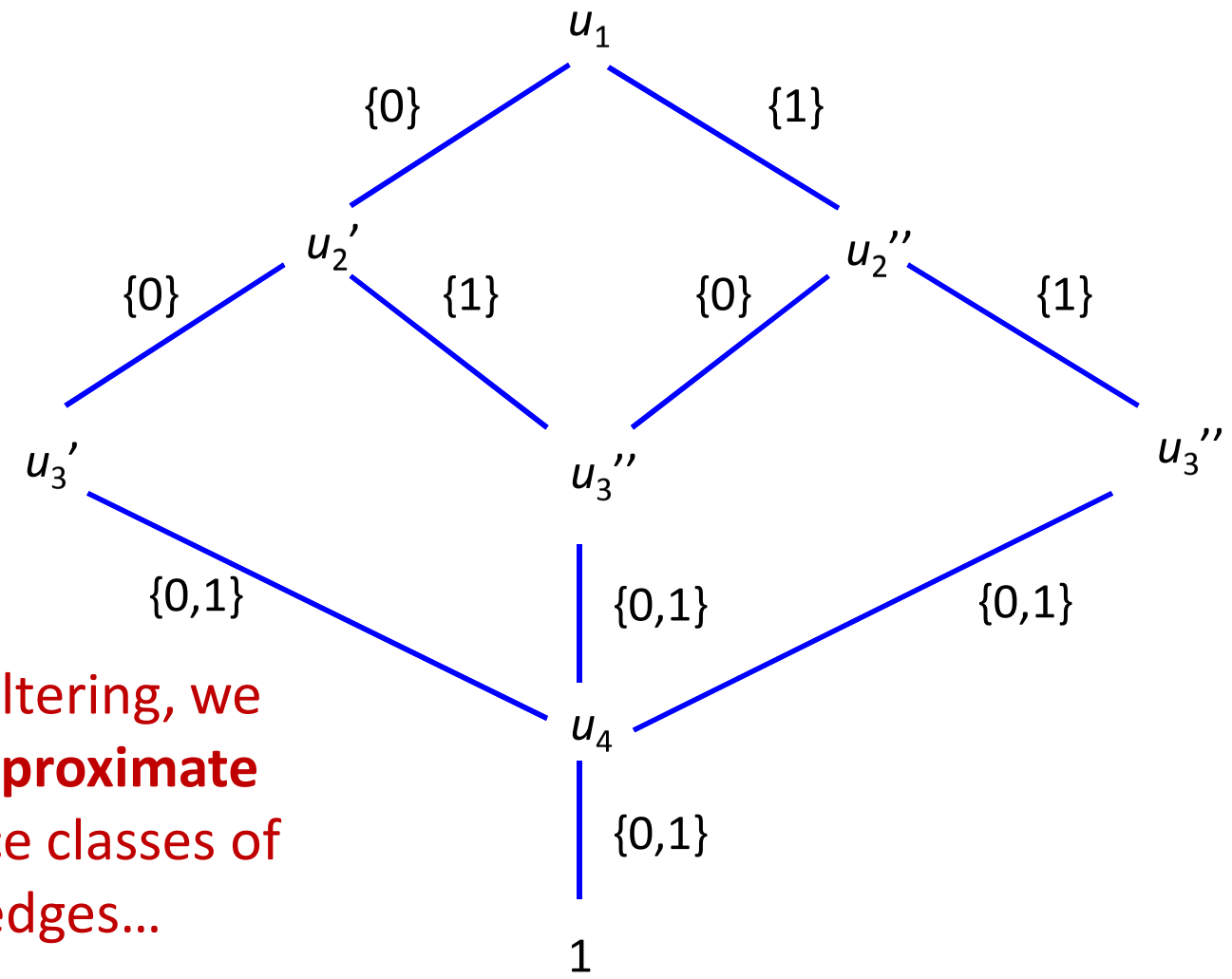
Example



Duplicate outgoing edges

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

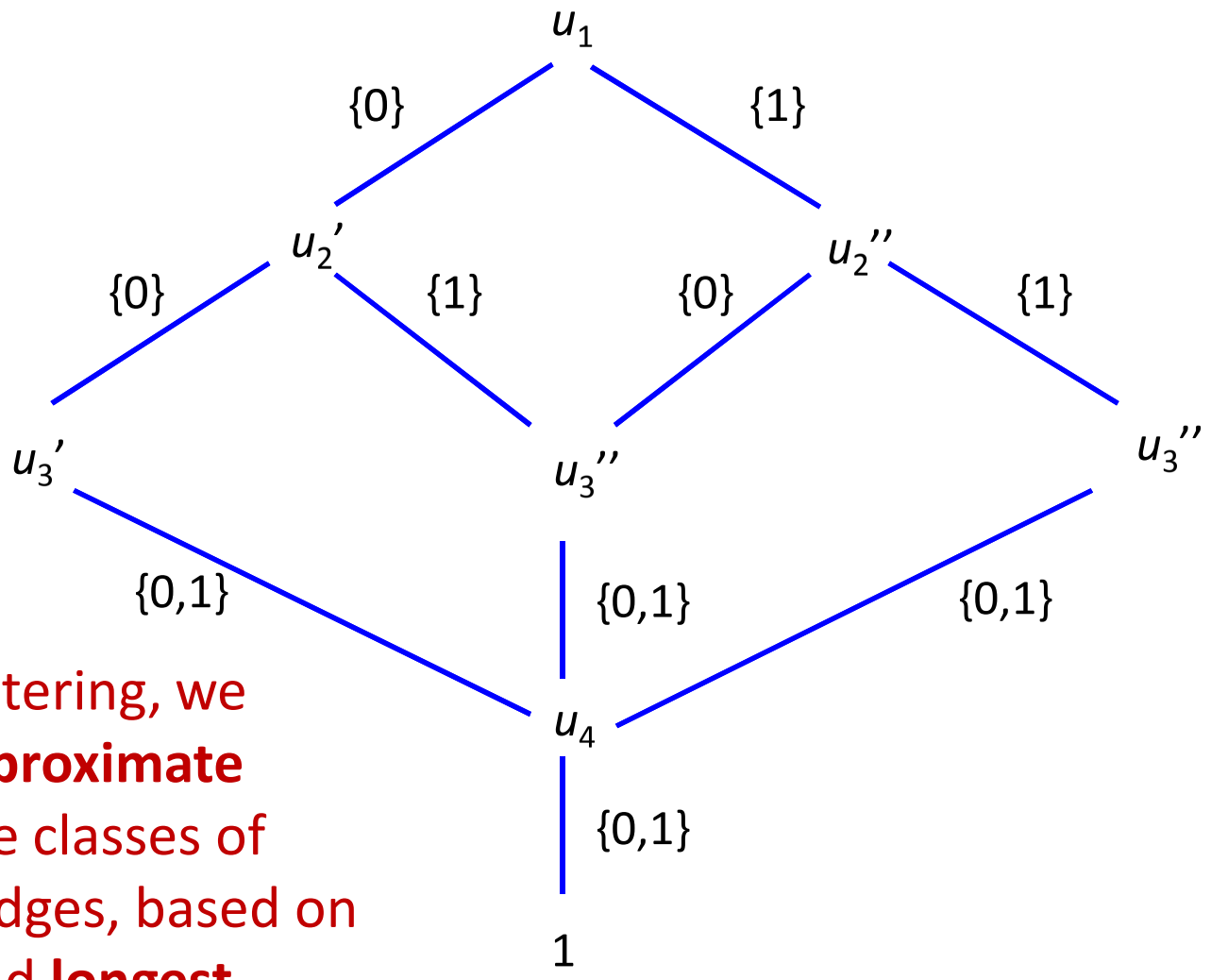
Example



To speed filtering, we identify **approximate** equivalence classes of incoming edges…

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

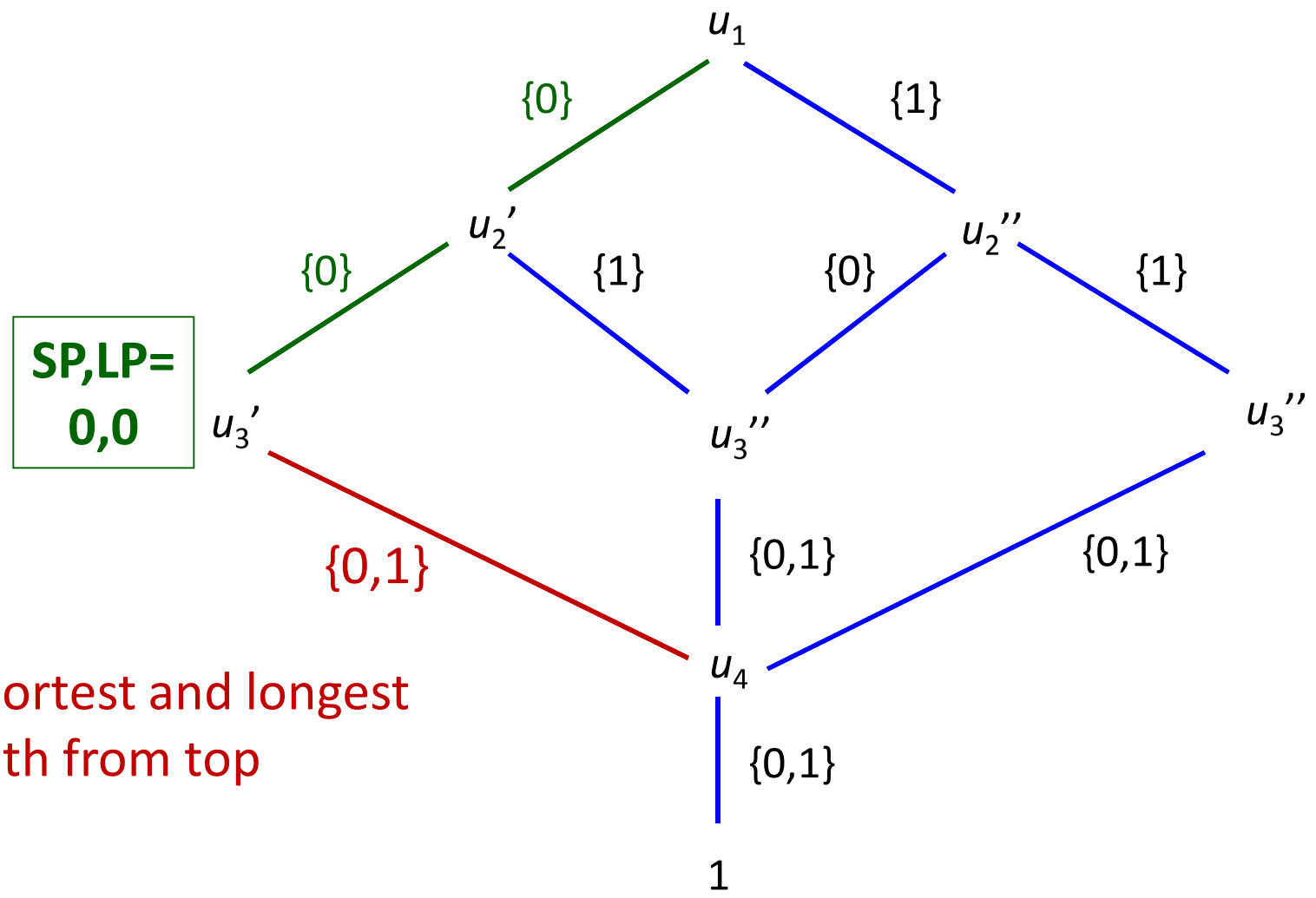
Example



To speed filtering, we identify **approximate** equivalence classes of incoming edges, based on **shortest** and **longest paths**.

