

# CONSTRAINT PROGRAMMING FOR REAL

2009-05-27

Christian Schulte, KTH, ICT

# Constraint Programming for Fun

2

- What is constraint programming?

**Sudoku is constraint programming**

- ... as a reminder ... for real, later

3

# Sudoku

...is constraint programming!

# Sudoku

4

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Assign blank fields digits such that:  
digits distinct per **rows**, columns, blocks

# Sudoku

5

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Assign blank fields digits such that:  
digits distinct per rows, **columns**, blocks

# Sudoku

6

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Assign blank fields digits such that:  
digits distinct per rows, columns, **blocks**

# Block Propagation

7

	<b>8</b>	
	<b>6</b>	<b>3</b>

- No field in block can take digits 3,6,8

# Block Propagation

8

1,2,4,5,7,9	<b>8</b>	1,2,4,5,7,9
1,2,4,5,7,9	<b>6</b>	<b>3</b>
1,2,4,5,7,9	1,2,4,5,7,9	1,2,4,5,7,9

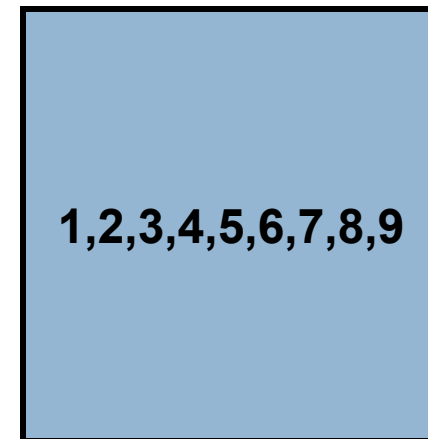
- No field in block can take digits 3,6,8
  - propagate to other fields in block
- Rows and columns: likewise



# Propagation

9

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

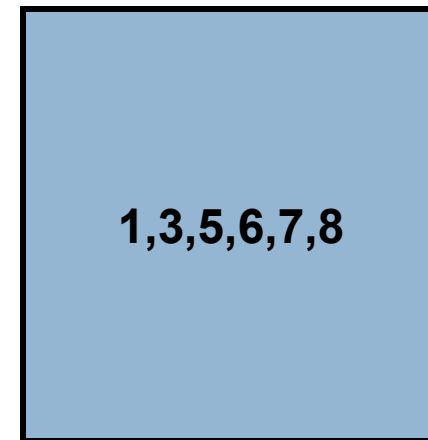


- Prune digits from fields such that:  
digits distinct per rows, columns, blocks

# Propagation

10

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

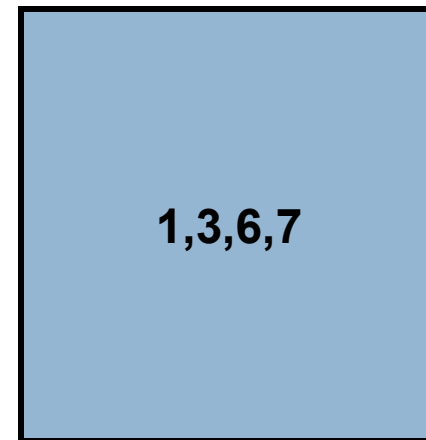


- Prune digits from fields such that:  
digits distinct per **rows**, columns, blocks

# Propagation

11

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

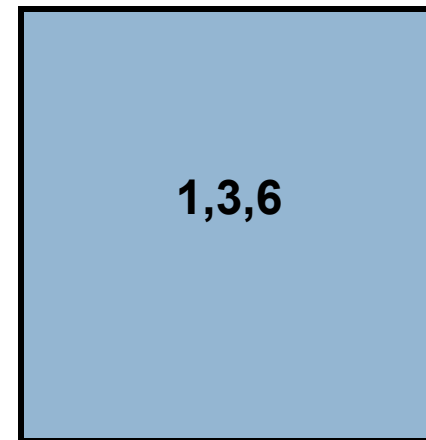


- Prune digits from fields such that:  
digits distinct per rows, **columns**, blocks

# Propagation

12

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			



- Prune digits from fields such that:  
digits distinct per rows, columns, **blocks**

# Iterated Propagation

13

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Iterate propagation for rows, columns, blocks
- What if no assignment: search... later

# Sudoku is Constraint Programming

14

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- **Variables:** fields
  - take **values:** digits
  - maintain set of **possible** values
- **Constraints:** distinct
  - relation among values for variables

- Modeling: variables, values, constraints
- Solving: propagation, search

# Constraint Programming

15

- Variable domains
  - finite domain integer, finite sets, multisets, intervals, ...
  
- Constraints
  - distinct, arithmetic, scheduling, graphs, ...
  
- Solving
  - propagation, branching, exploration, ...
  
- Modeling
  - variables, values, constraints, heuristics, symmetries, ...

# Constraint Programming for Real

16

- Key ideas and principles
  - constraint propagation and search
- Why does constraint programming matter?
- Excursions
  - capturing structure:                   distinct reconsidered
  - local reasoning:                       admitting failure
  - user-defined constraints:           rostering
  - compositional modeling:           scheduling [if time allows]
- Summary
  - strength and challenges
  - two entry pointers



17

# Key Ideas and Principles

# Running Example: SMM

18

- Find distinct digits for letters such that

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline = \text{MONEY} \end{array}$$

# Constraint Model for SMM

19

- Variables:

$$S, E, N, D, M, O, R, Y \in \{0, \dots, 9\}$$

- Constraints:

$$\text{distinct}(S, E, N, D, M, O, R, Y)$$

$$\begin{aligned} & 1000 \times S + 100 \times E + 10 \times N + D \\ + & 1000 \times M + 100 \times O + 10 \times R + E \\ = & 10000 \times M + 1000 \times O + 100 \times N + 10 \times E + Y \end{aligned}$$

$$S \neq 0$$

$$M \neq 0$$

# Solving SMM

20

- Find values for variables

such that

**all constraints satisfied**

# Finding a Solution

21

- Compute with possible values
  - rather than enumerating assignments
  
- Prune inconsistent values
  - constraint propagation
  
- Search
  - branch:                      define search tree
  - explore:                    explore search tree for solution

# Constraint Propagation

constraint store

propagators

constraint propagation

# Constraint Store

23

$x \in \{1, 2, 3, 4\}$   $y \in \{1, 2, 3, 4\}$   $z \in \{1, 2, 3, 4\}$

- Maps variables to possible values

# Constraint Store

24

finite domain constraints

$x \in \{1, 2, 3, 4\}$   $y \in \{1, 2, 3, 4\}$   $z \in \{1, 2, 3, 4\}$

- Maps variables to possible values
  - other domains: finite sets, float intervals, graphs, ...



# Propagators

25

- Implement constraints

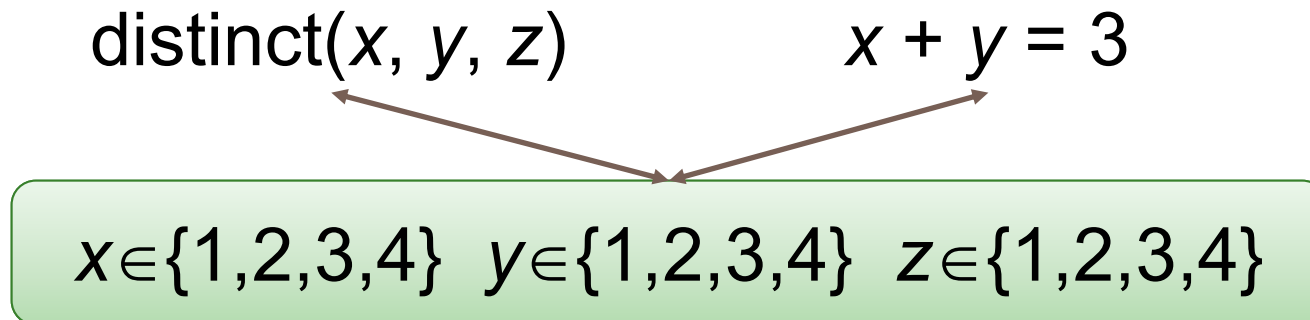
$\text{distinct}(x_1, \dots, x_n)$

$x + 2 \times y = z$

$\text{schedule}(t_1, \dots, t_n)$

# Propagators

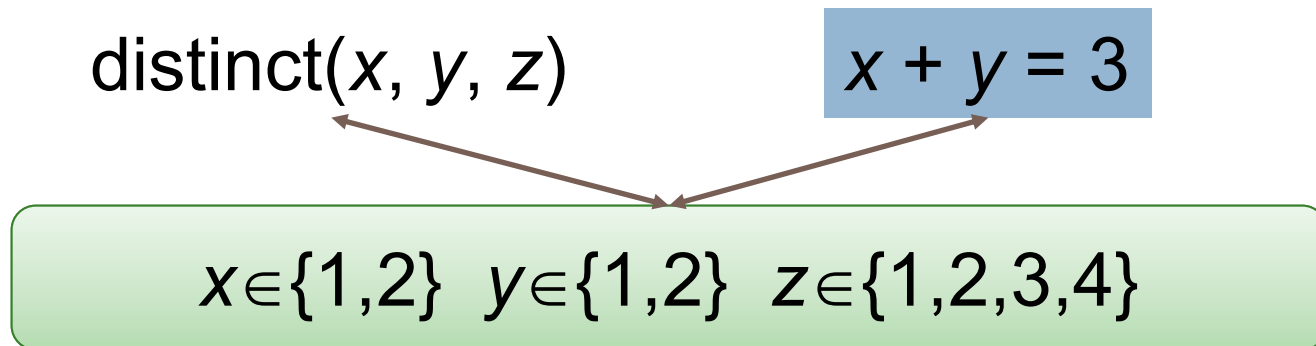
26



- Strengthen store by constraint propagation
  - prune values in conflict with implemented constraint