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

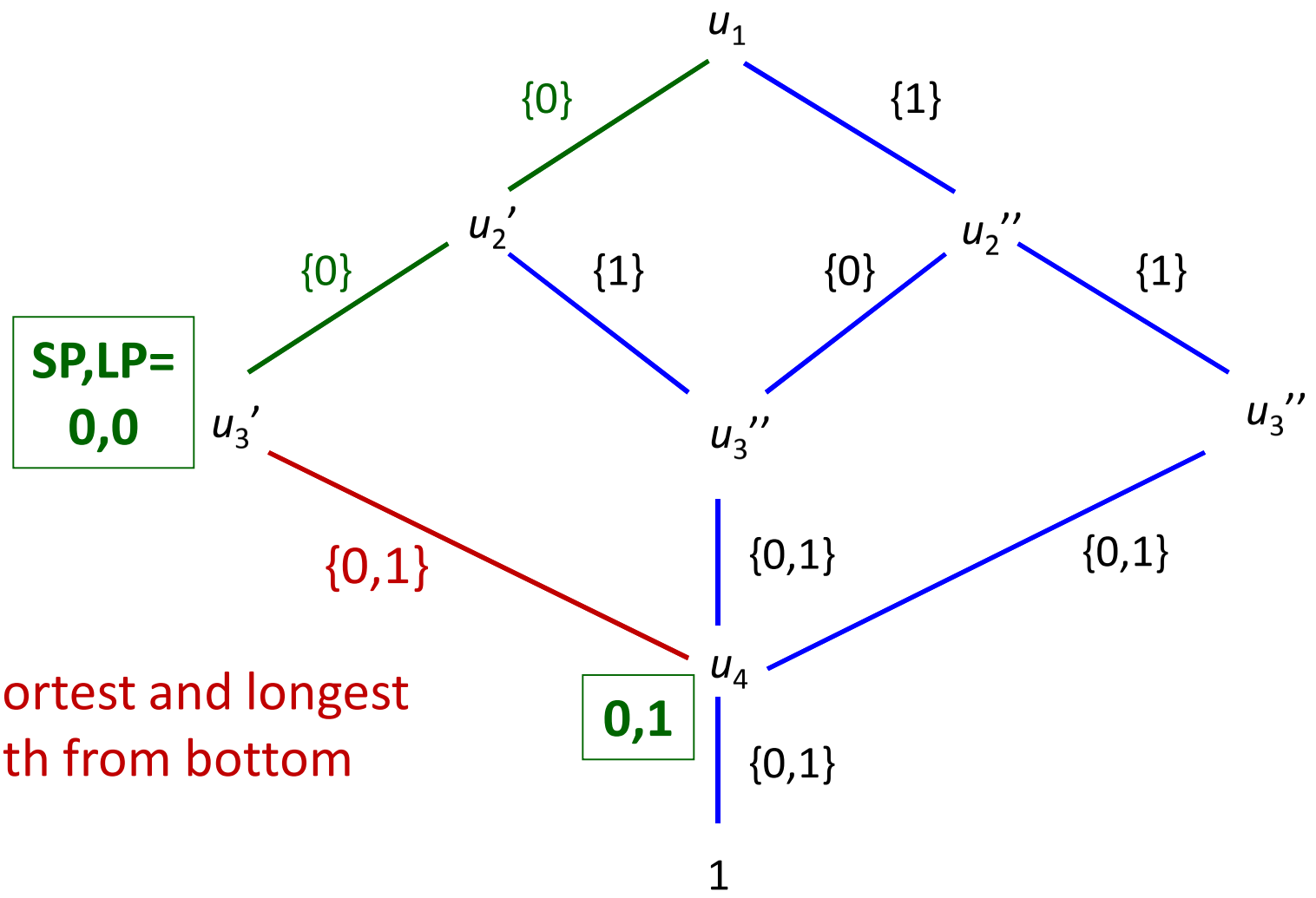
Example



Shortest and longest path from top

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

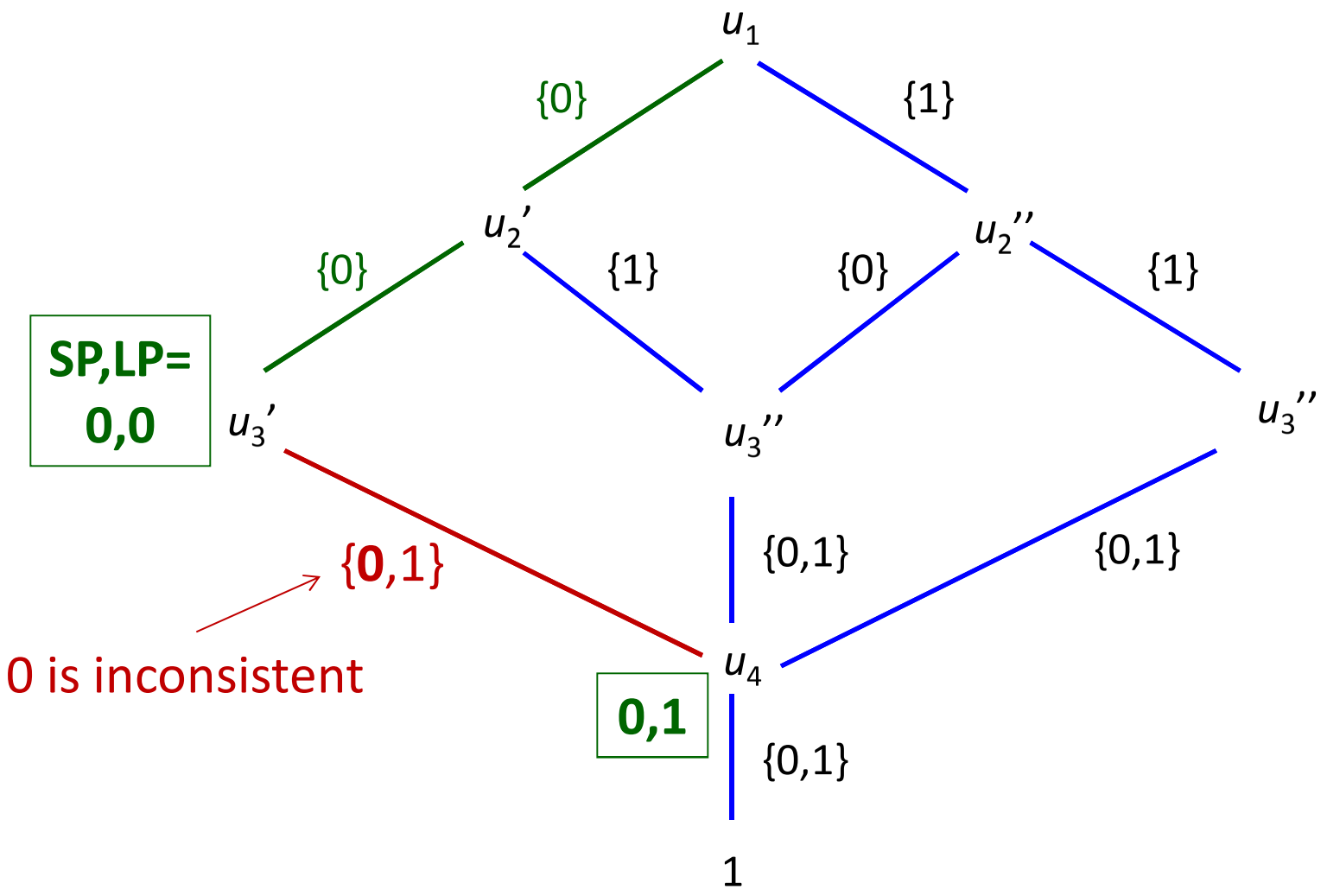
Example



Shortest and longest path from bottom

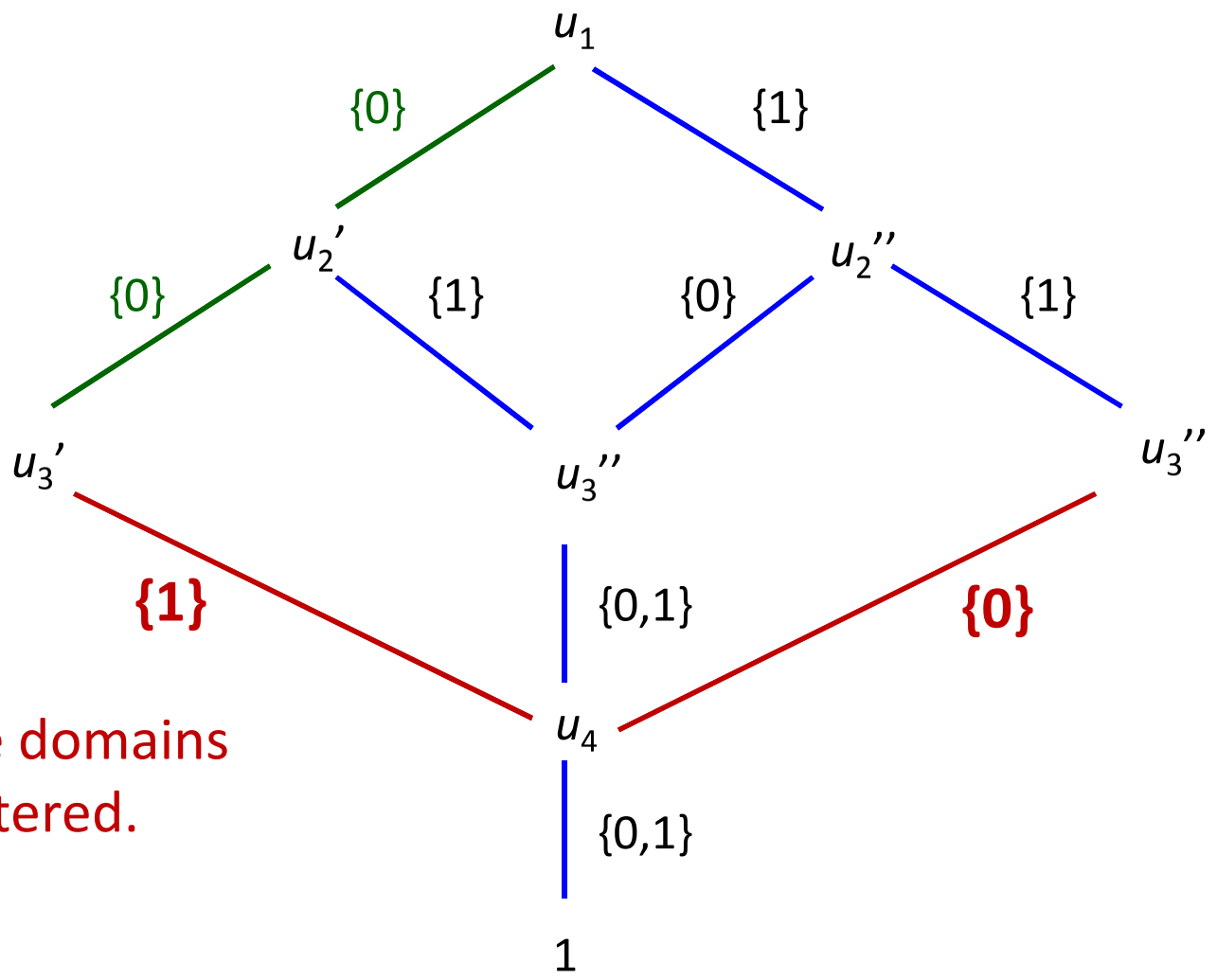
among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example



among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

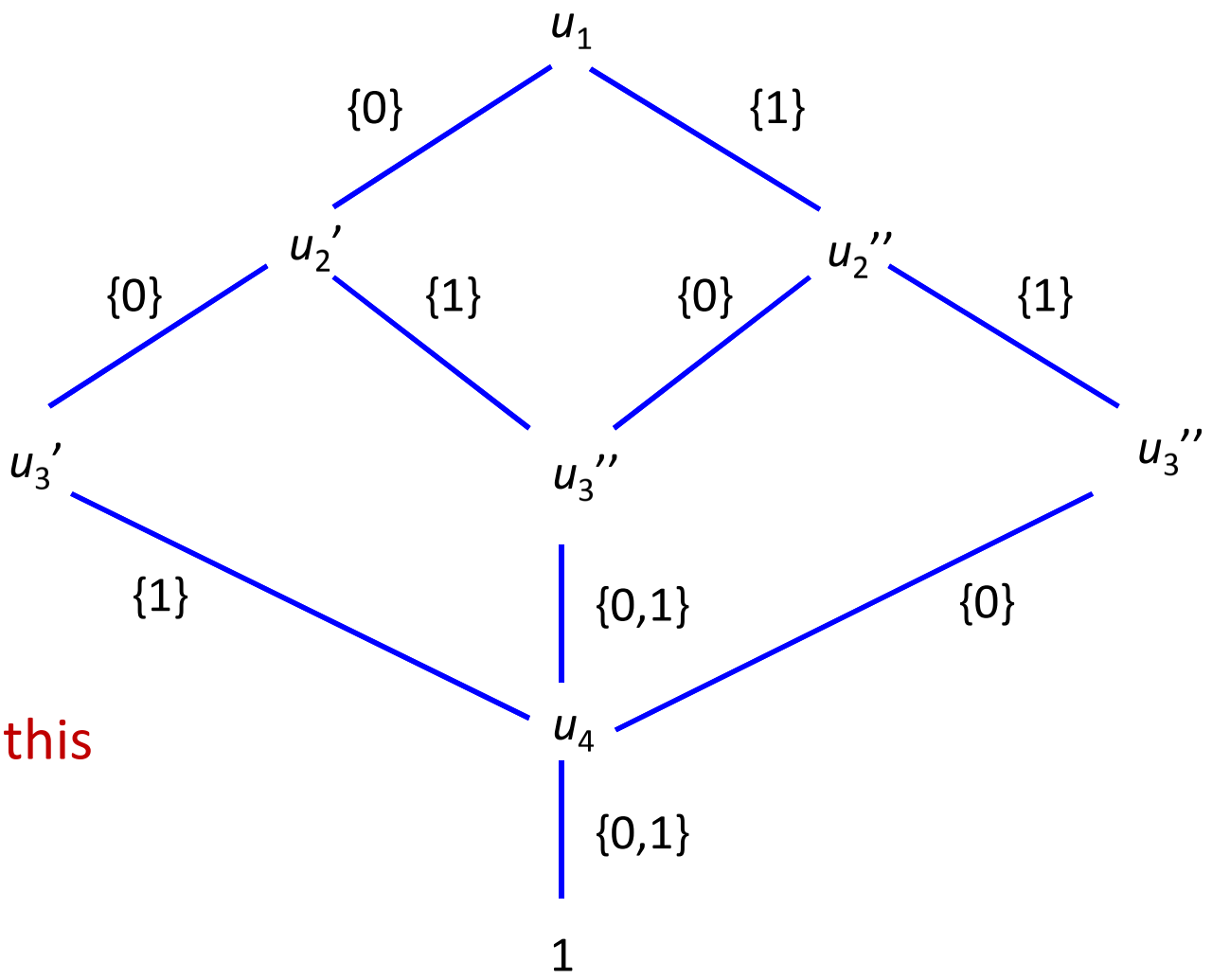
Example



Two edge domains can be filtered.