# Propagators

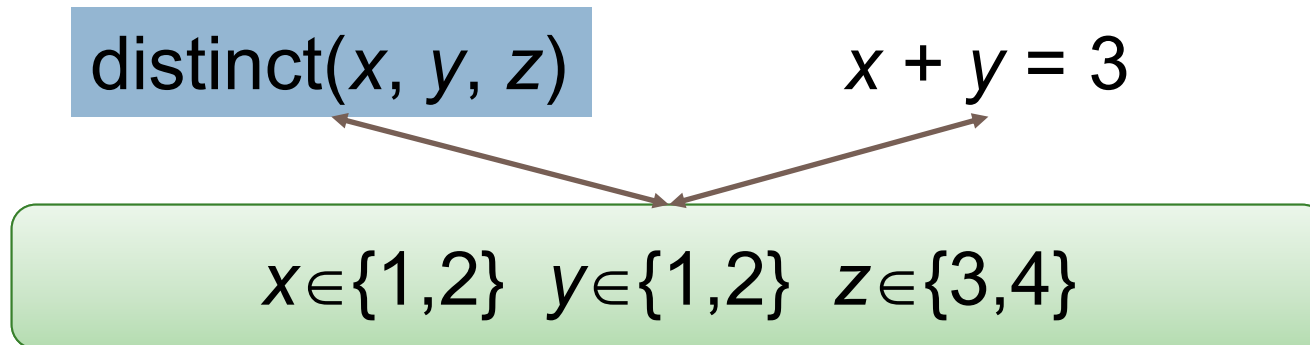
27



- Strengthen store by constraint propagation
  - prune values in conflict with implemented constraint

# Propagators

28



- Iterate propagator execution until fixpoint
  - no more pruning possible

# Propagation for SMM

29

## □ Results in store

$S \in \{9\}$     $E \in \{4, \dots, 7\}$     $N \in \{5, \dots, 8\}$     $D \in \{2, \dots, 8\}$   
 $M \in \{1\}$     $O \in \{0\}$     $R \in \{2, \dots, 8\}$     $Y \in \{2, \dots, 8\}$

## □ Propagation **alone** not sufficient!

- decompose into simpler sub-problems
- **branching**

# Constraints and Propagators

30

- Constraints state relations among variables
  - which value combinations satisfy constraint
- Propagators implement constraints
  - prune values in conflict with constraint
  - freedom of what to implement (more later)
- Constraint propagation executes propagators
  - until no more pruning possible (fixpoint)

# Well-behaved Propagators

31

- **Semantic: propagator implements constraint**
  - **correct**            no solution of constraint ever removed
  - **complete**        decision procedure for assignments  
                          propagation + search is complete
- **Operational: constraint propagation works**
  - **contracting**      values are removed
  - **monotonic**        stronger pruning only on stronger input
- **No restriction on**
  - **strength**         how much pruning
  - **how**               how propagator is implemented

# Search

branching

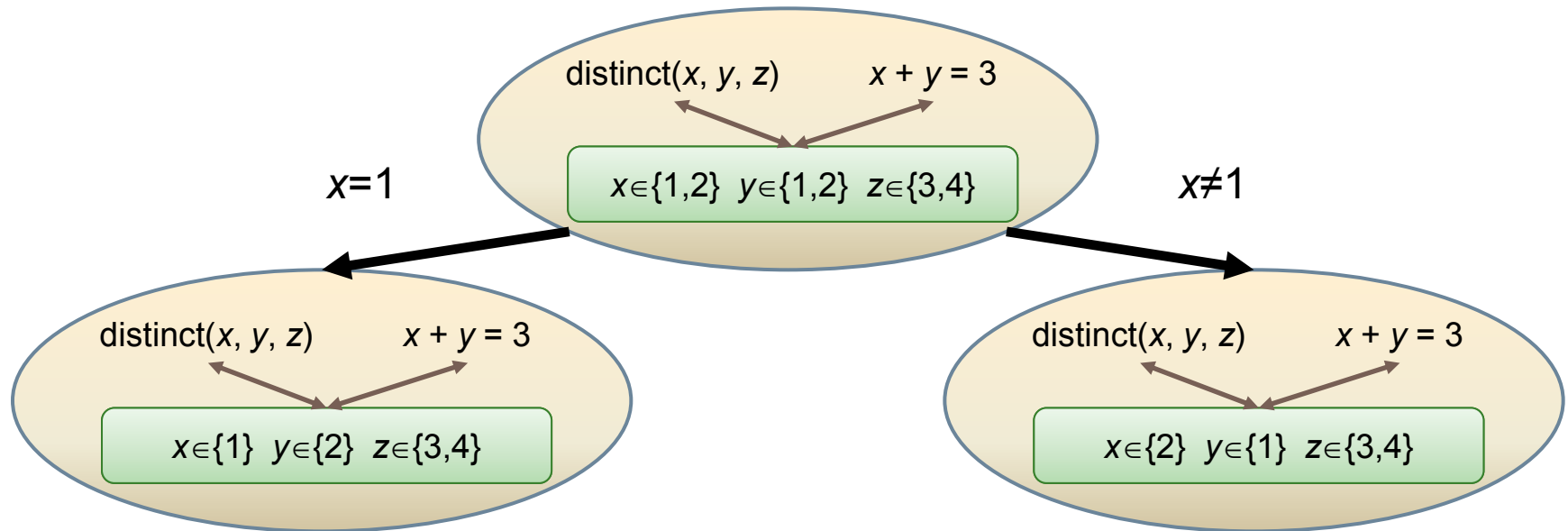
exploration

best solution search



# Branching

33



- Create subproblems with additional constraints
  - enables further propagation
  - defines search tree

# Example Branching Strategy

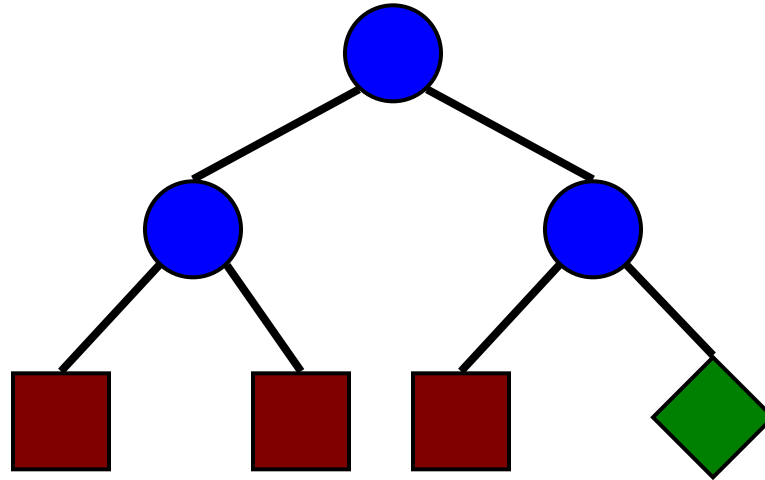
34

- Pick variable  $x$  with at least two values
- Pick value  $n$  from domain of  $x$
- Branch with

$x=n$                       and                       $x \neq n$

# Exploration

35



- Iterate propagation and branching
- Orthogonal: branching  $\Leftrightarrow$  exploration
  - exploration: interactive, parallel, ...
- Nodes:
  - **unsolved**
  - **failed**
  - **solved**

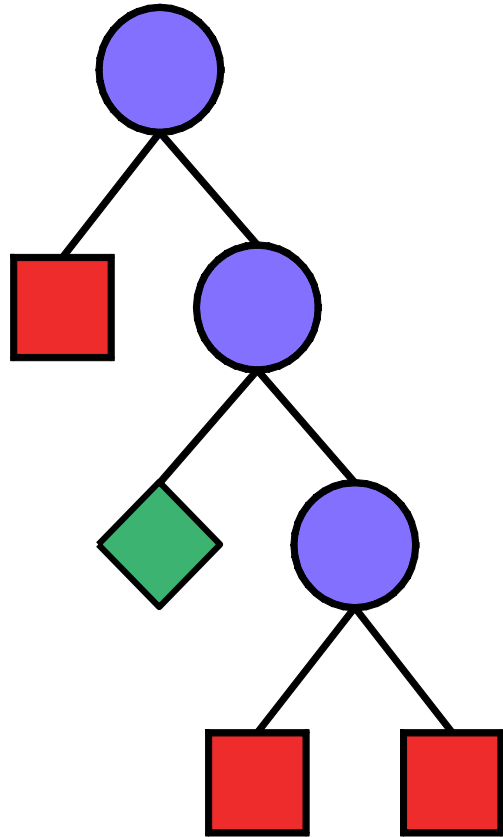
# Heuristics for Branching

36

- Which variable
  - least possible values (first-fail)
  - application dependent heuristic
- Which value
  - minimum, median, maximum
  - $x=n$                       or                       $x \neq n$
  - split with median  $n$
  - $x < n$                       or                       $x \geq n$
- Problem specific
  - most loaded resource, task with least slack, ...
  - order tasks on same resource, ...

# SMM: Solution With First-fail

37



$$\begin{array}{r} \text{SEND} \\ + \text{ MORE} \\ \hline = \text{ MONEY} \\ \\ 9567 \\ + 1085 \\ \hline = 10652 \end{array}$$

# Best Solution Search

38

- Naïve approach infeasible
  - compute all solutions
  - choose best
  
- Branch-and-bound approach
  - compute first solution
  - add “betterness” constraint to open nodes
  - next solution will be “better”
  - prunes search space

# Summary

39

## □ Modeling

- variables with domain
- constraints to state relations
- branching strategy

## □ Solving

- constraint propagation
- constraint branching
- search tree exploration

40

# Why Does CP Matter?



# Widely Applicable

41

- Timetabling
- Scheduling
- Crew rostering
- Resource allocation
- Workflow planning and optimization
- Gate allocation at airports
- Sports-event scheduling
- Railroad: track allocation, train allocation, schedules
- Automatic composition of music
- Genome sequencing
- Frequency allocation
- ...

# Draws on Variety of Techniques

42

- Artificial intelligence
  - basic idea, search, ...
- Operations research
  - scheduling, flow, ...
- Algorithms
  - graphs, matchings, networks, ...
- Programming languages
  - programmability, extensionability, ...

# Essential

43

## Compositional middleware for combining

- smart algorithmic (solving)
- problem substructures (modeling)

## components (propagators)

- scheduling, graphs, flows, ...