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

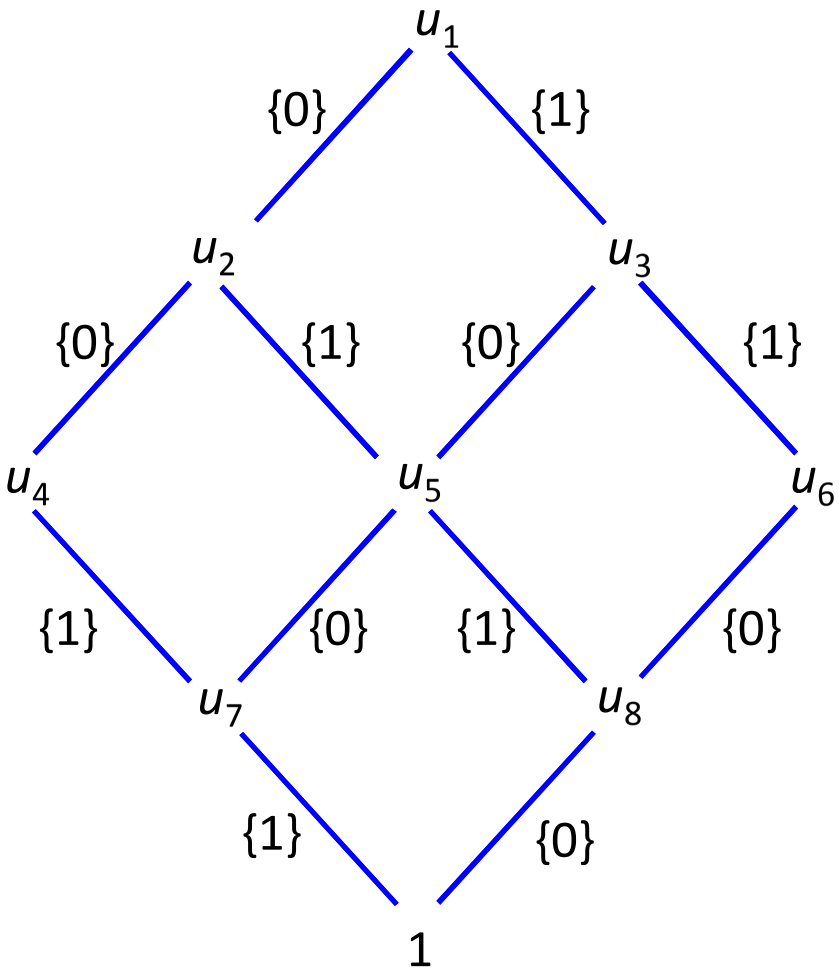
Example



Continue this process…

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

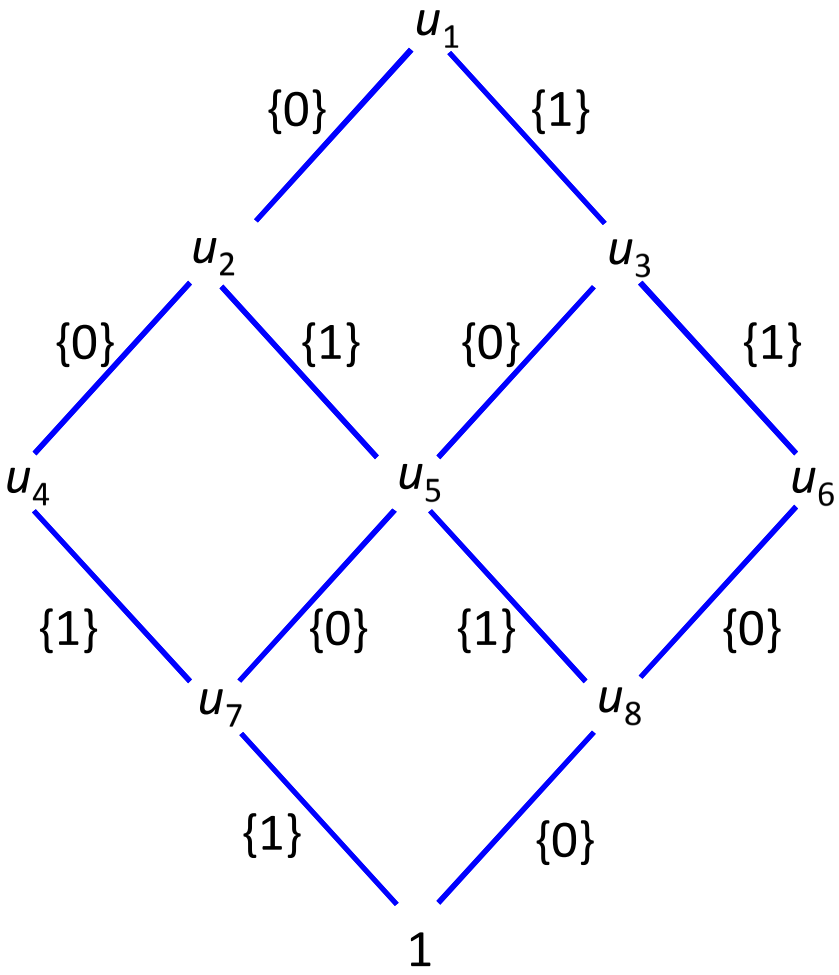
Example



…and obtain a relaxed MDD of width 3

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

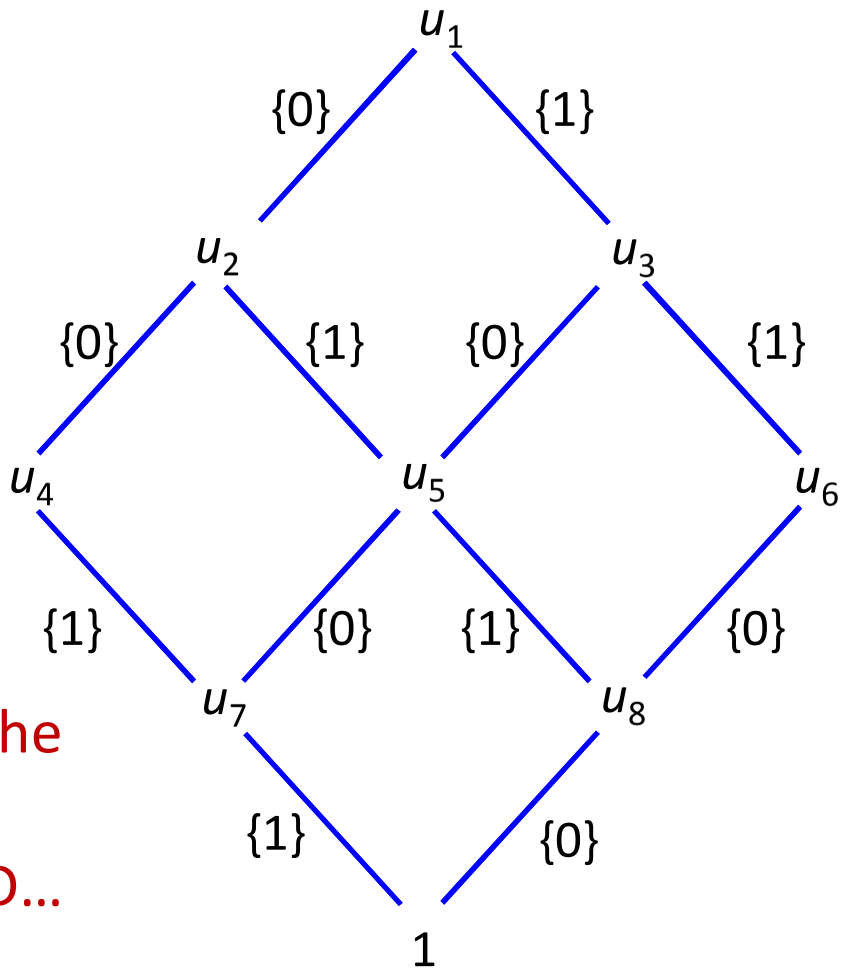
Example



In this case, the MDD is exact.

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

Example



Now, propagate the next constraint through **this** MDD…

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

General Scheme for Node Splitting

- Identify (approximate) equivalence classes of incoming edges.
- Create a copy of the node for each equivalence class.
  - But without exceeding the max width.
- Duplicate outgoing arcs.

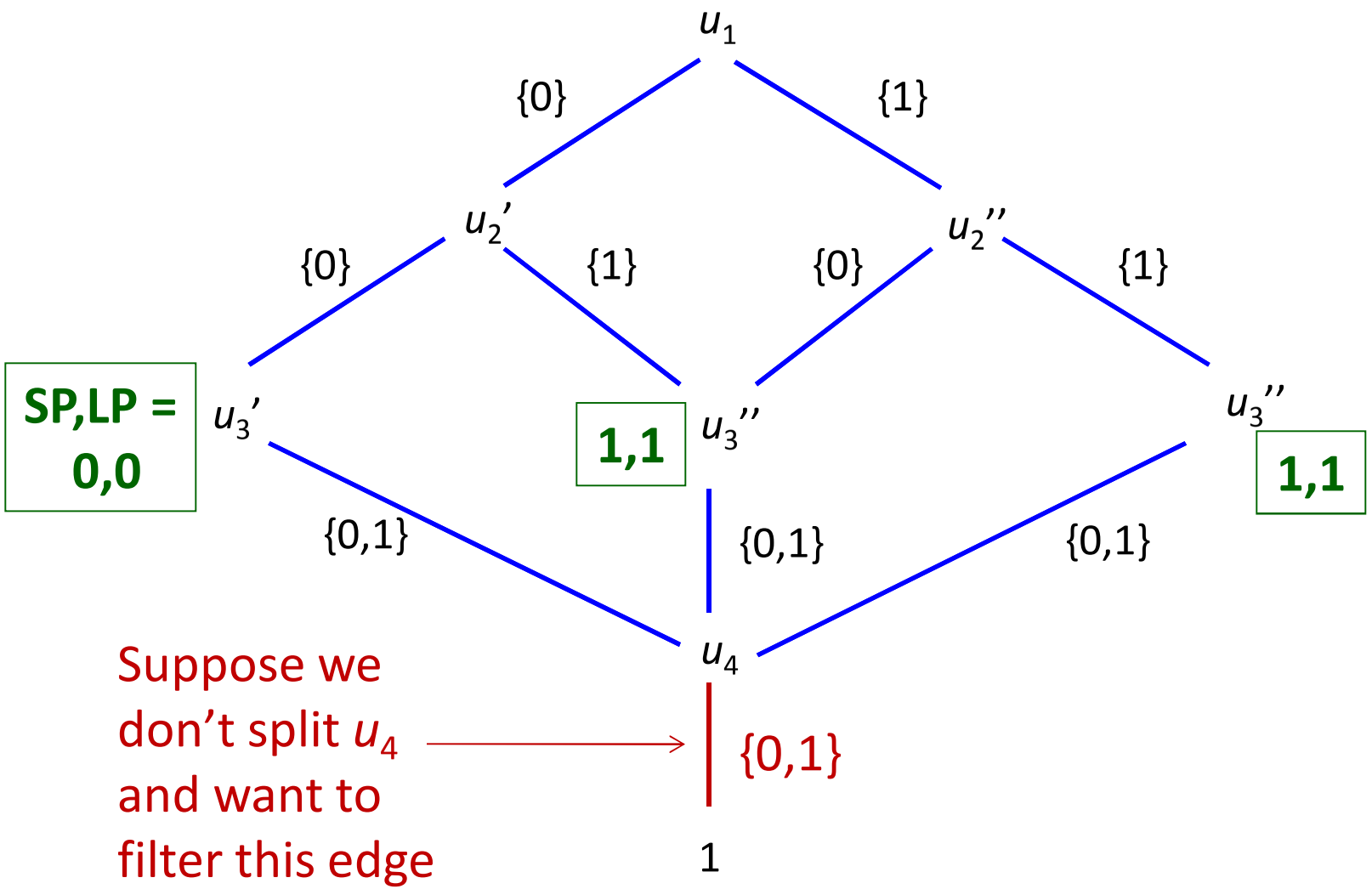
- Method for identifying equivalence classes is **constraint-specific**.
  - This exploits **special structure**.
  - Can also be **dynamic**, e.g. depending on current width.

General Scheme for Filtering

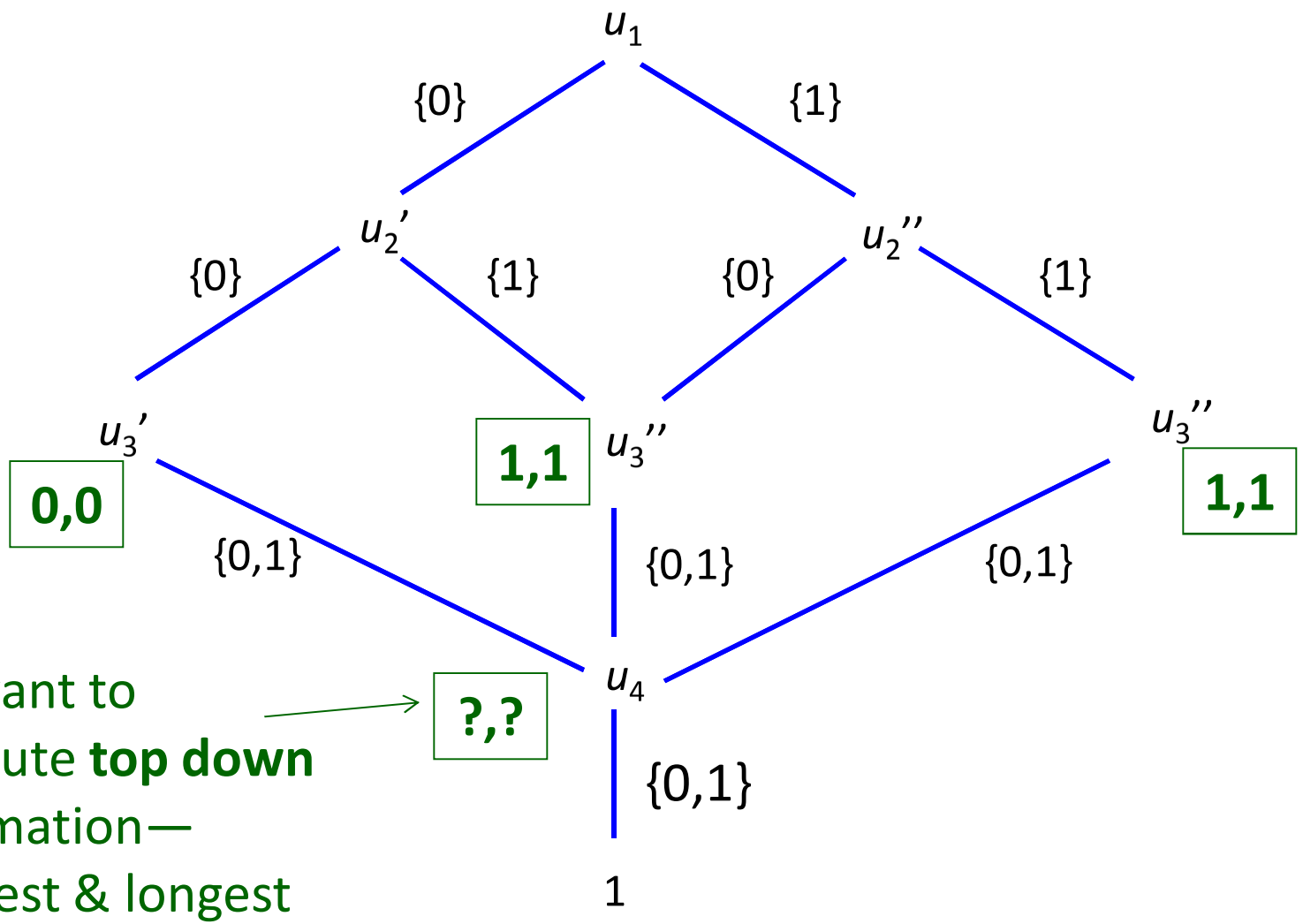
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Filtering on an edge is based on **information** at either end of the edge.
- Information is accumulated during **top-down** and **bottom-up** passes.
  - Based on **extend** (⊗) and **merge** (⊕) operations.
- For example…