## while supporting

- essential extra constraints
- to be explored in the following excursions

44

# Capturing Structure

distinct (alldifferent) reconsidered

# Distinct Propagator

45

- Infeasible: no dedicated propagator
  - decompose  $\text{distinct}(x_1, \dots, x_n)$
  - into  $x_i \neq x_j$  ( $1 \leq i < j \leq n$ ) disequality propagators
  - too many propagators  $O(n^2)$ , propagation too weak
  
- Not much better: naive distinct propagator
  - wait until variable becomes assigned
  - remove value from all other variables
  - propagation too weak

# Naïve Is Not Good Enough

46

- $\text{distinct}(x, y, z)$ 
  - decomposition:  $x \neq y$  and  $x \neq z$  and  $y \neq z$
  
- $x \in \{1, 2, 3\}, y \in \{1, 2\}, z \in \{1, 2\}$ 
  - should propagate  $x \in \{1\}$
  
- $x \in \{1, 2\}, y \in \{1, 2\}, z \in \{1, 2\}$ 
  - should exhibit failure without search

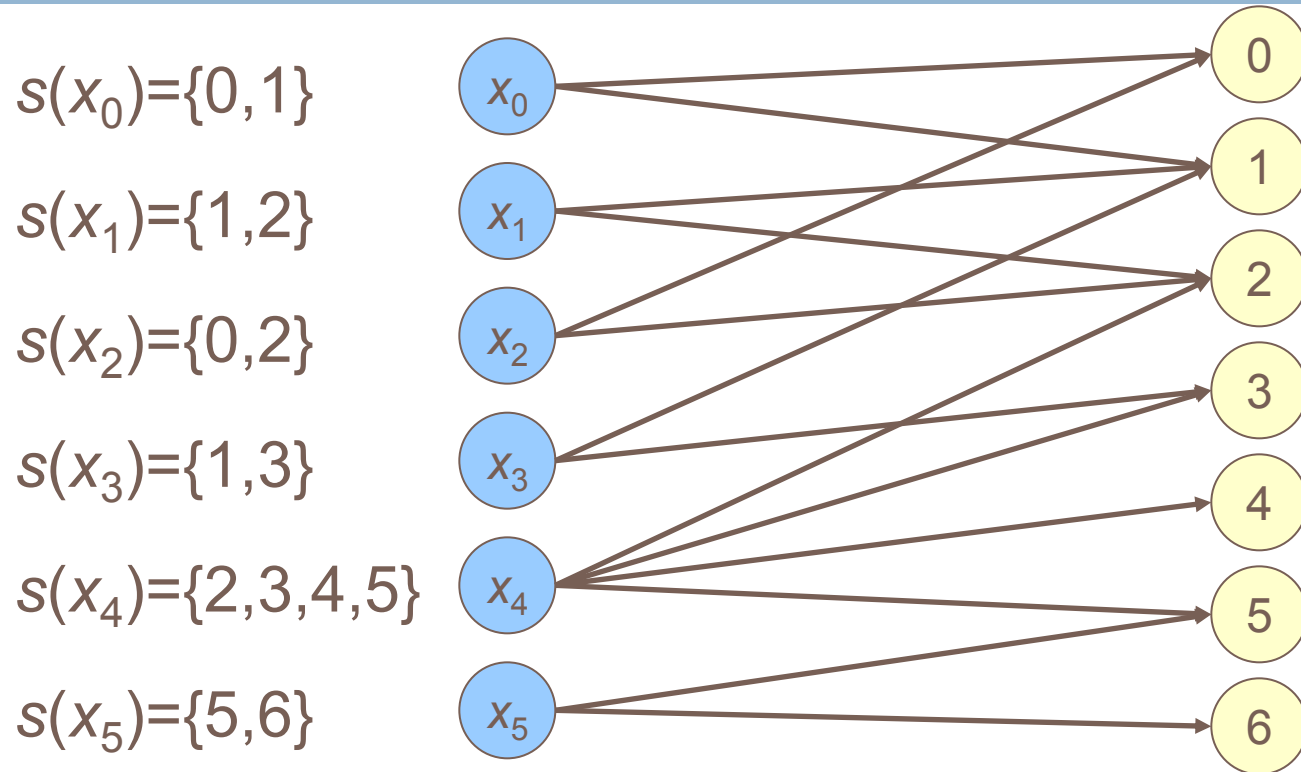
# Strong Distinct Propagator

47

- Strong - global - distinct propagator
  - only keep values appearing in a solution to constraint
  - essential for many problems (permutation problems)
  - takes global perspective on constraint
  - is strongest: domain-consistent, hyper-arc consistent, ...
  
- Can be propagated efficiently
  - $O(n^{2.5})$  is efficient [Régin, 1994]
  
- Uses graph algorithms
  - solutions of constraint  $\Leftrightarrow$  properties of graph
  - characterize all solutions: prune excess values