General Scheme for Filtering



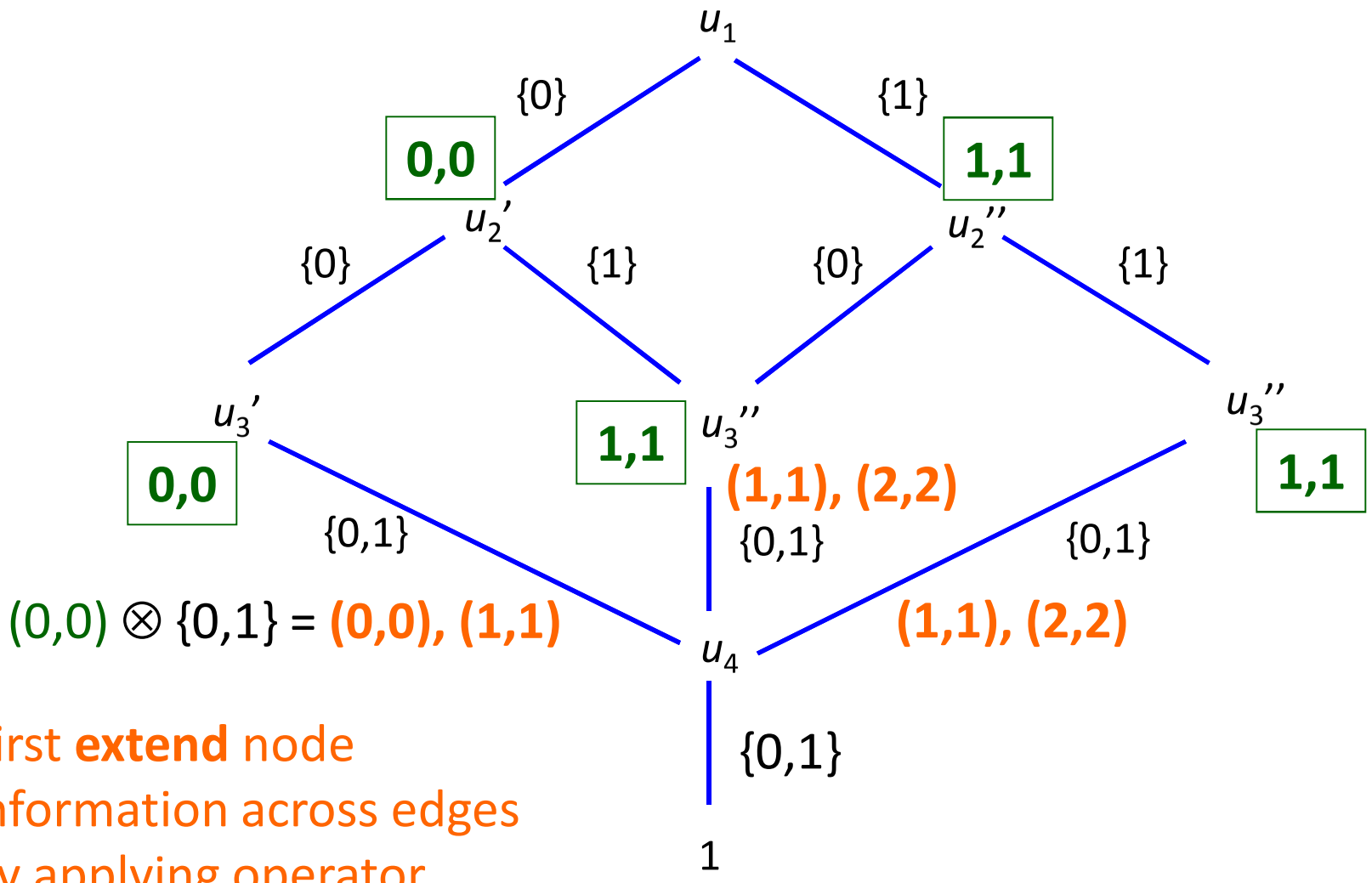
General Scheme for Filtering



We want to compute **top down** information—shortest & longest path length

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

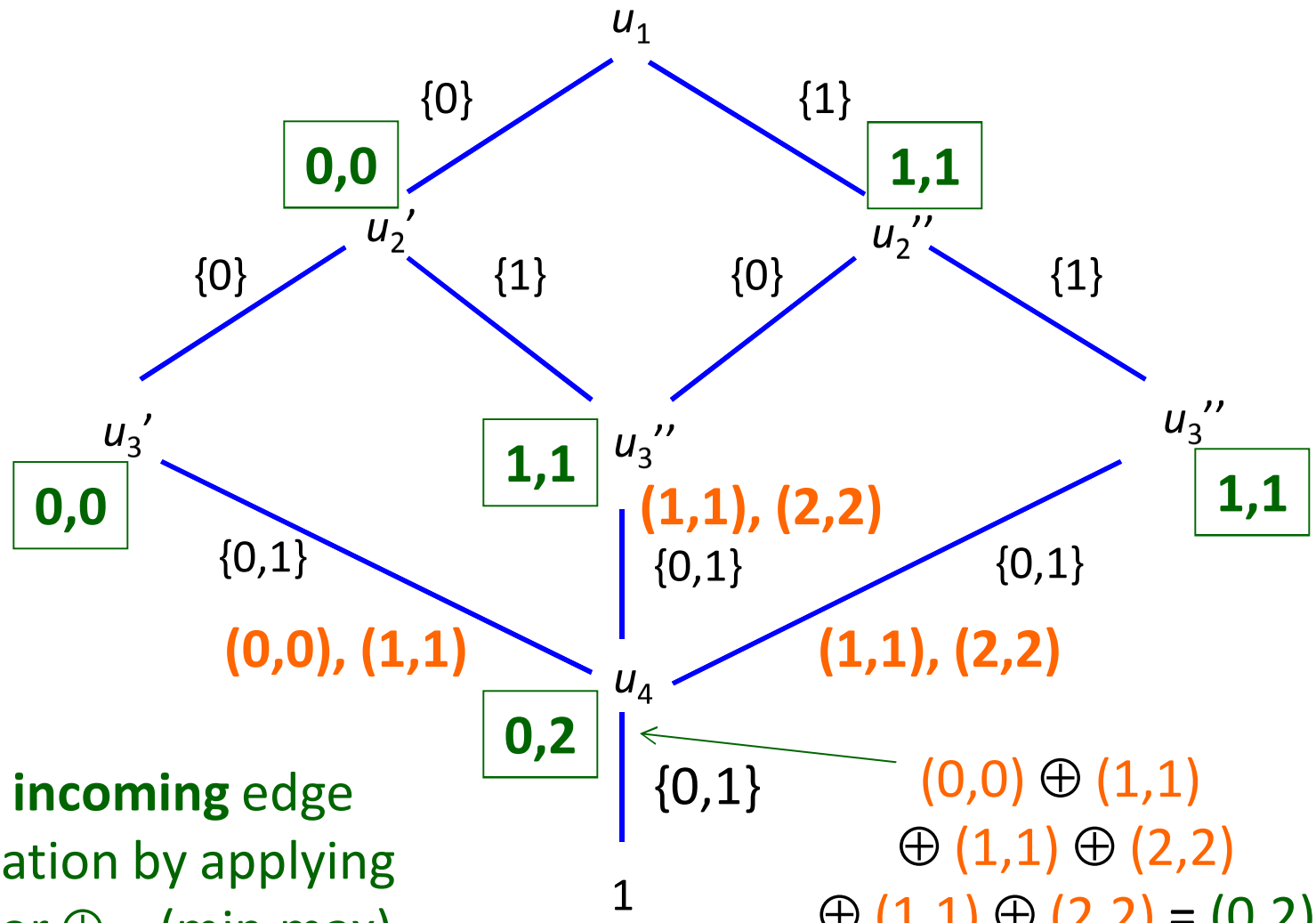
General Scheme for Filtering



First **extend** node information across edges by applying operator ⊗ = +

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

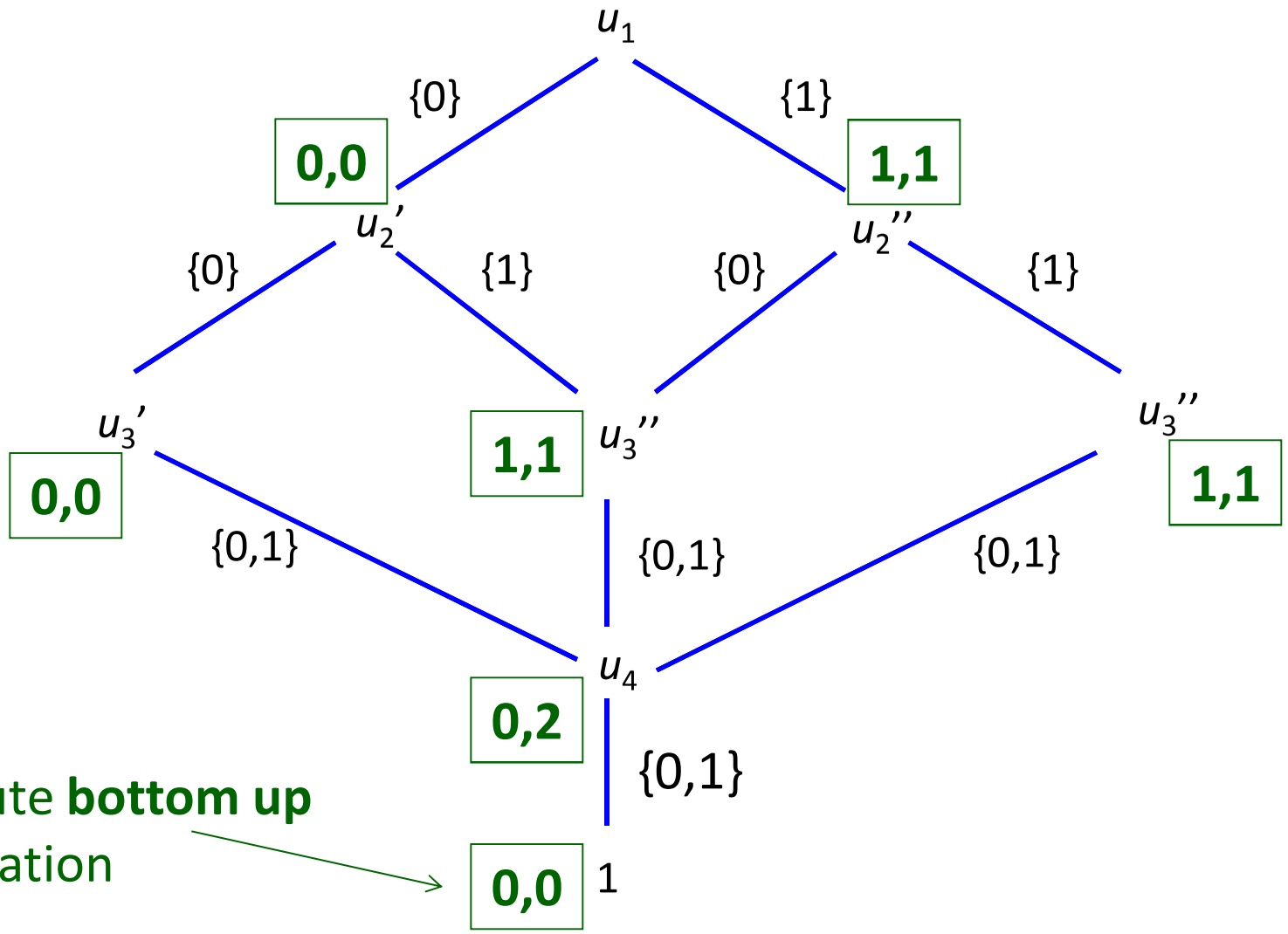
General Scheme for Filtering



Merge incoming edge information by applying operator ⊕ = (min,max)

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

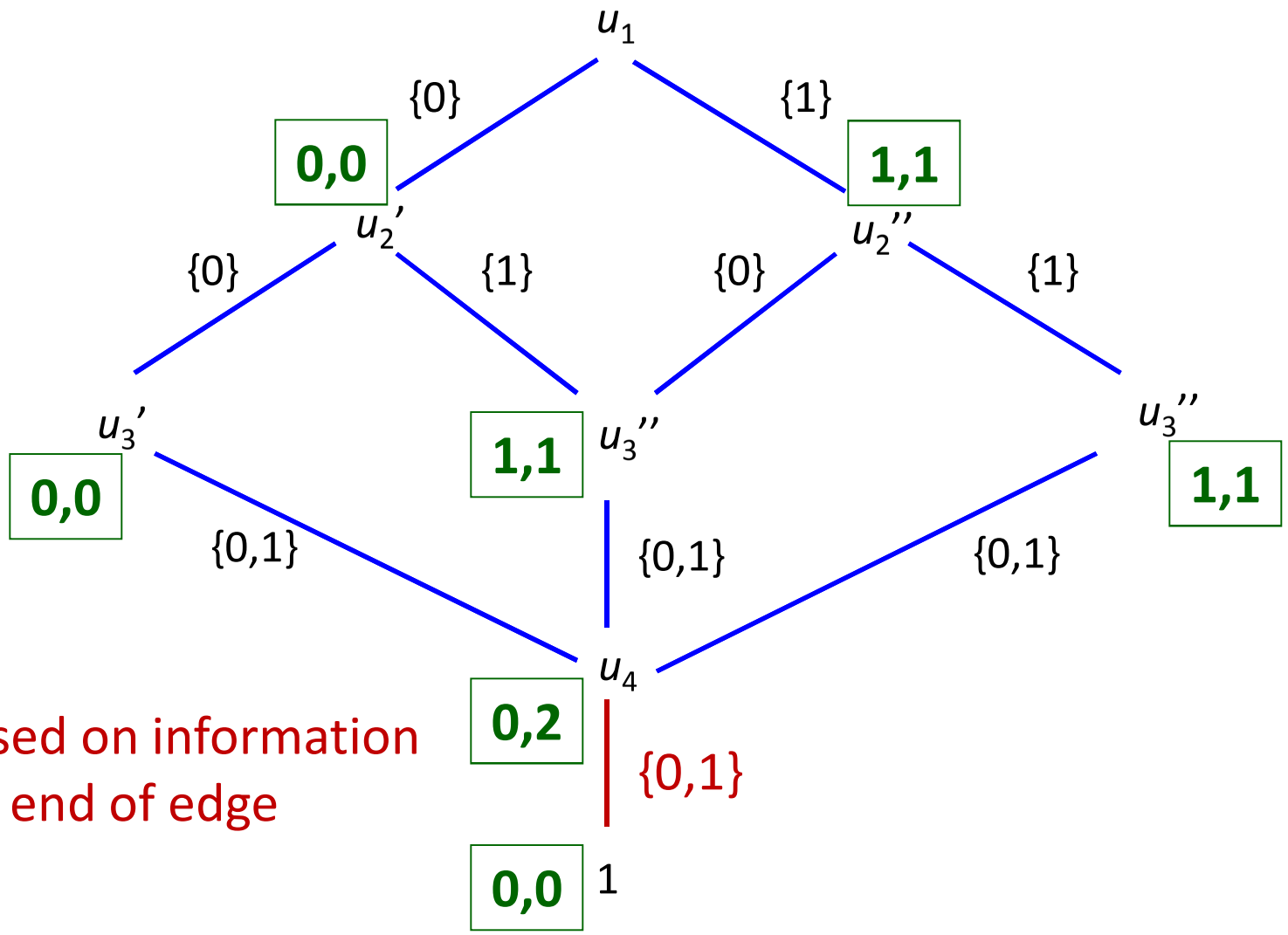
General Scheme for Filtering



Compute **bottom up** information

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

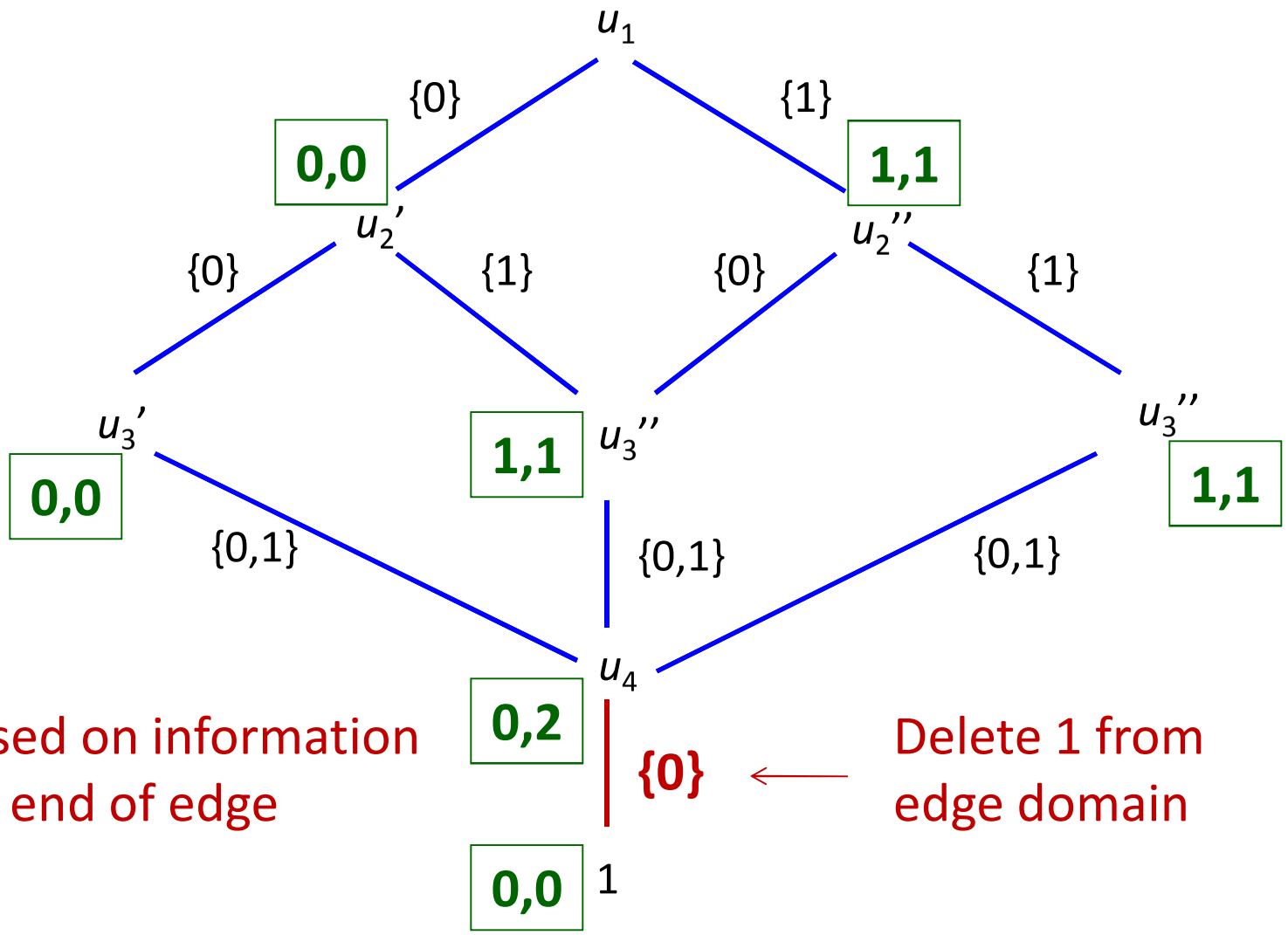
General Scheme for Filtering



Filter based on information at either end of edge

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

General Scheme for Filtering

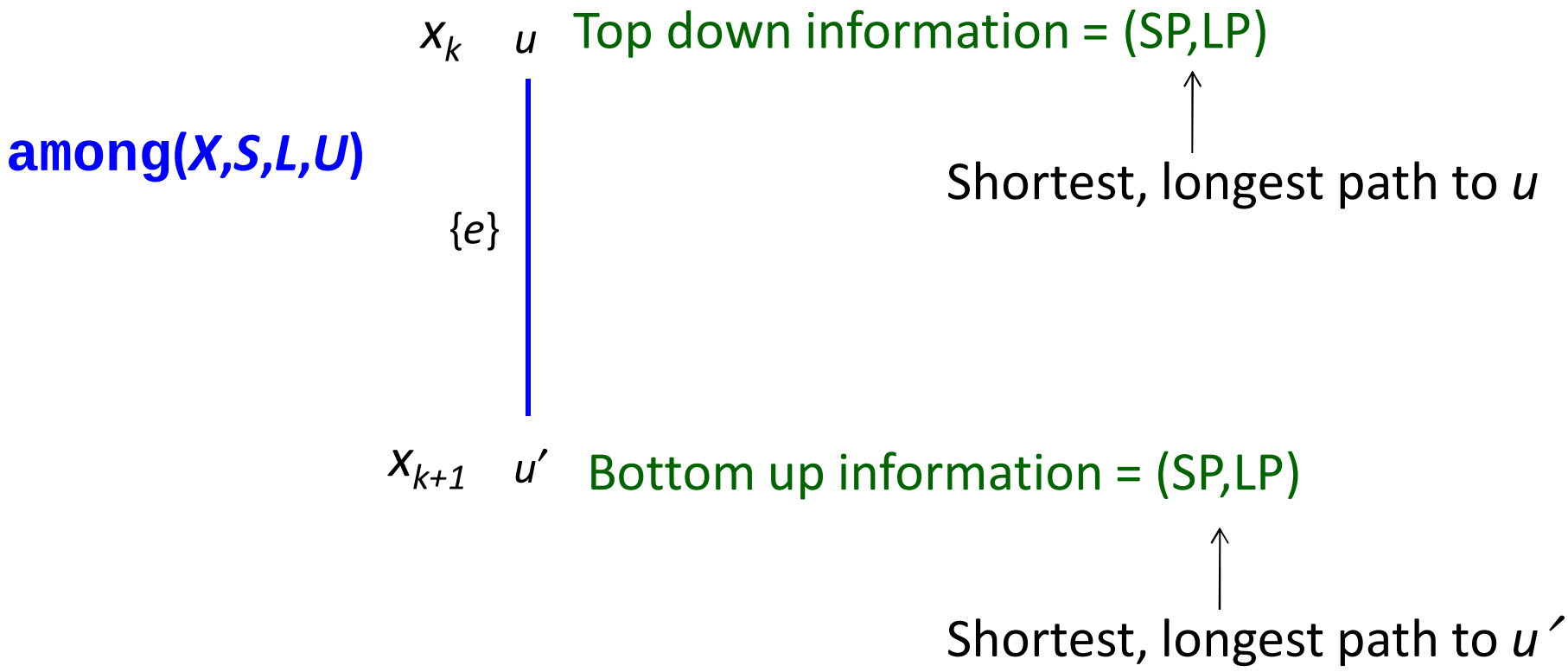


Filter based on information at either end of edge

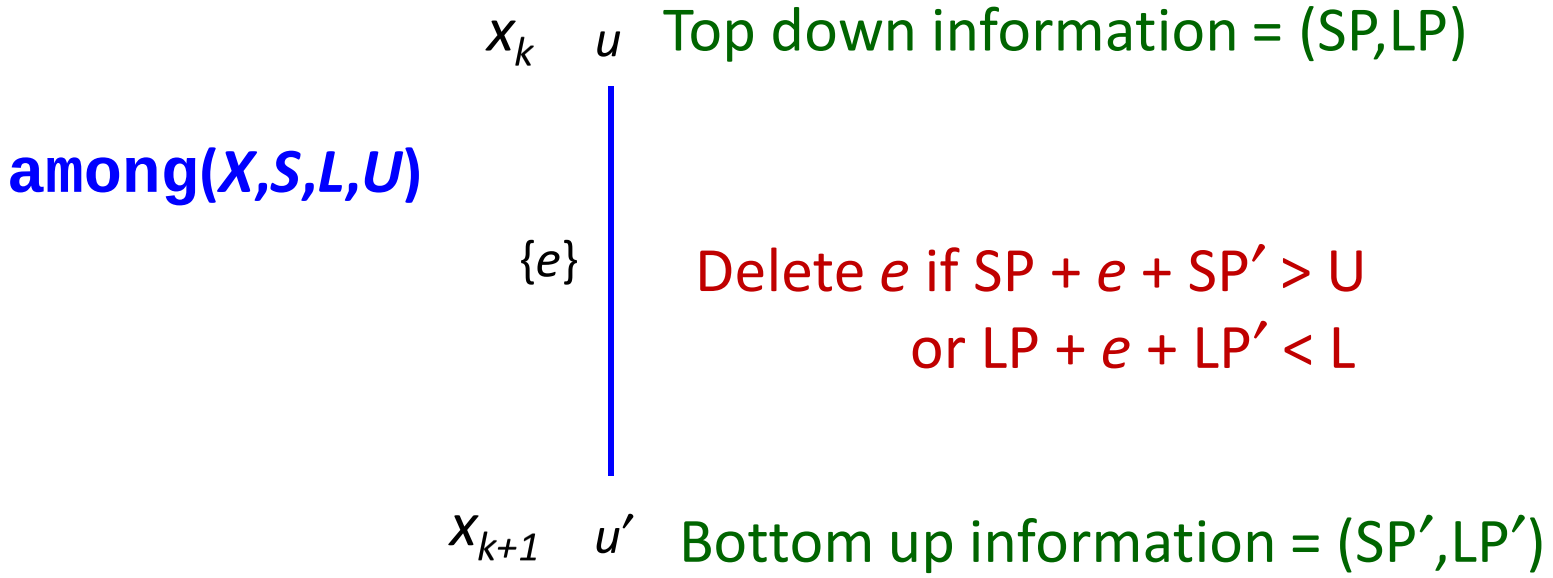
Delete 1 from edge domain

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)

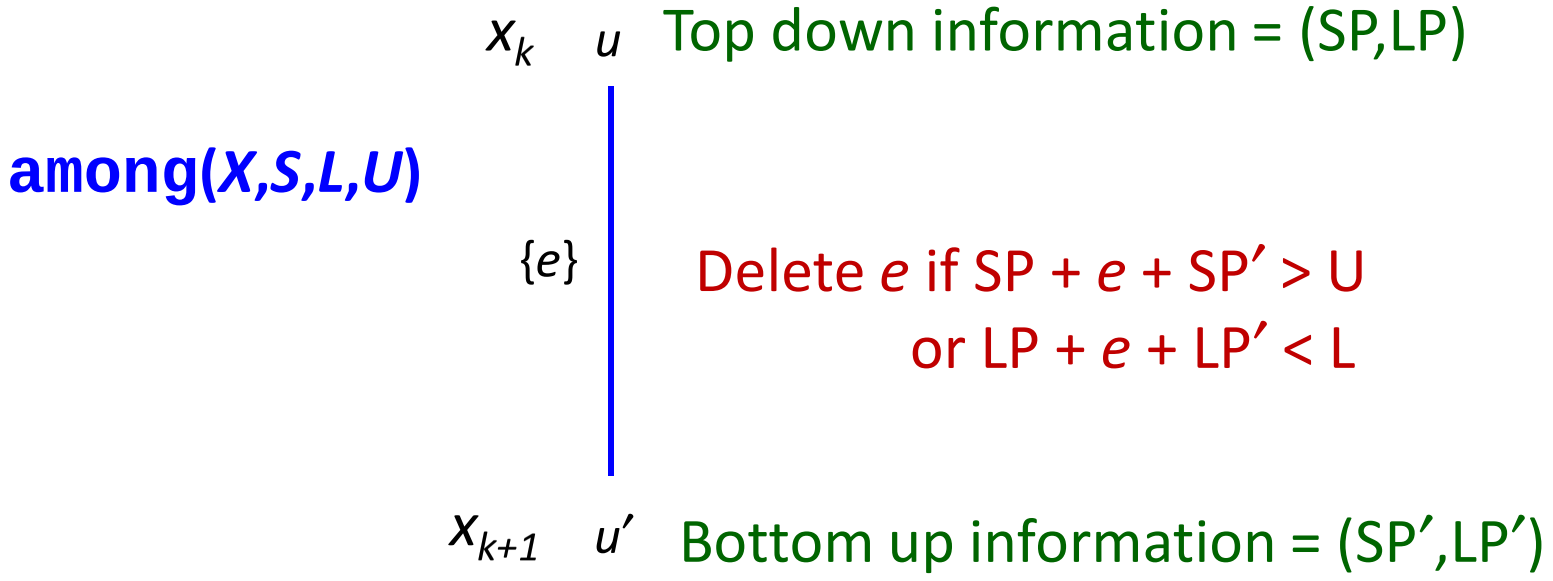
General Scheme for Filtering



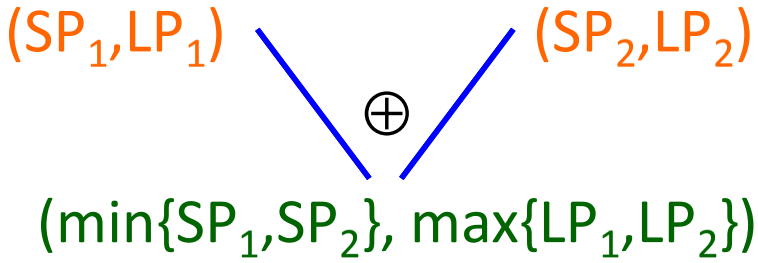
General Scheme for Filtering



General Scheme for Filtering

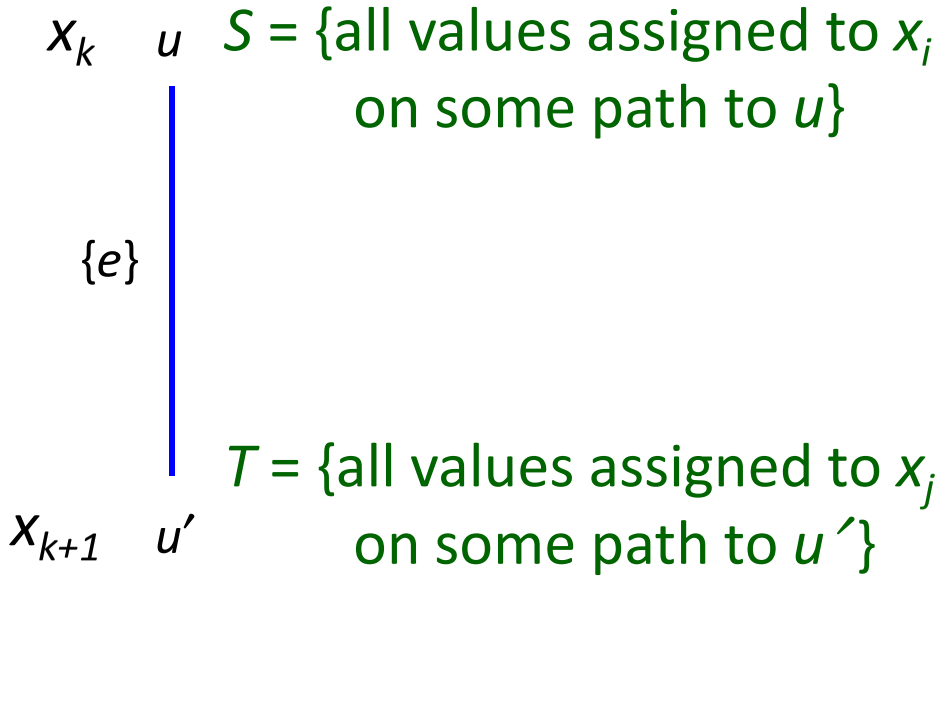


$(SP,LP) \otimes \{e\} = (SP + e,\ LP + e)$



General Scheme for Filtering

Any binary constraint B with variables x_i, x_j



General Scheme for Filtering

Any binary constraint B with variables x_i, x_j

x_k u S = {all values assigned to x_i on some path to u}

{e} Delete e if $k = j$ and $(x_i,x_j) = (v,e)$ satisfies B for no $v \in S$ or $\underline{k} = i$ and $(x_i,x_j) = (v,e)$ satisfies B for no $v \in T$

x_{k+1} u' T = {all values assigned to x_j on some path to u'}

General Scheme for Filtering

Any binary constraint B with variables x_i, x_j

x_k u S = {all values assigned to x_i on some path to u}

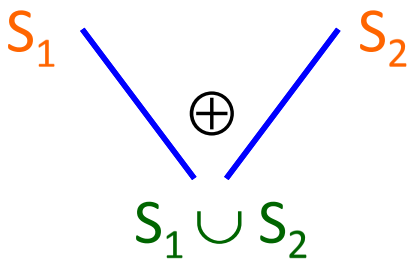
Delete e if $k = j$ and $(x_i, x_j) = (v, e)$ satisfies B for no $v \in S$

or $\underline{k} = i$ and $(x_i, x_j) = (v, e)$ satisfies B for no $v \in T$

{e}

x_{k+1} u' $\underline{T}$ = {all values assigned to x_j on some path to u'}

$S \otimes \{e\} = \{e\}$ if $k = i$, S otherwise



General Scheme for Filtering

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Any binary constraint *B* with variables x_i, x_j

Theorem. This scheme achieves MDD consistency for any binary constraint *B* in time that is polynomial in size of *B* and the MDD.