# Variable Value Graph

48



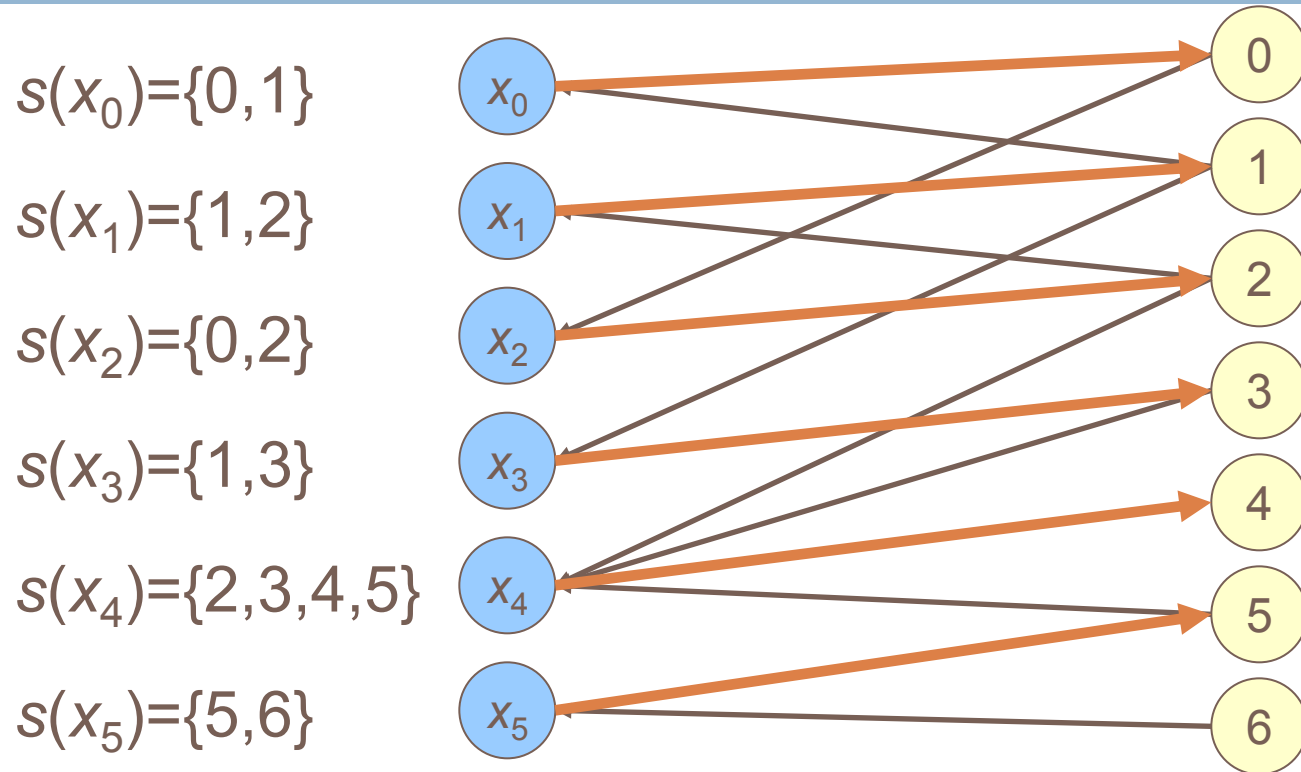
## □ Bipartite graph

- variable nodes → value nodes



# Solution: Maximal Matching

49



□ Compute single maximal matching

■ **matched edge**

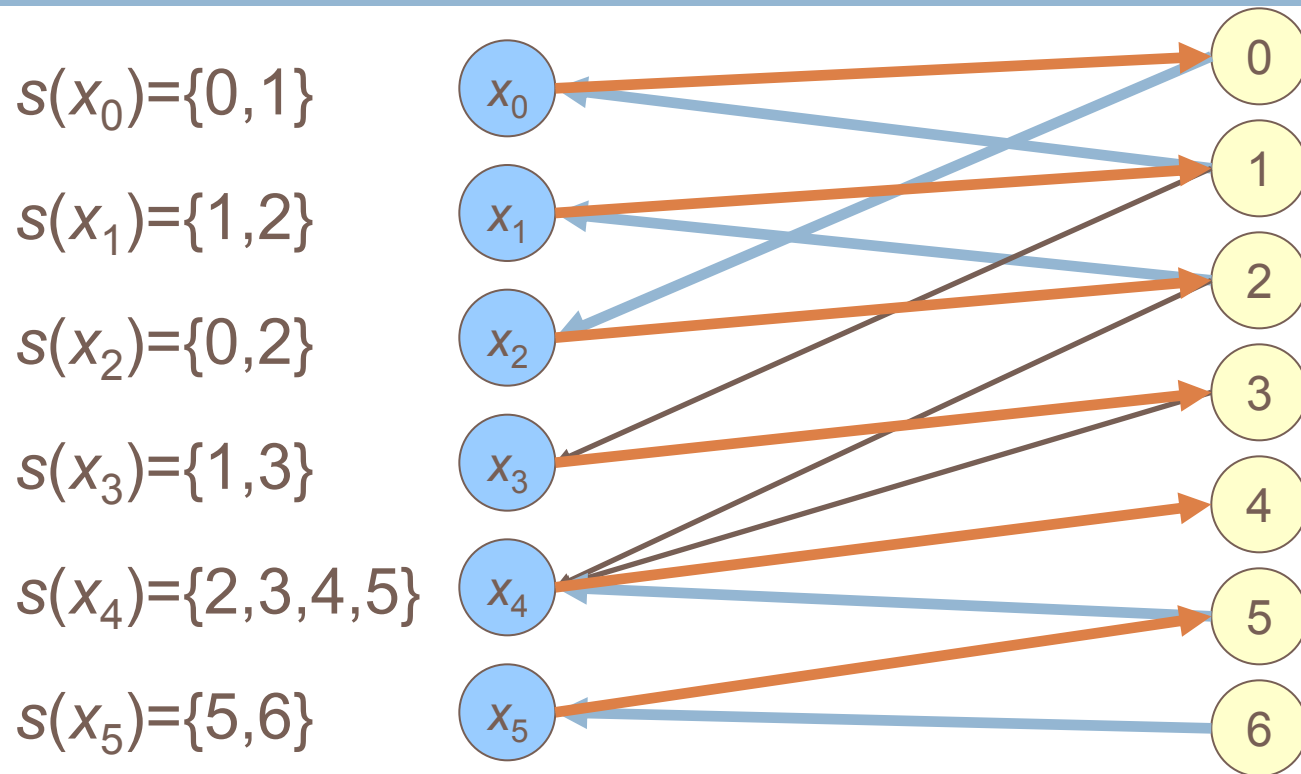
■ **free edge**

variable node  $\rightarrow$  value node

value node  $\rightarrow$  variable node

# Characterize All Solutions

50

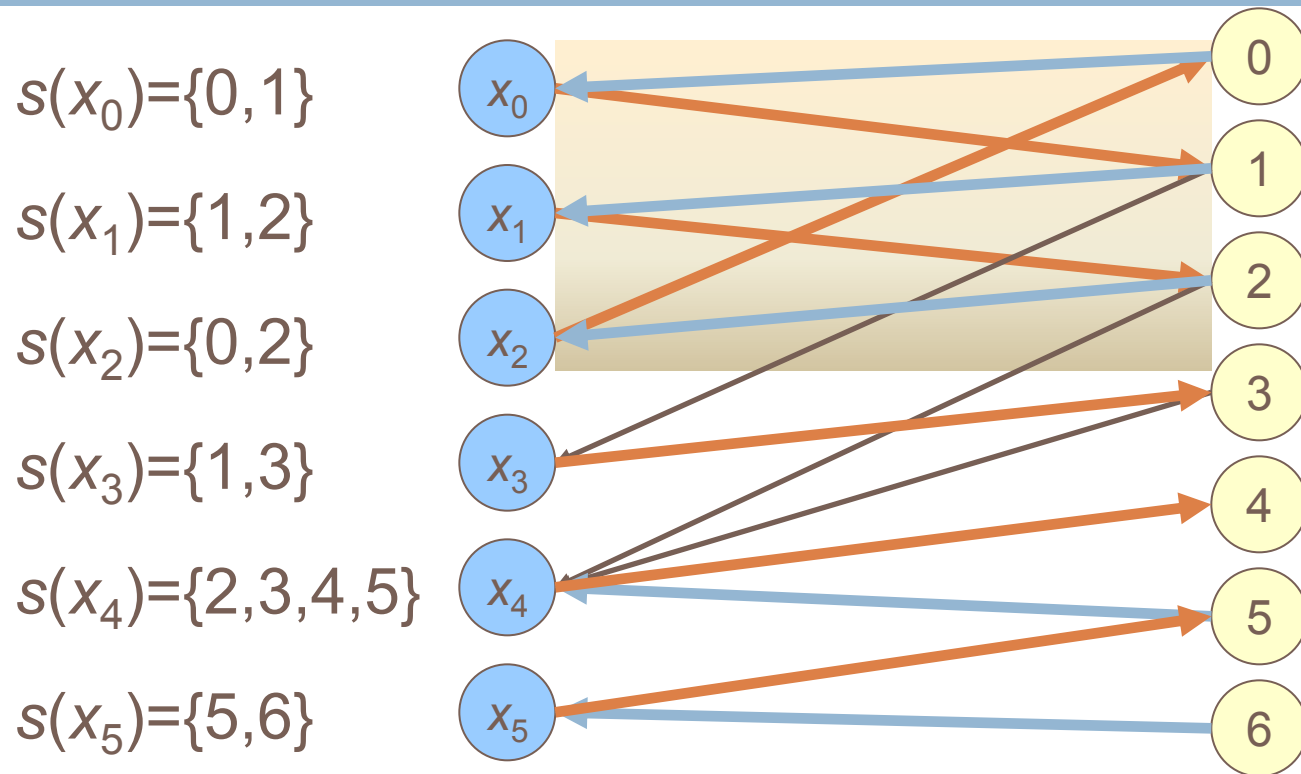


## □ Edges that can appear in any matching

- even alternating cycles  $(x_0 \rightarrow 0 \rightarrow x_2 \rightarrow 2 \rightarrow x_1 \rightarrow 1 \rightarrow x_0)$
- even alternating paths  $(6 \rightarrow x_5 \rightarrow 5 \rightarrow x_4 \rightarrow 4)$

# Characterize All Solutions

51

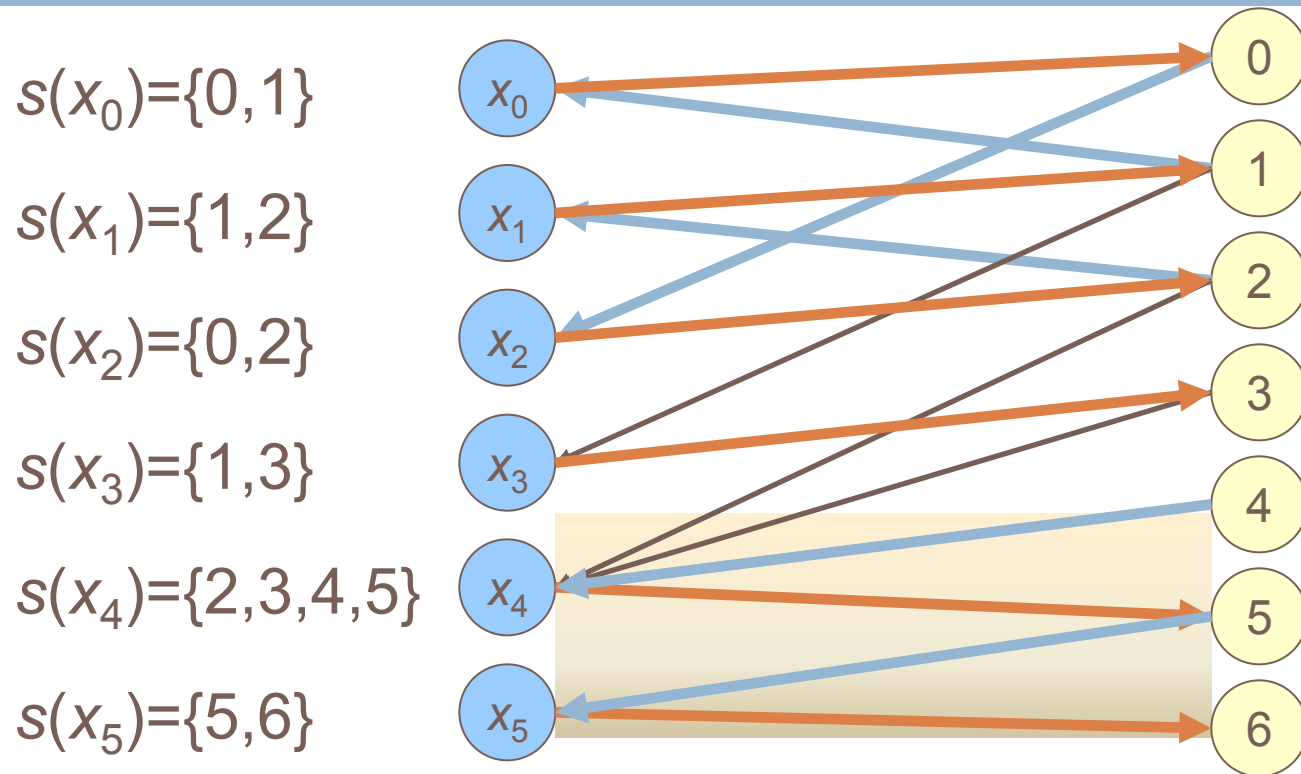


## □ Edges that can appear in any matching

- even alternating cycles  $(x_0 \rightarrow 1 \rightarrow x_1 \rightarrow 2 \rightarrow x_2 \rightarrow 0 \rightarrow x_0)$
- even alternating paths  $(6 \rightarrow x_5 \rightarrow 5 \rightarrow x_4 \rightarrow 4)$

# Characterize All Solutions

52

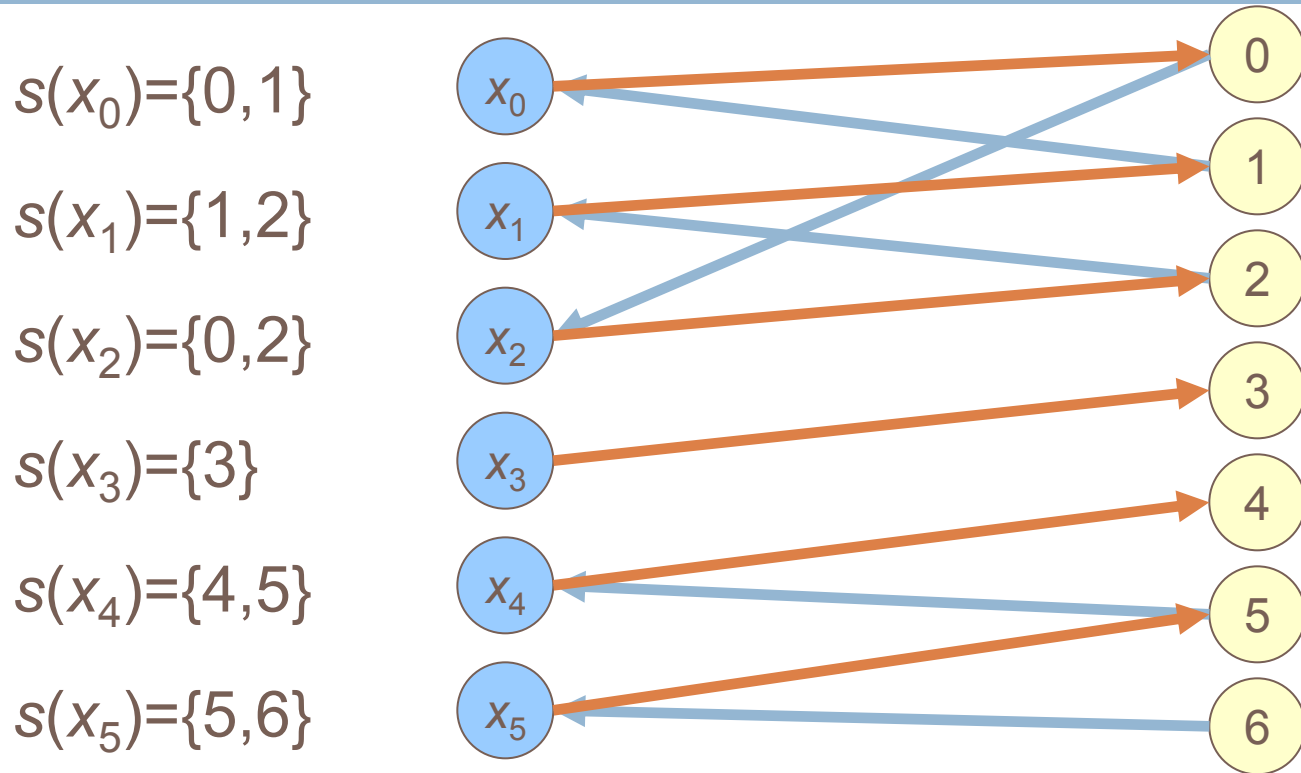


## □ Edges that can appear in any matching

- even alternating cycles  $(x_0 \rightarrow 0 \rightarrow x_2 \rightarrow 2 \rightarrow x_1 \rightarrow 1 \rightarrow x_0)$
- even alternating paths  $(4 \rightarrow x_4 \rightarrow 5 \rightarrow x_5 \rightarrow 6)$

# Prune Edges (Values)

53



- Prune edges that cannot appear in any matching
  - accordingly: prune values from variables

# Global Constraints

54

- Reasons for globality: decomposition...
  - semantic: ...not possible
  - operational: ...less propagation
  - algorithmic: ...less efficiency
  
- Plethora available
  - scheduling, sequencing, cardinality, sorting, circuit, ...
  - systematic catalogue with hundreds available  
<http://www.emn.fr/x-info/sdemasse/gccat/>
  - sometimes not straightforward to pick the right one (strength versus efficiency, etc)

# Summary

55

- Constraints capture problem structure
    - **ease modeling** (commonly recurring structures)
    - **enable solving** (efficient algorithms available)
  
  - Constraints as
    - **reusable**
    - **powerful**
- software components

# How to Deal with Distinct...

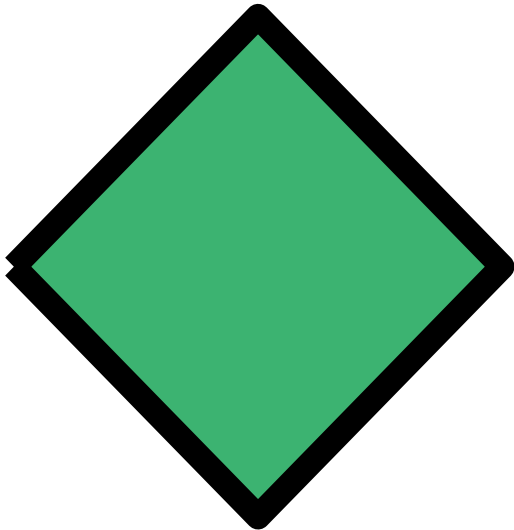
56

- Assume  $n$  variables, at most  $d$  values
- SAT (propositional formulae)
  - $O(nd)$  clauses [Gent, Nightingale, 2004]
  - other encodings possible
  
- MILP (mixed integer linear programs)
  - introduce  $O(nd)$  new 0/1 variables
  - decompose into  $O(n+d)$  linear (in)equations  
[Hooker, 2007, p 368]



# SMM: Strong Propagation

57



$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline = \text{MONEY} \\ \\ \text{9567} \\ + \text{1085} \\ \hline = \text{10652} \end{array}$$

58

# Local Reasoning

beauty and curse of  
constraint programming

# Kakuro

59

		11	4		
	5 14			10	
17					3
6			4 3		
	10				
		3			

# Kakuro

60

		11	4		
	5 14			10	
17					3
6			4 3		
	10				
		3			

- Fields take digits
- Hints describe
  - for row or column
  - digit sum must be hint
  - digits must be distinct

# Kakuro

61

		11	4		
	5 14			10	
17					3
6			4 3		1
	10				2
		3			

- For hint 3  
1 + 2

# Kakuro

62

		11	4		
	5 14			10	
17					3
6			4 3		2
	10				1
		3			

□ For hint 3

$$1 + 2$$

or

$$2 + 1$$

# Kakuro

63

		11	4		
	5 14			10	
17					3
6			4 3	1	3
	10				
		3			

- For hint 4  
 $1 + 3$

# Kakuro

64

		11	4		
	5 14			10	
17					3
6			4 3	3	1
	10				
		3			

□ For hint 4

$$1 + 3$$

or

$$3 + 1$$



# Kakuro

65

		11	4		
	5 14			10	
17					3
6			4 3	3	1
	10				2
		3			

□ For hint 3

$$1 + 2$$

□ For hint 4

$$1 + 3$$

# Kakuro Solution

66

		11	4		
	<sup>5</sup> 14	<b>2</b>	<b>3</b>	10	
17	<b>9</b>	<b>5</b>	<b>1</b>	<b>2</b>	<sub>3</sub>
<sub>6</sub>	<b>5</b>	<b>1</b>	<sup>4</sup> <sub>3</sub>	<b>3</b>	<b>1</b>
	<sub>10</sub>	<b>3</b>	<b>1</b>	<b>4</b>	<b>2</b>
		<sub>3</sub>	<b>2</b>	<b>1</b>	

# Modeling and Solving Kakuro

67

- Obvious model: for each hint
  - distinct constraint
  - sum constraint
- Good case... (?)
  - few variables per hint
  - few values per variable
- Let's try it...
  - $22 \times 14$ , 114 hints: 9638 search nodes, 2min 40sec
  - $90 \times 124$ , 4558 hints: ? search nodes, ? years  
years? centuries? eons?

# Failing for Kakuro...

68

- Beauty of constraint programming
  - local reasoning
  - propagators are independent
  - variables as simple communication channels
  
- Curse of constraint programming
  - local reasoning
  - propagators are independent
  - variables as simple communication channels

69

# User-defined Constraints

workforce rostering

Kakuro reconsidered

# Modeling Rostering: User-defined

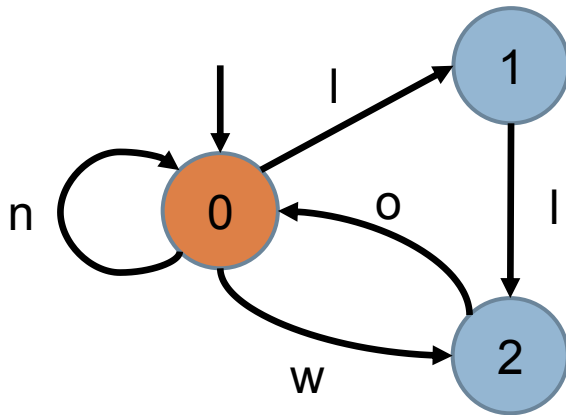
70

- Personnel rostering: example
  - one day off (o) after weekend shift (w)
  - one day off (o) after two consecutive long shifts (l)
  - normal shifts (n)
- Infeasible to implement propagator for ever-changing rostering constraints
- User-defined constraints: describe legal rosters by regular expression
  - $(wo \mid llo \mid n)^*$

# Regular Constraint

71

$(wo \mid llo \mid n)^*$



$\text{regular}(x_1, \dots, x_n, r)$

- $x_1 \dots x_n$  word in  $r$
- or, accepted by DFA  $d$  for  $r$

- Propagation idea: maintain all accepting paths
  - from start state (0) to a final state (0): solutions!
  - symbols on transitions comply with variable values

# Propagating Regular

72

$x$

$y$

$z$

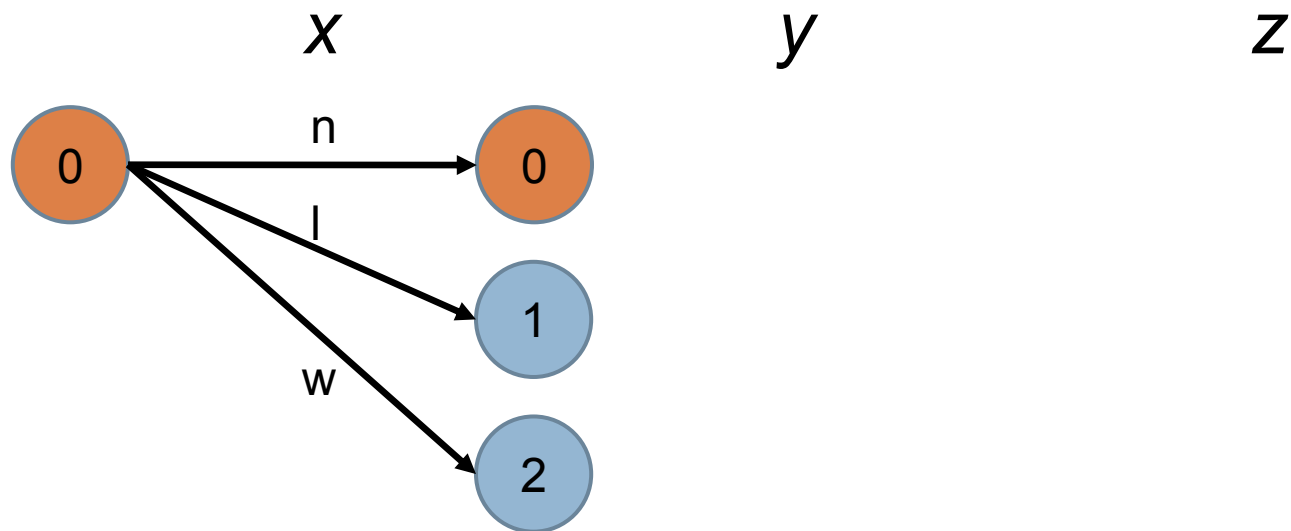
0

- Example:  $\text{regular}(x, y, z, d)$ 
  - $x, y, z$  in  $\{w, o, l, n\}$
  - in reality:  $w=0, o=1, l=2, n=3$