General Scheme for Filtering

Carnegie Mellon Tepper SCHOOL OF BUSINESS

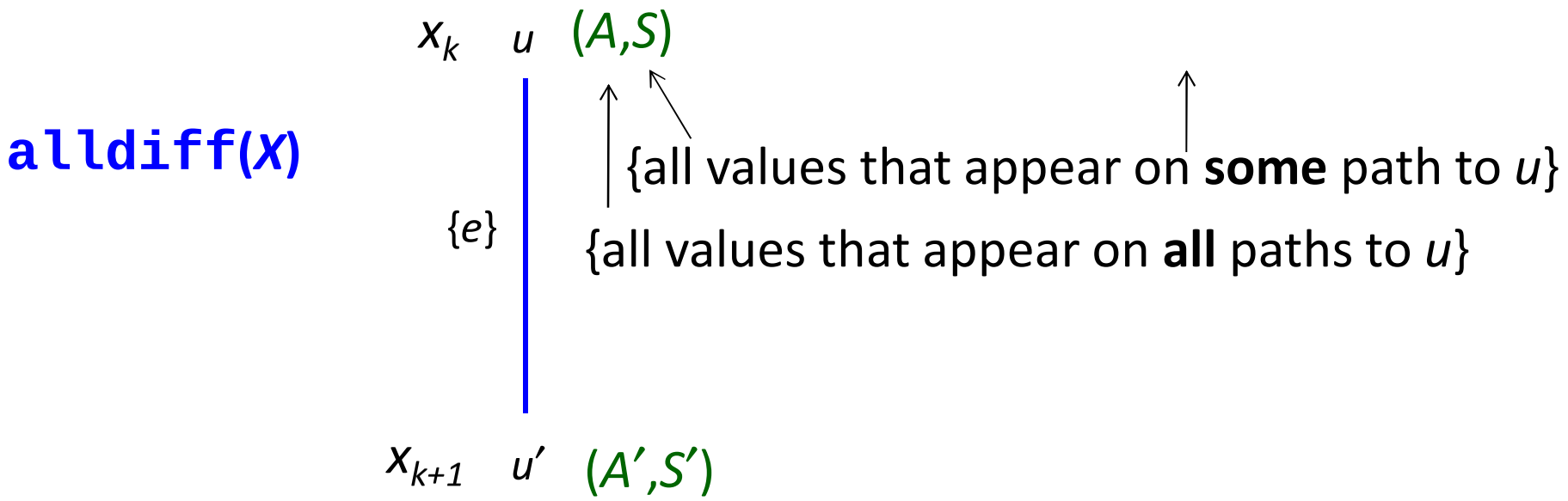
Any binary constraint *B* with variables x_i, x_j

Theorem. This scheme achieves MDD consistency for any binary constraint *B* in time that is polynomial in size of *B* and the MDD.

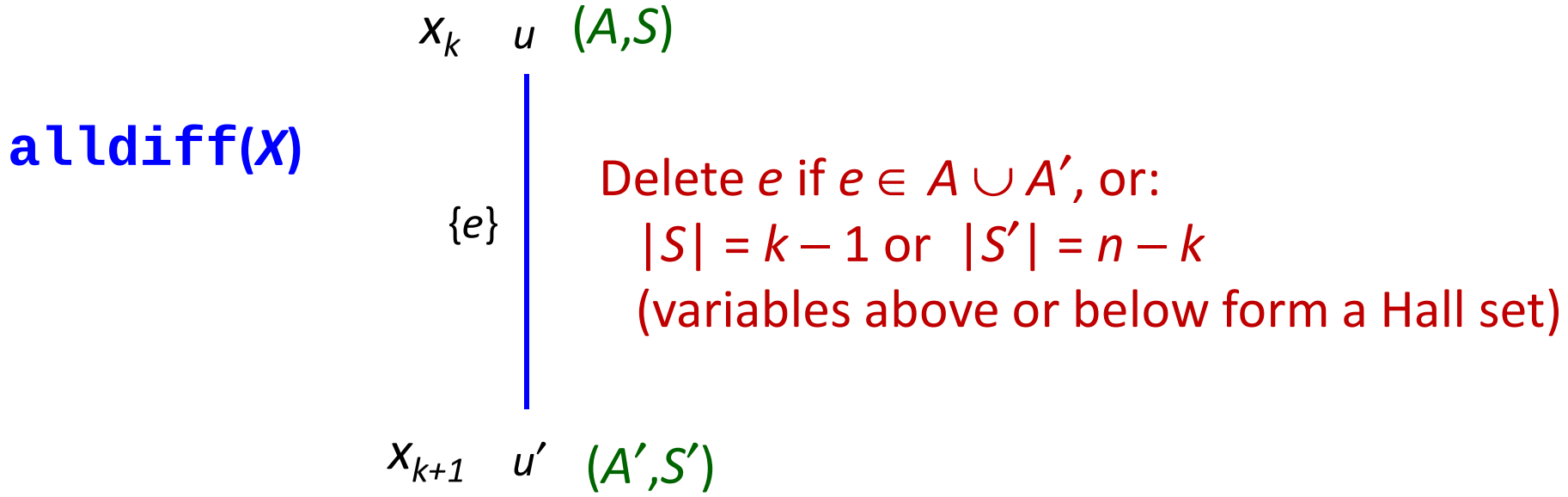
Corollary. Polytime MDD consistency for:

equality, not-equal constraints, `element` constraint, etc.

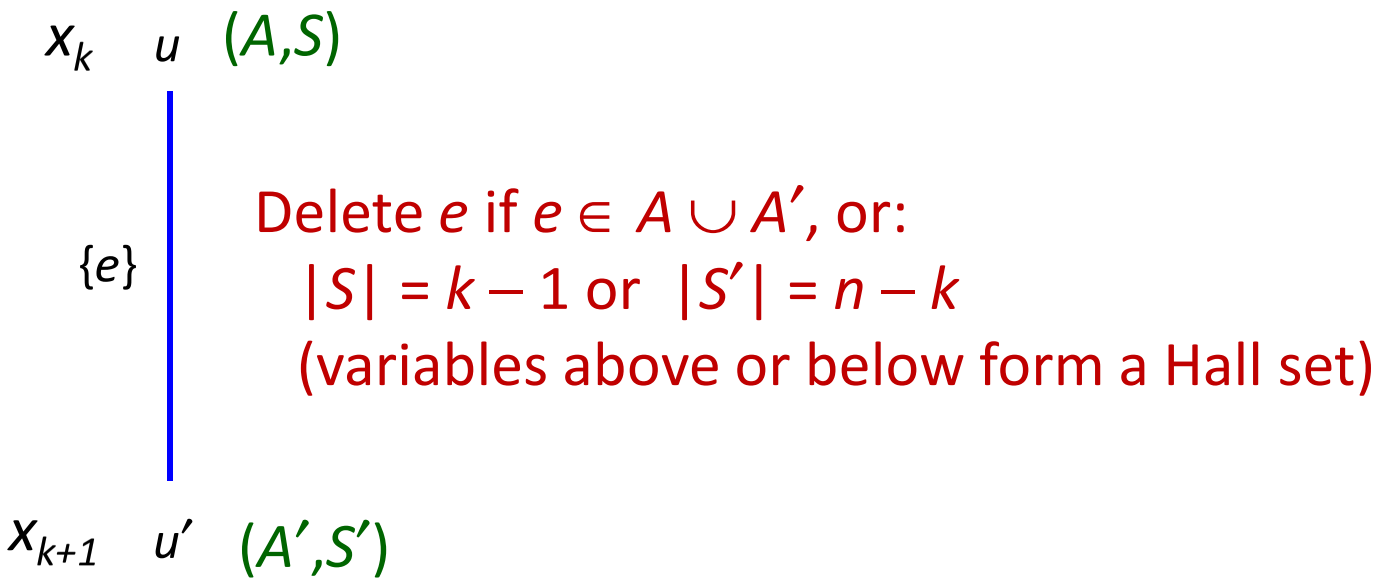
General Scheme for Filtering



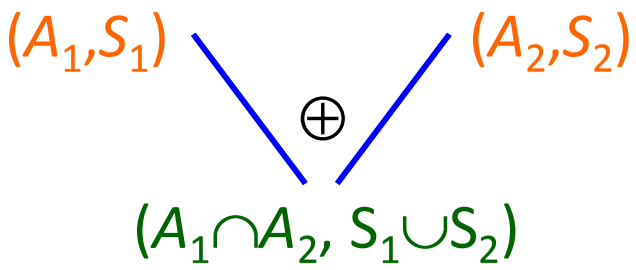
General Scheme for Filtering



General Scheme for Filtering



$(A,S) \otimes \{e\} = (A \cup \{e\}, S \cup \{e\})$



General Scheme for Filtering

Unary resource constraint

x_i = activity in position i of sequence

r_i, d_i = release time, deadline

Enforce `alldiff`(X) plus:

x_k u EST

$\{e\}$

Earliest start time of x_k, given previous activities in sequence

Latest finish time of x_{k+1}, given subsequent activities in sequence

x_{k+1} u' LFT

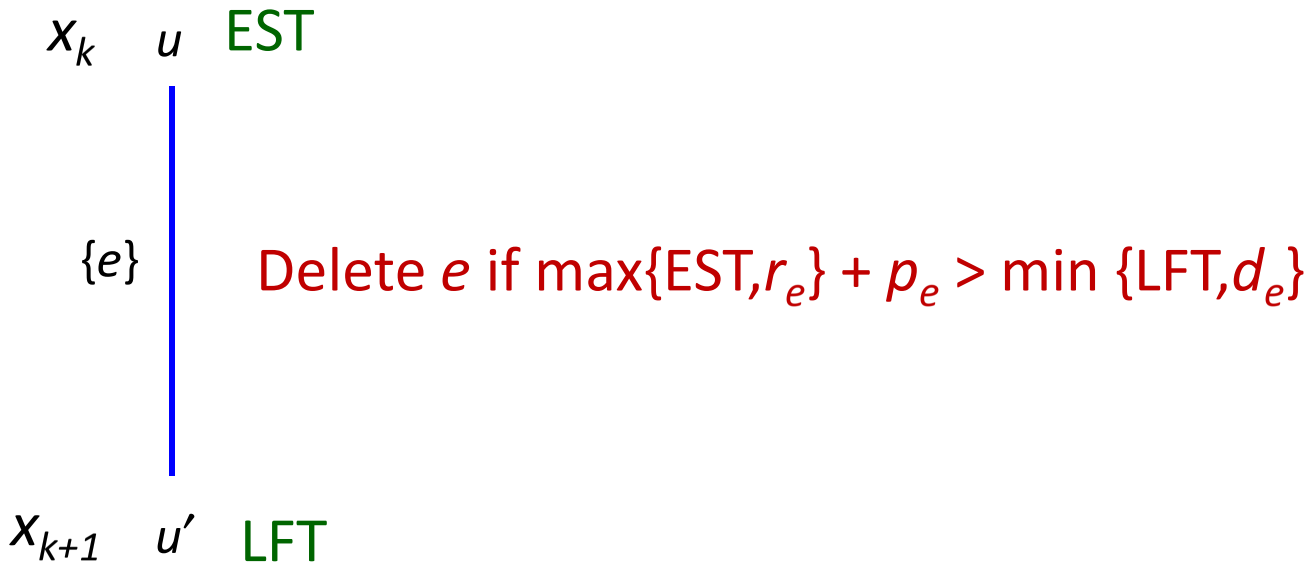
General Scheme for Filtering

Unary resource constraint

x_i = activity in position i of sequence

r_i, d_i = release time, deadline

Enforce `alldiff`(X) plus:



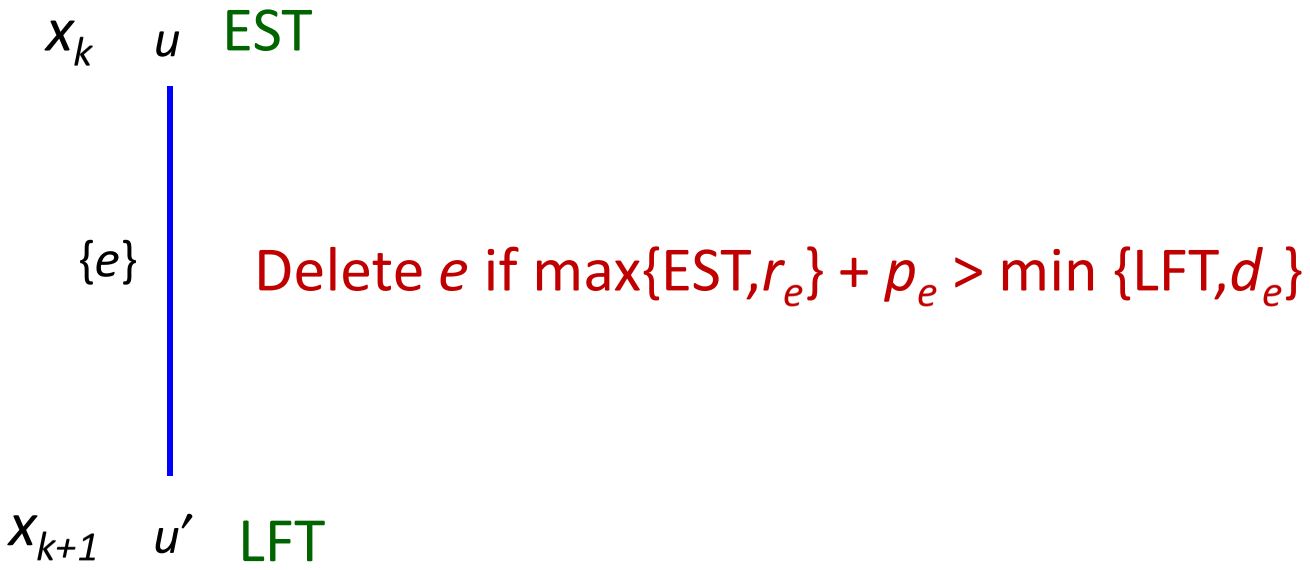
General Scheme for Filtering

Unary resource constraint

x_i = activity in position i of sequence

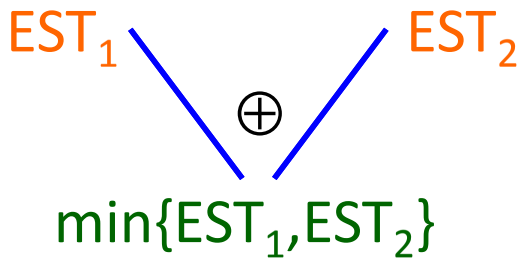
r_i, d_i = release time, deadline

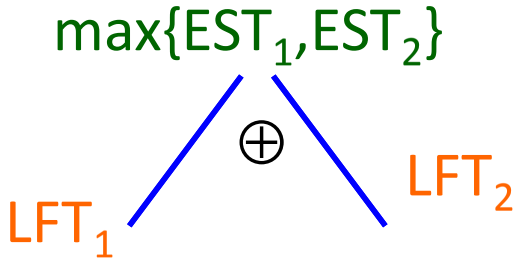
Enforce `alldiff(`X`)` plus:



$\text{EST} \otimes \{e\} = \max\{\text{EST} + p_e, r_e + p_e\}$

$\text{LFT} \otimes \{e\} = \min\{\text{LFT} - p_e, d_e - p_e\}$





Solver Architecture

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Our MDD-based solver is built around this general scheme for splitting and filtering.
- Plug in constraint-specific (approximate) equivalence test for splitting.
- Plug in constraint-specific $\otimes$, $\oplus$ for filtering.