# Propagating Regular

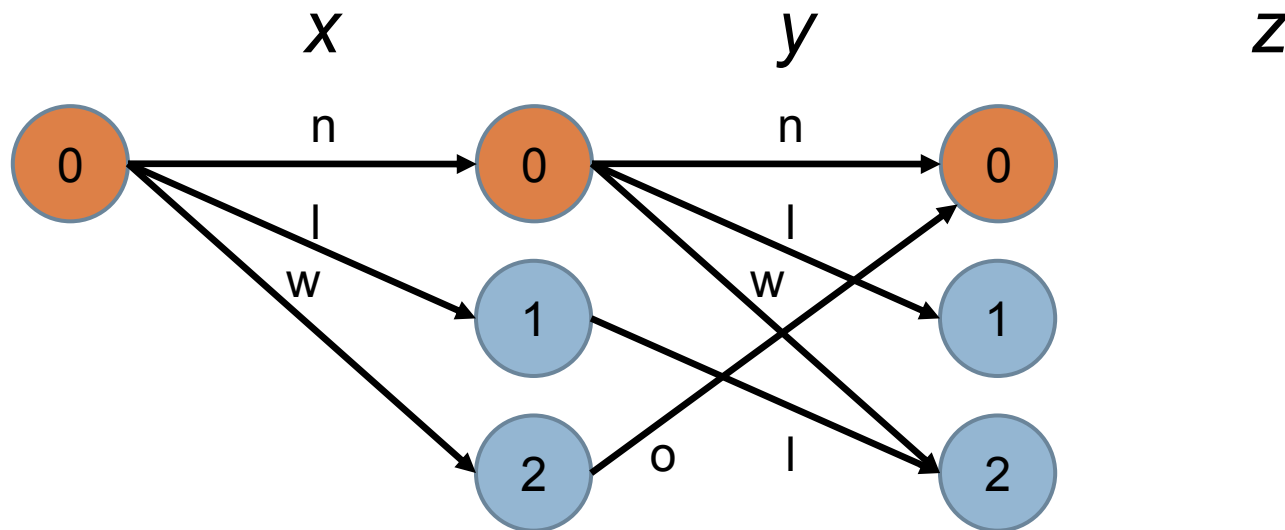
73



- Forward pass
  - all paths from start state

# Propagating Regular

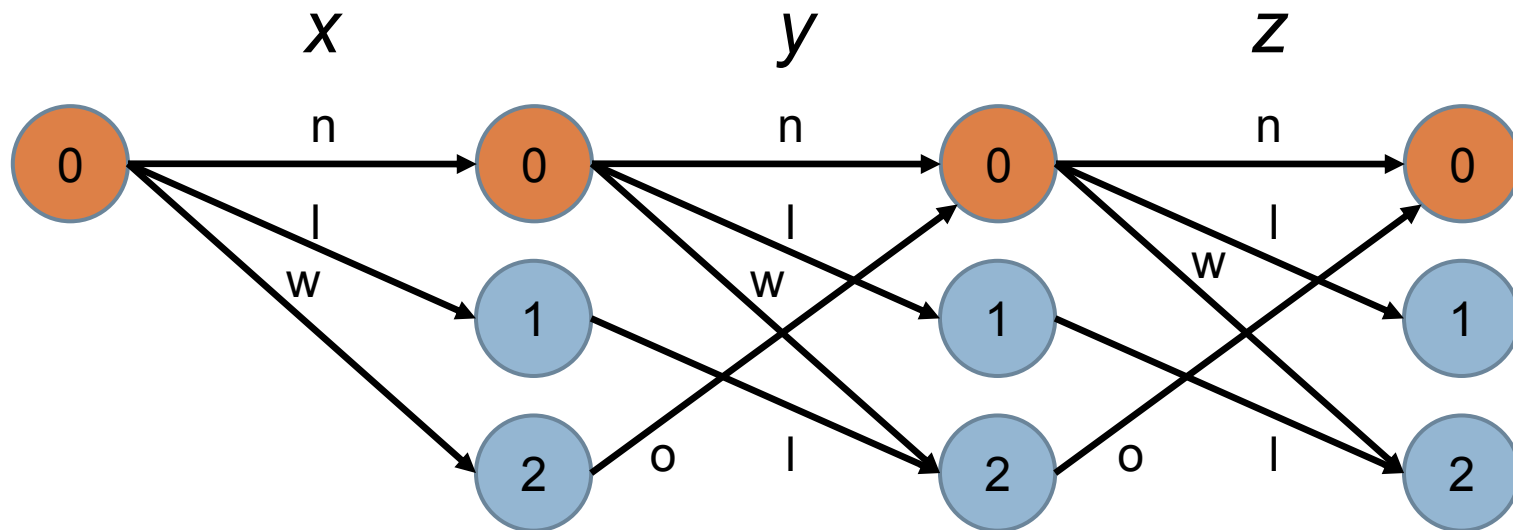
74



- Forward pass: optimization
  - each state at most once for each variable (“layer”)
  - several incoming/outgoing edges per state

# Propagating Regular

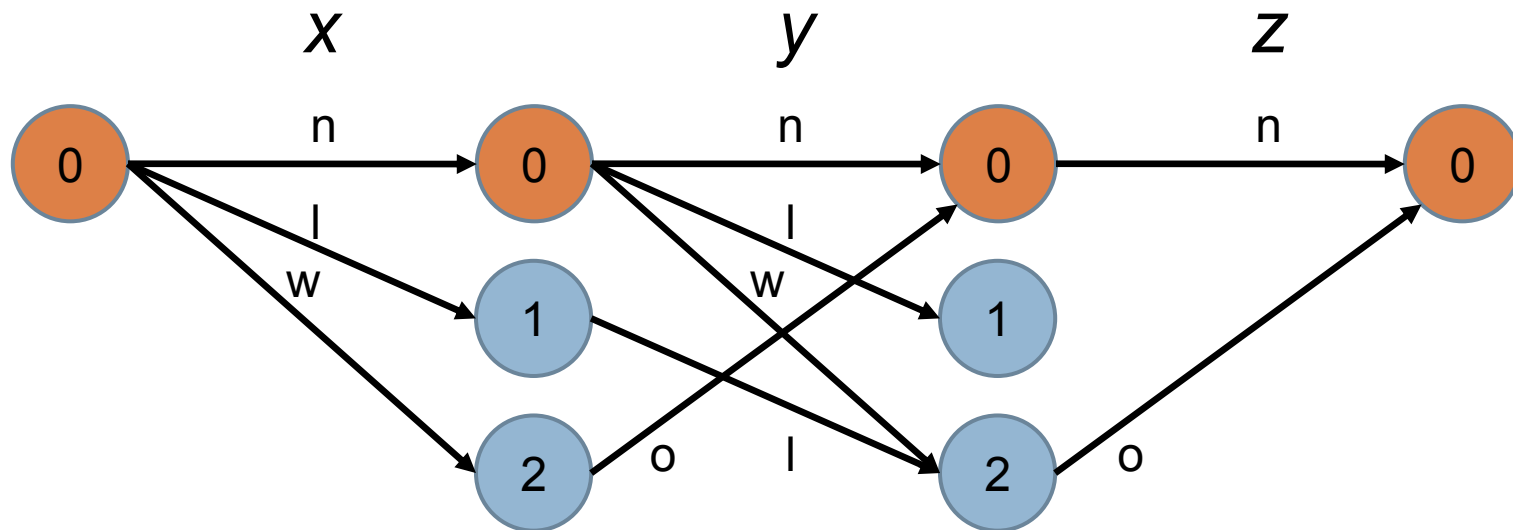
75



- Forward pass finished

# Propagating Regular

76

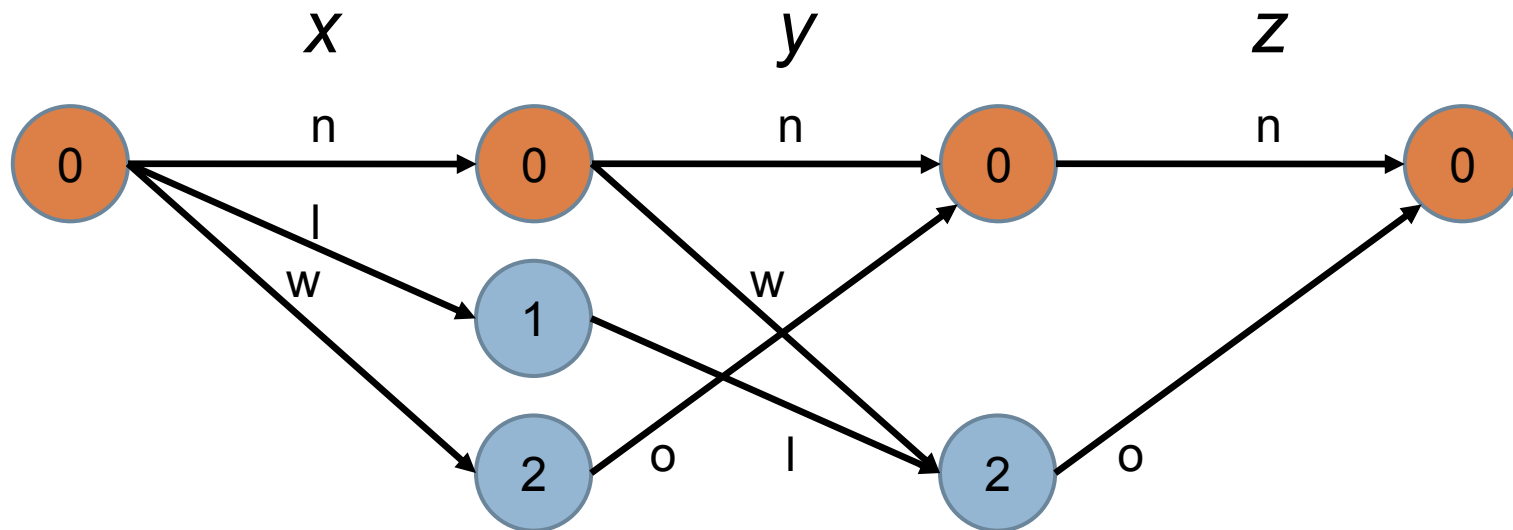


## □ Backward pass

- start: remove non-final states for last layer

# Propagating Regular

77

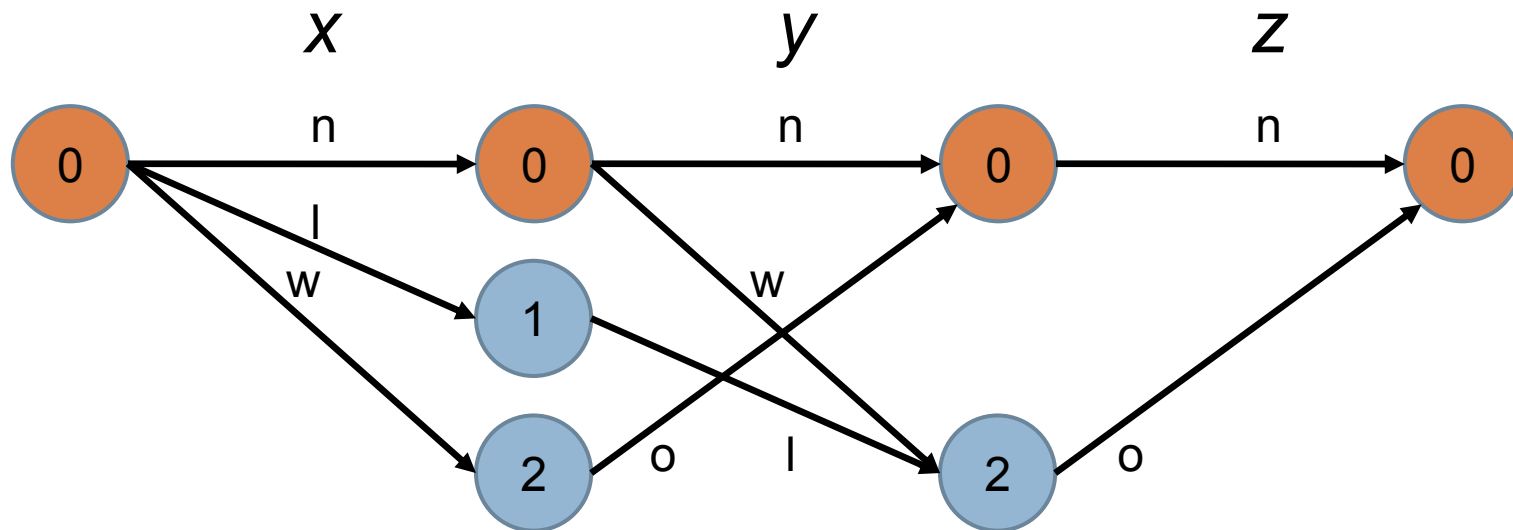


## □ Backward pass

- start: remove non-final states for last layer
- continue: remove states with no outgoing edges

# Propagating Regular

78



## □ Pruning

$$x \in \{n, l, w\} \quad y \in \{n, l, w, o\} \quad z \in \{n, o\}$$

# Getting Even Better

79

- Variants of regular constraint
  - original regular constraint [Pesant, 2004]
  - use way more efficient MDD instead of DFA [Yap ea, 2008]
  - cost-based variants available [Pesant, ea, 2007]

# AI's Legacy

80

- Original model for constraint propagation
  - constraints used for propagation in extension (list of solutions): no propagators
  - single algorithm for all constraints (consistency)
  - often restricted to binary constraints
- Beautiful model
  - insightful for understanding propagation
  - rich connections (complexity, relational databases, ...)
  - rich notion of levels of pruning: arc consistency, path consistency,  $k$ -consistency, ...