Experiments – Previous work

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Multiple `alldiff` constraints
  - Andersen et al., CP 2007
  - MDDs of width 5
  - Reduce search tree of 1 million+ nodes to 1 node.
  - Reduce time by factor of 30
  - Using old software
- Equality constraints $\Sigma_j\, g_j(x_j) = b$
  - Hadzic et al., CPAIOR 2008
- Certain configuration problems
  - Hadzic et al, CP 2008

Experiments

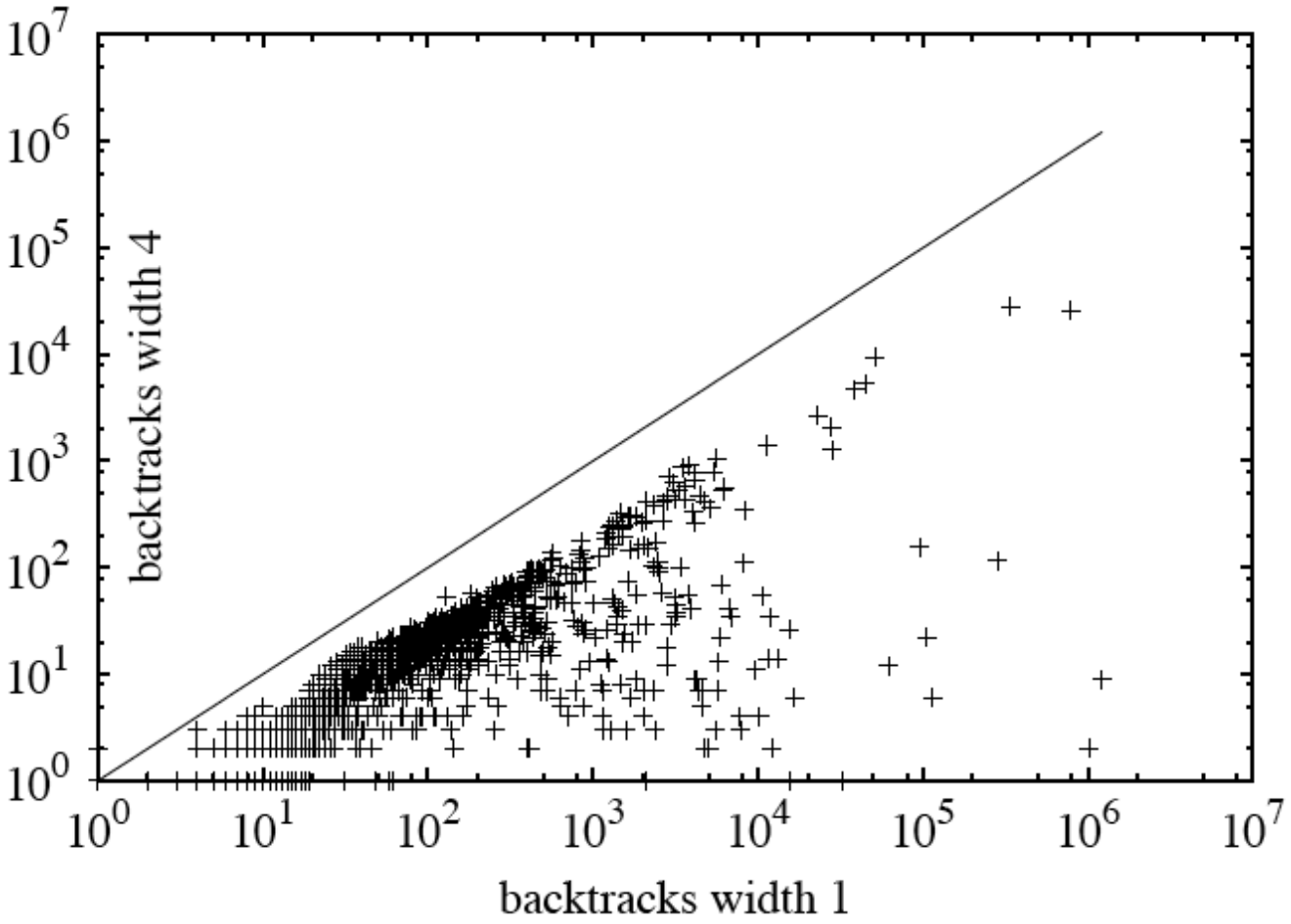
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Random multiple `among` constraints
- Nurse rostering instances
- Compare domain store (conventional filtering) with MDD store
  - Using MDDs of increasing widths.

Experiments

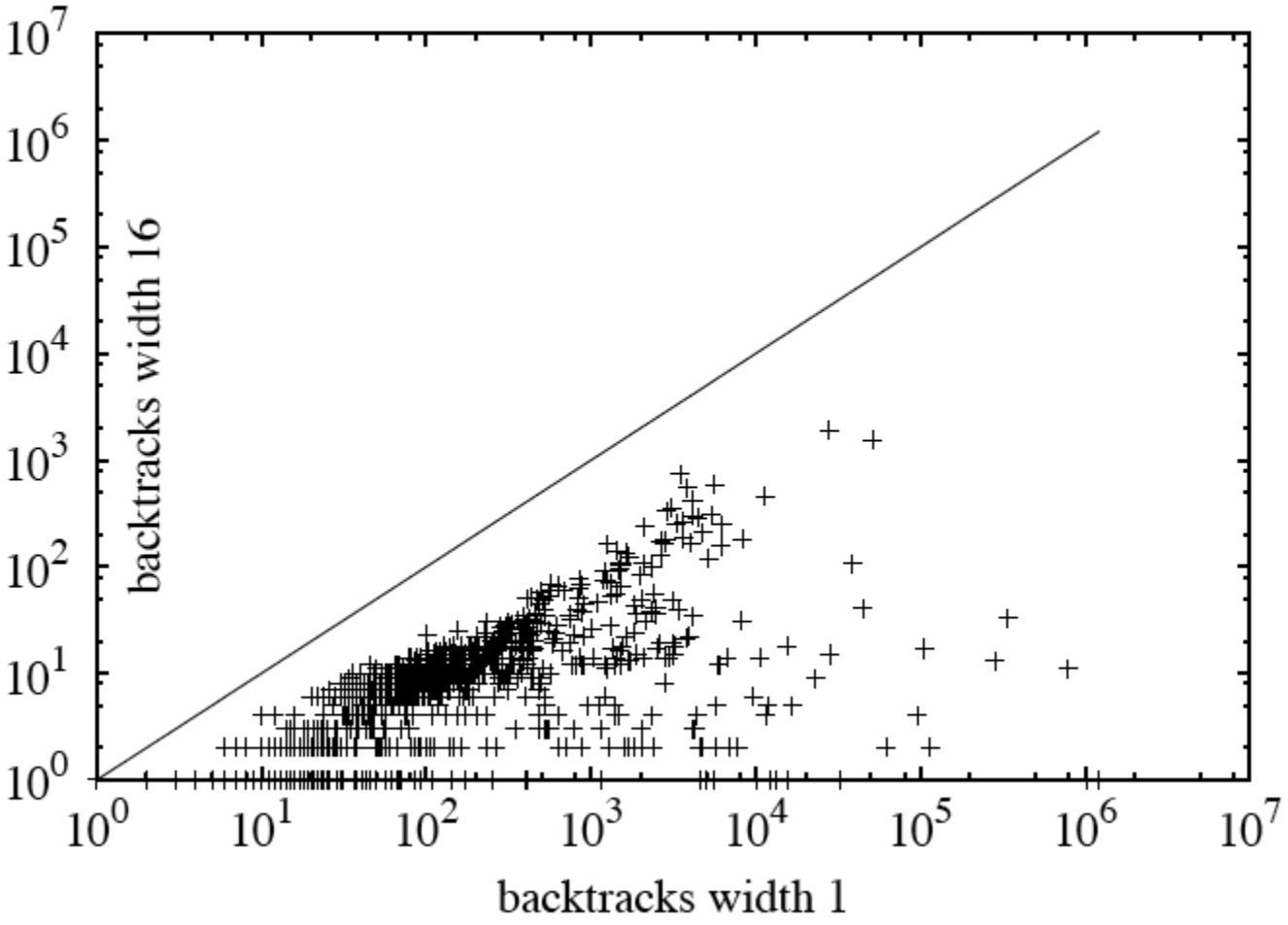
- Random multiple among constraints
  - 50 binary variables total
  - 5 variables per among constraint
  - Indices chosen from normal distribution with mean chosen from U[1..50] mod 50, and standard deviation 2.5
  - So, indices in a given among are near-consecutive, typical of applications.
  - (L,U) = (2,3)
  - Number of amongs varies from 5 to 200 (in steps of 5), 100 instances for each number.
  - This range includes phase transition.

Multiple Amongs: Backtracks



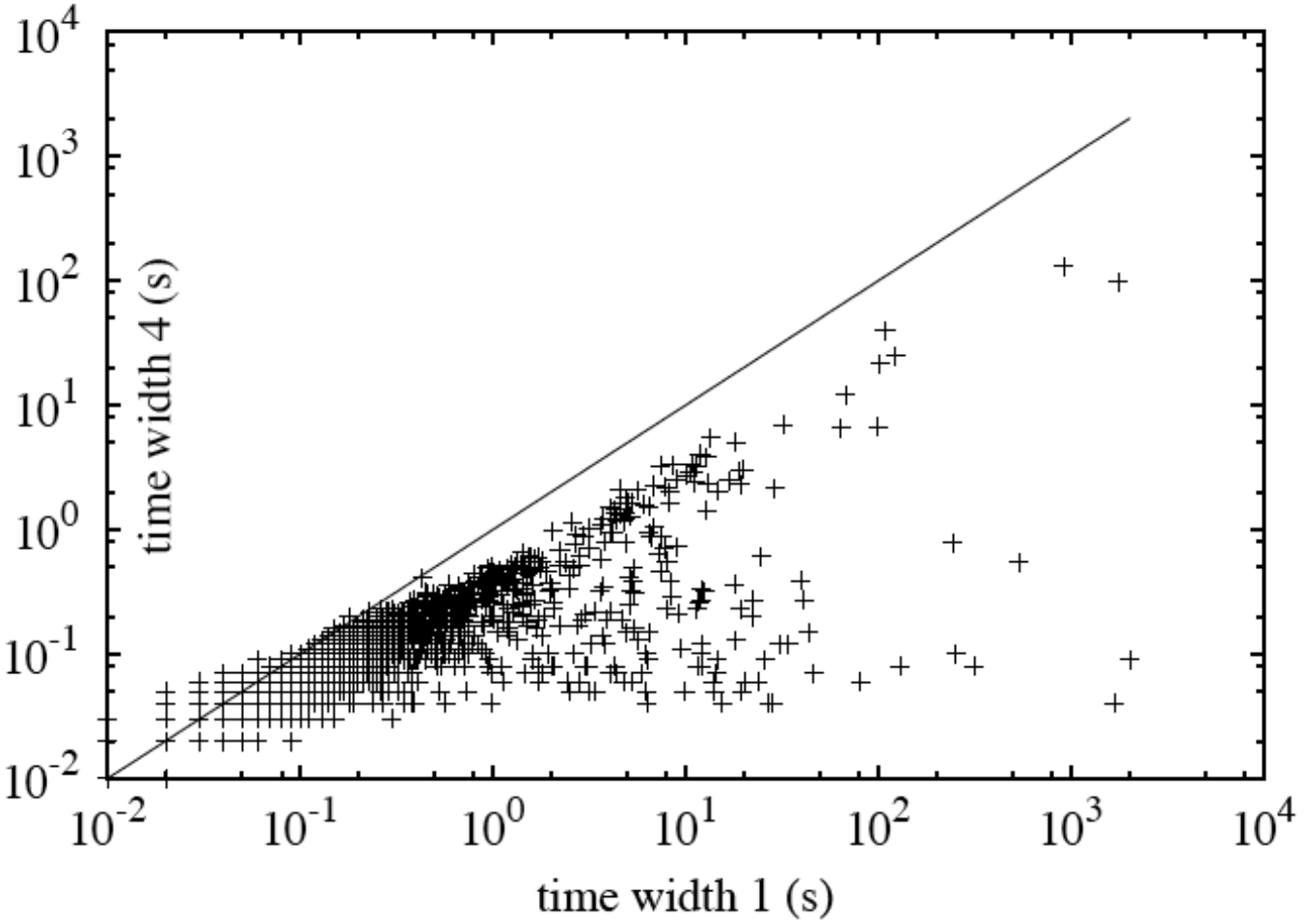
Domain store vs width **4** MDD

Multiple Amongs: Backtracks



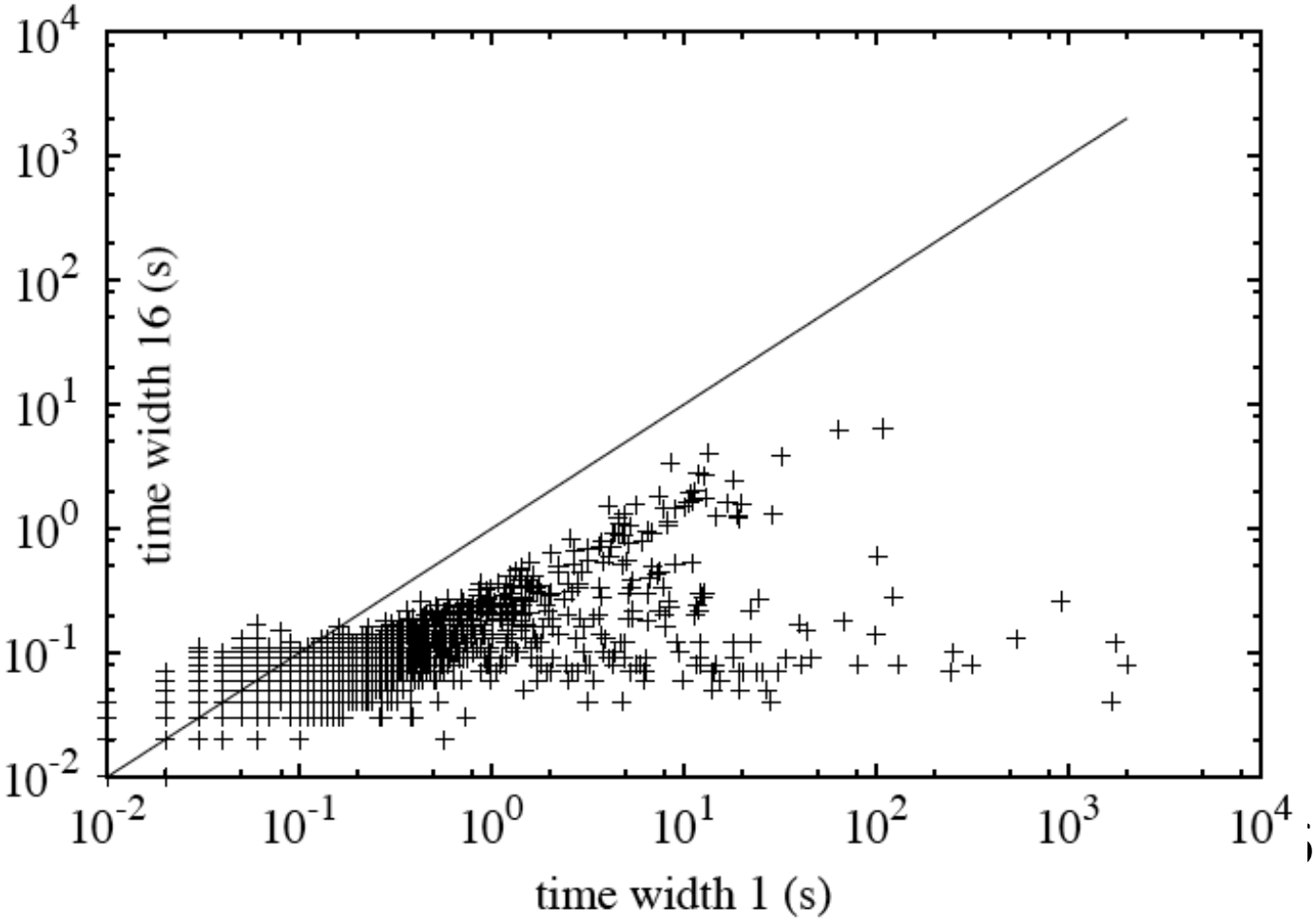
Domain store vs width **16** MDD

Multiple Amongs: Running Time



Domain store vs width **4** MDD

Multiple Amongs: Running Time

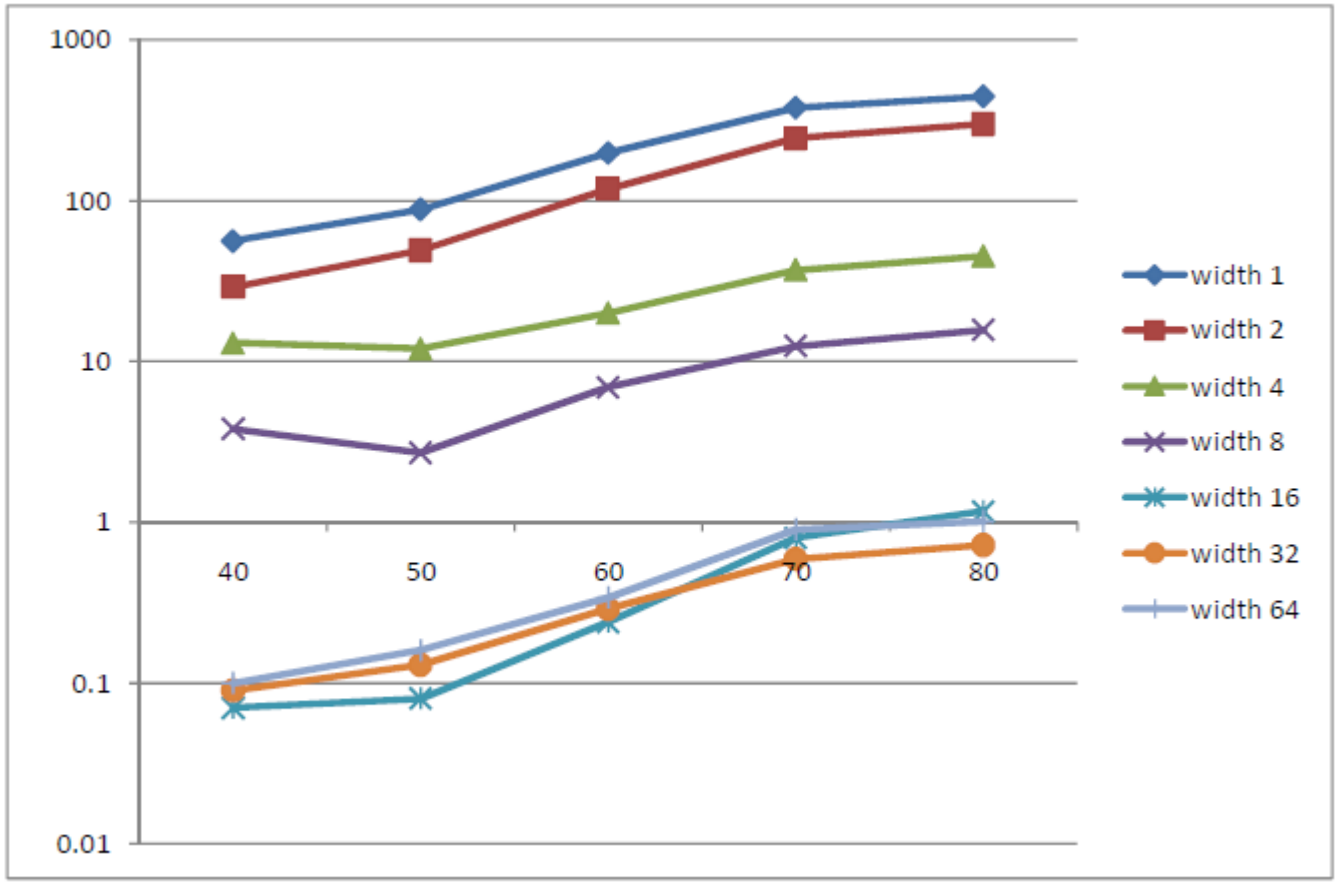


Domain store vs width **16** MDD

Experiments

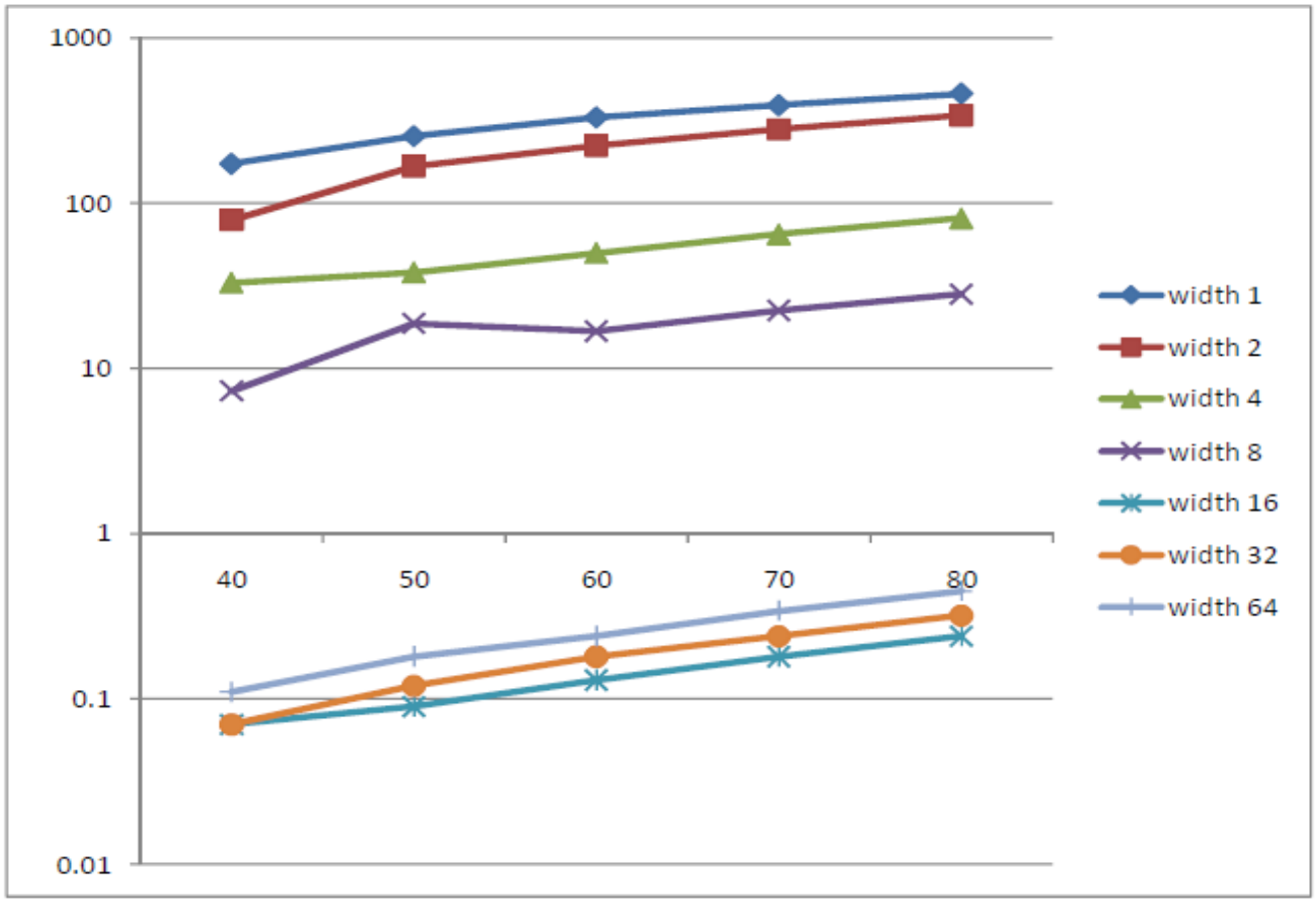
- Nurse rostering instances
  - Based on van Hoeve et al., *Constraints* 2009
  - Work 4-5 days per week
  - Max A days every B days (Max A/B)
  - Min C days every D days (Min A/B)
  - Time horizon ranges from 40 to 80 days
- Three problem classes
  - I: max 6/8, min 22/30
  - II: max 6/9, min 20/30
  - III: max7/9, min 22/30

Nurse rostering problems



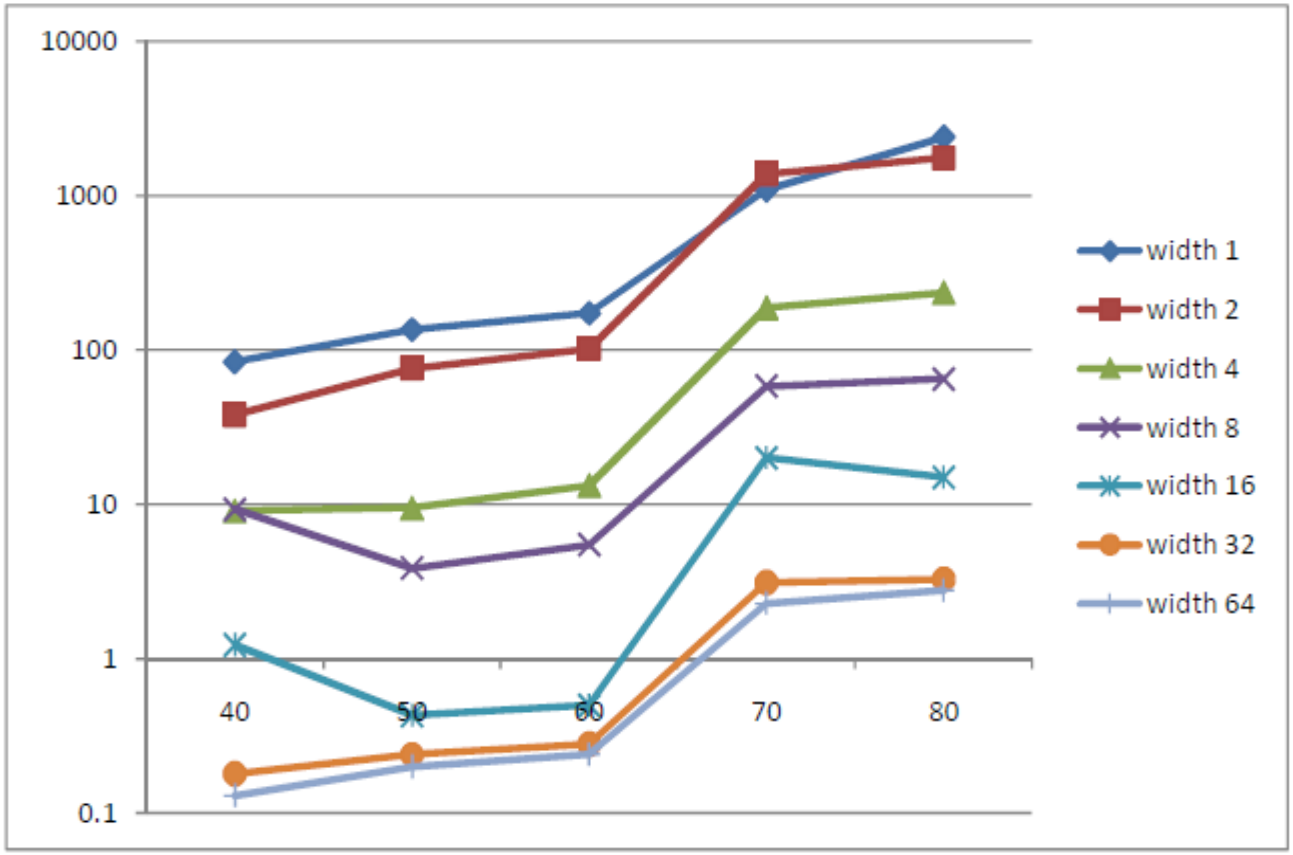
Computation time (sec) vs time horizon, Class I

Nurse rostering problems



Computation time (sec) vs time horizon, Class II

Nurse rostering problems



Computation time (sec) vs time horizon, Class III

Conclusions

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- MDD-based propagation can be carried out according to a general scheme.
  - Equivalence testing for node splitting
  - Information operators for filtering
  - Yet the scheme allows exploitation of constraint-specific structure
- MDD-based constraint solving yields substantial speedup over domain store for multiple among constraints.
- Intensive processing at search nodes can pay off when more information is communicated between constraints.