# AI's Legacy: Solving for Real?

81

- Constraints used for propagation in extension
  - unable to exploit structure for efficient solving
  - unrealistic for large arity: distinct with  $n$  variables has  $n!$  solutions, ....
- Single algorithm for all constraints
  - infeasible in general: constraints may be NP-hard
  - no compromise between pruning and efficiency
- Often restricted to binary constraints
  - decomposition destroys propagation

# The Best of Both Worlds

82

- Start from propagator-based constraint propagation
  - take advantage of dedicated algorithms
  
- Dedicated propagator for user-defined constraints
  - only pay, if needed
  - incredibly efficient: MDD-based propagator [Yap ea, 2008]

# Kakuro Reconsidered

83

- Real model: for each hint
  - one regular constraint combining distinct and sum
  - precompute when model is setup
- Good case...
  - few solutions for combined constraint
- Let's try again (precomputation time included)
  - $22 \times 14$ , 114 hints: 0 search nodes, 28 msec
  - $90 \times 124$ , 4558 hints: 0 search nodes, 345 msec

# Summary

84

- User-defined constraints
  - high degree of flexibility
  - efficient and perfect propagation
  - limited to medium-sized constraints
  - use specialized propagator rather than extensional framework
  
- Kakuro: decomposition is harmful [again]
  - capture essential structure by few constraints
  - best by single constraint

85

# Compositional Modeling

scheduling resources

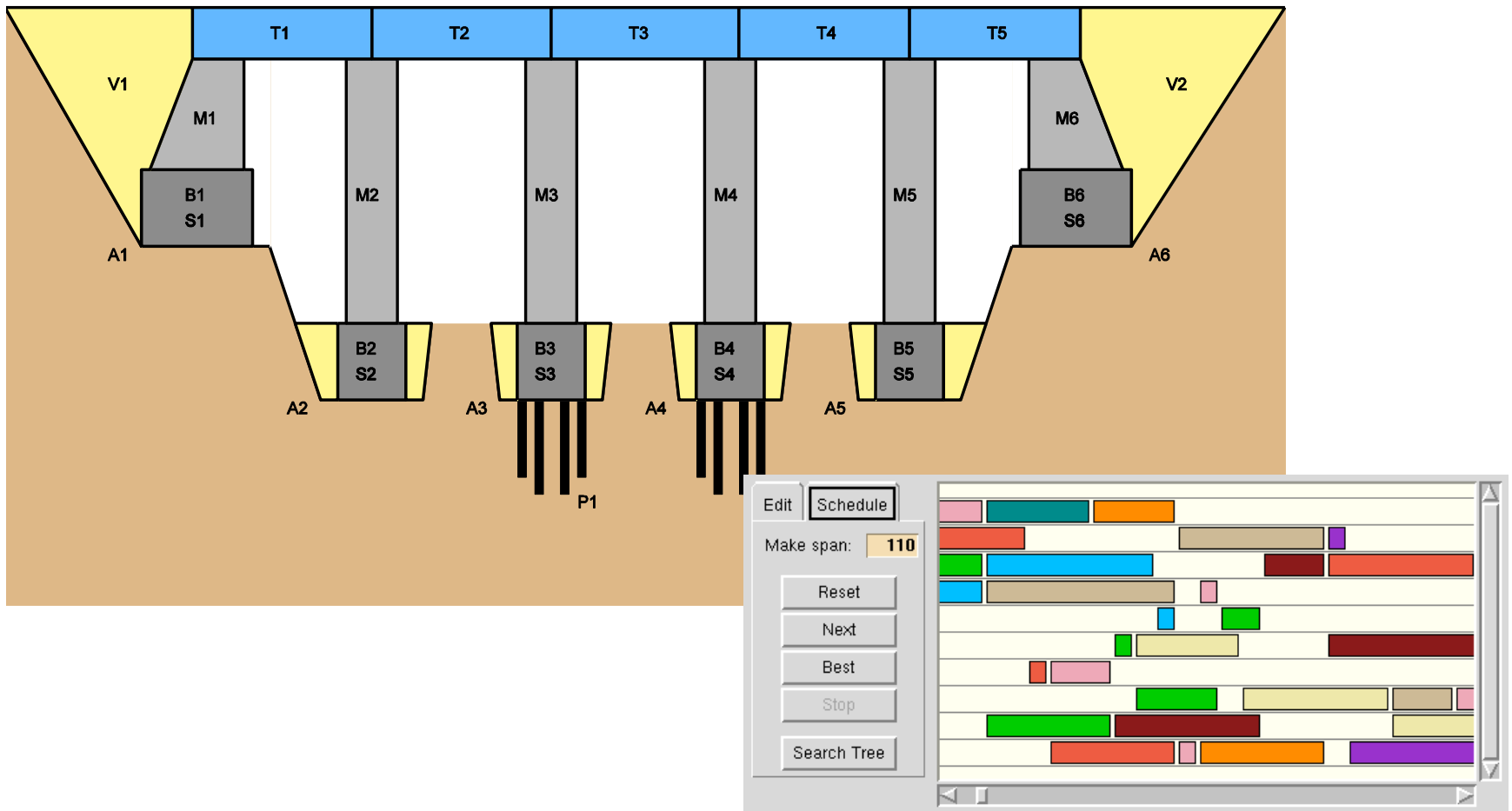
# Scheduling Resources: Problem

86

- Tasks
  - duration
  - resource
  
- Precedence constraints
  - determine order among two tasks
  
- Resource constraints
  - at most one task per resource  
[disjunctive, non-preemptive scheduling]

# Scheduling: Bridge Example

87



# Scheduling: Solution

88

- Start time for each task
- All constraints satisfied
- Earliest completion time
  - minimal make-span



# Scheduling: Model

89

- Variable for start-time of task  $a$   
 $\text{start}(a)$
- Precedence constraint:  $a$  before  $b$   
 $\text{start}(a) + \text{dur}(a) \leq \text{start}(b)$

# Scheduling: Model

90

- Variable for start-time of task  $a$   
 $\text{start}(a)$
- Precedence constraint:  $a$  before  $b$   
 $\text{start}(a) + \text{dur}(a) \leq \text{start}(b)$
- Resource constraint:  
 $a$  before  $b$   
or  
 $b$  before  $a$

# Scheduling: Model

91

- Variable for start-time of task  $a$   
 $\text{start}(a)$
- Precedence constraint:  $a$  before  $b$   
 $\text{start}(a) + \text{dur}(a) \leq \text{start}(b)$
- Resource constraint:  
 $\text{start}(a) + \text{dur}(a) \leq \text{start}(b)$   
or  
 $b$  before  $a$

# Scheduling: Model

92

- Variable for start-time of task  $a$   
 $start(a)$
- Precedence constraint:  $a$  before  $b$   
 $start(a) + dur(a) \leq start(b)$
- Resource constraint:

$$start(a) + dur(a) \leq start(b)$$

or

$$start(b) + dur(b) \leq start(a)$$

[use so-called reification for this]

# Model: Easy But Too Naive

93

- Local view
  - individual task pairs
  - $O(n^2)$  propagators for  $n$  tasks
  
- Global view (again a global constraint)
  - all tasks on resource
  - single propagator
  - smarter algorithms possible

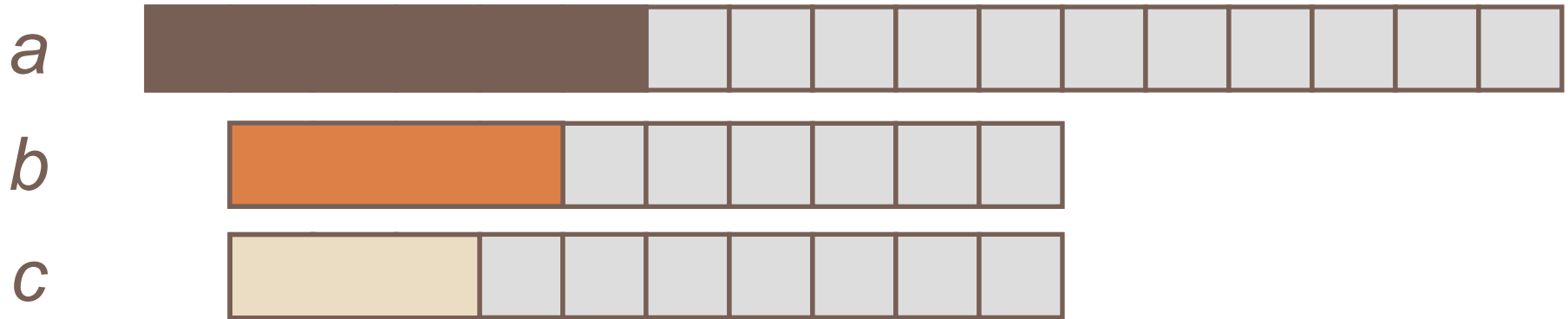
# Edge Finding: Idea

94

- Assume a subset  $O$  of tasks and a task  $t \in O$ 
  - compute earliest completion time of  $O$   
 $ect(O)$
  - compute latest completion time of  $O - \{t\}$   
 $lct(O - \{t\})$
  - if  
 $ect(O) > lct(O - \{t\})$   
then  
 $t$  must run last in  $O$
- Can be done in  $O(n \log n)$  for  $n$  tasks  
[Carlier & Pinson, 1994] [Vilím ea., 2004]

# Edge Finding

95

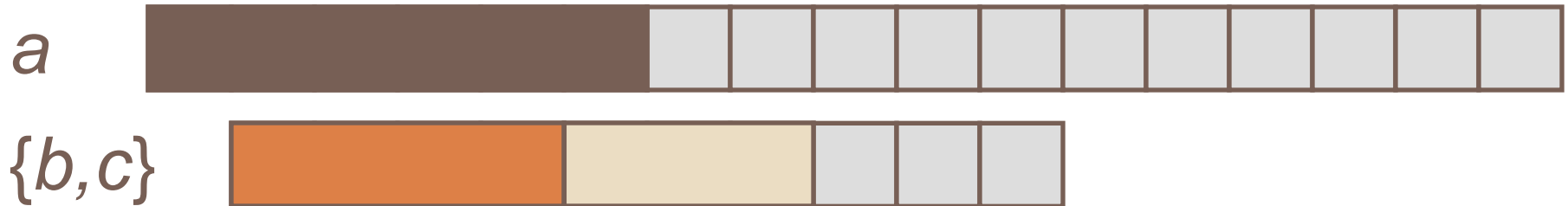


## □ Assume

- $\text{start}(a) \in \{0, \dots, 11\}$      $\text{dur}(a) = 6$
- $\text{start}(b) \in \{1, \dots, 7\}$      $\text{dur}(b) = 4$
- $\text{start}(c) \in \{1, \dots, 8\}$      $\text{dur}(c) = 3$

# Edge Finding

96

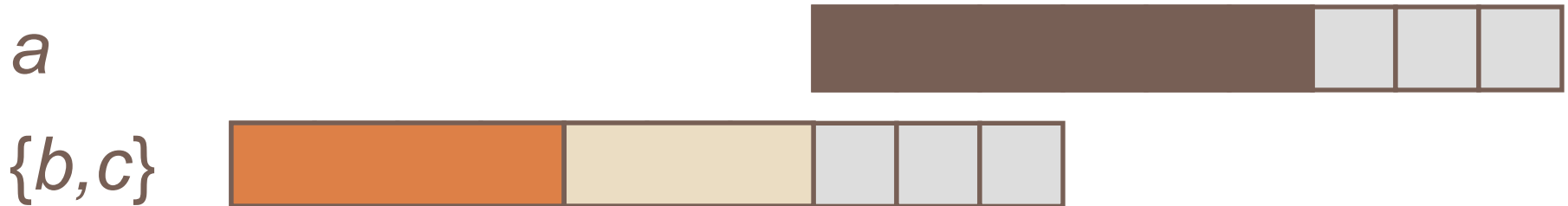


- Assume  $O=\{a,b,c\}$ ,  $t=a$
- Clearly,  $a$  must go last



# Edge Finding

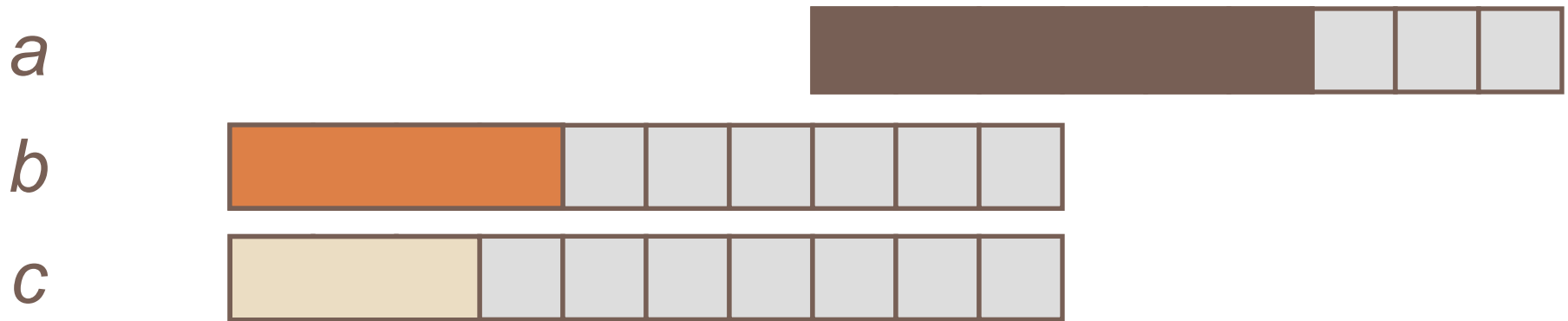
97



- Assume  $O=\{a,b,c\}$ ,  $t=a$
- Clearly,  $a$  must go last

# Edge Finding

98



## □ Propagate

- $\text{start}(a) \in \{8, \dots, 11\}$

# Constraint-based Scheduling

99

## □ Rich set of methods

- propagation
- branching heuristics
- search methods

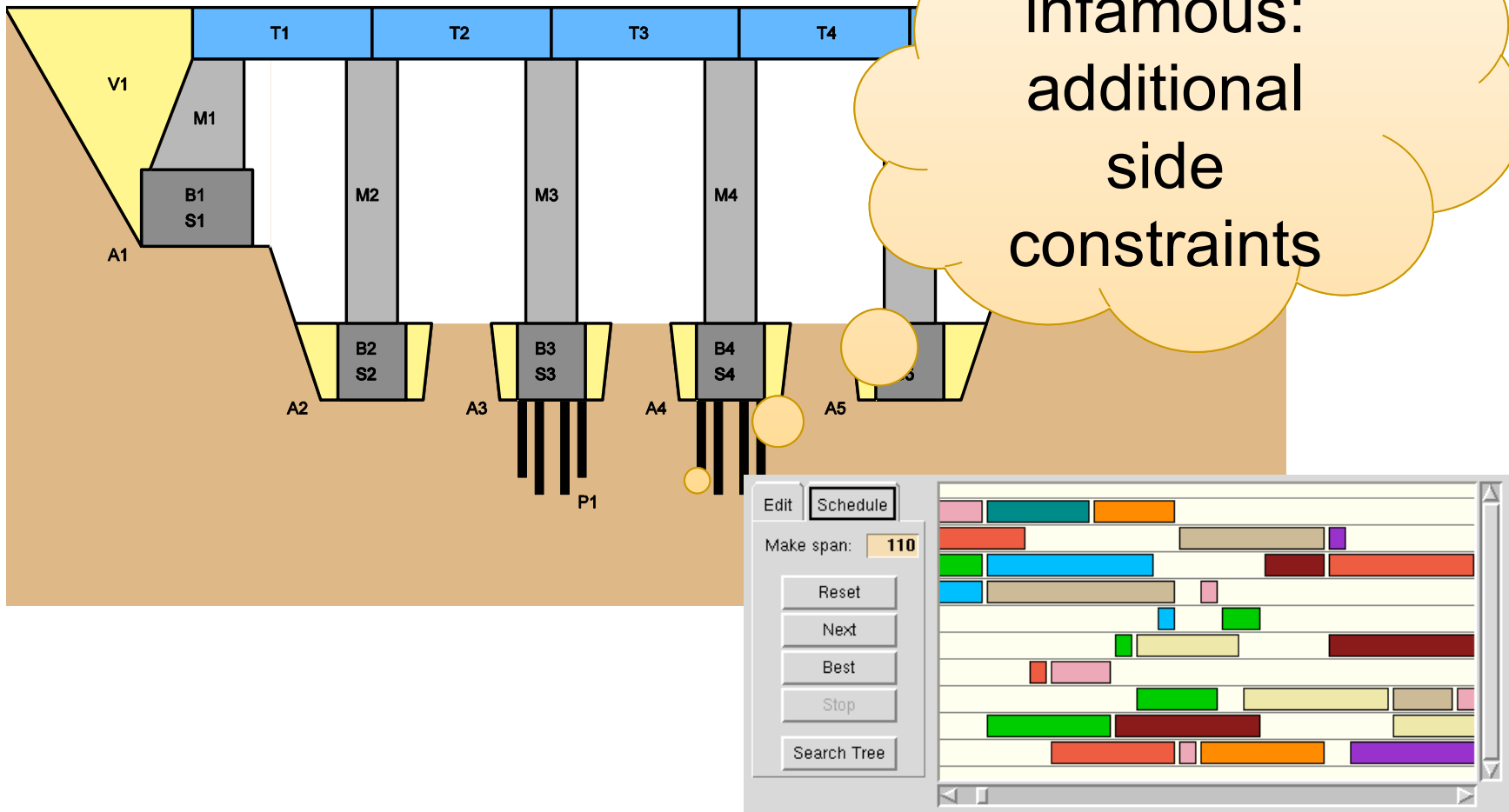
## □ Many variants

- disjunctive, cumulative, elastic, preemptive, ...
- batch processing, setup times, ...

[Baptiste, Le Pape, Nuijten, Constraint-based Scheduling. Kluwer, 2001]

# Scheduling: Bridge Example

100



# Summary

101

- Modeling is compositional
  - reasoning is too!
  
- Powerful global constraints...
  - plus...
    - essential additional side constraints
  
- Scheduling domain
  - show case of constraint programming

102

# Strength And Challenges

# Strength

103

- Captures structure
  - use structure for efficient reasoning
  - unique distinction from SAT and LP
- Flexible, compositional, reusable
  - add additional side constraints
  - add new algorithmic components
  - high return on investment into global constraints
- Simple
  - clear model based on propagators
- Efficient systems available
  - commercial and open source

# Challenges

104

- Modeling: art not science
  - true to some extent for most approaches
  - here: identify substructures, know strength of different methods
  - array of techniques: symmetry breaking, implied constraints, heuristics, ...
- Search: mostly naive
  - local decision making
  - no global techniques such as learning (SAT), or strong branching, impact-based search (LP)
  - remedies in their infancy



# The Essence

105

- Constraint programming is about...
  - ...local reasoning exploiting structure
- Strength
  - simplicity, compositionality, exploiting structure
- Challenges
  - lack of global picture during search
  - difficult to find global picture due to rich structure
- Future
  - part of hybrid solutions

# Resources

106

- Complete and recent overview
  - Rossi, Van Beek, Walsh, eds. Handbook of Constraint Programming, Elsevier, 2006 (around 950 pages).
  
- National perspective
  - Flener, Carlsson, Schulte. Constraint Programming in Sweden, *IEEE Intelligent Systems*, pages 87-89. IEEE Press, March/April, 2009.