Tutorial on Constraint Programming and Scheduling

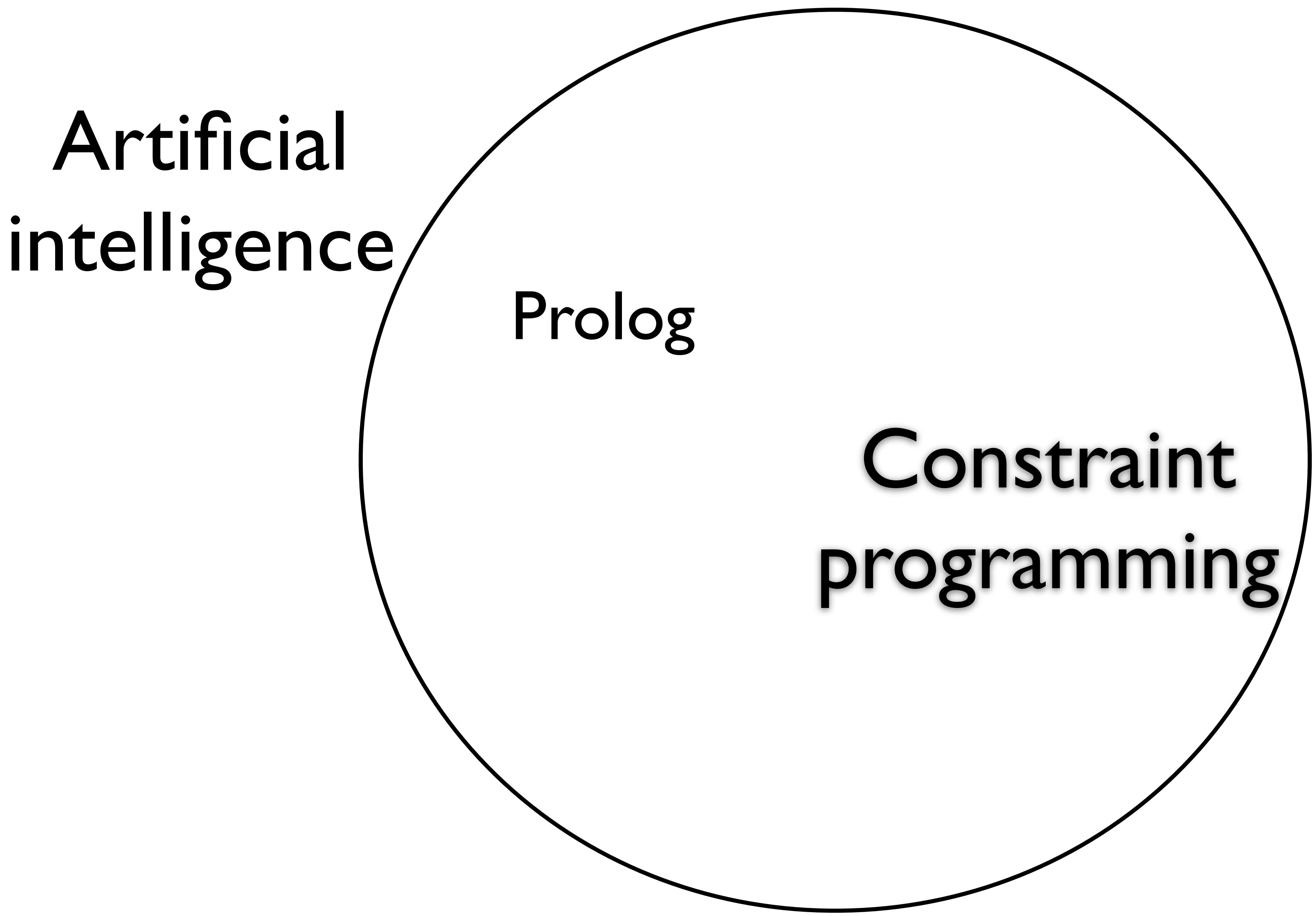
Claude-Guy Quimper

UNIVERSITÉ LAVAL

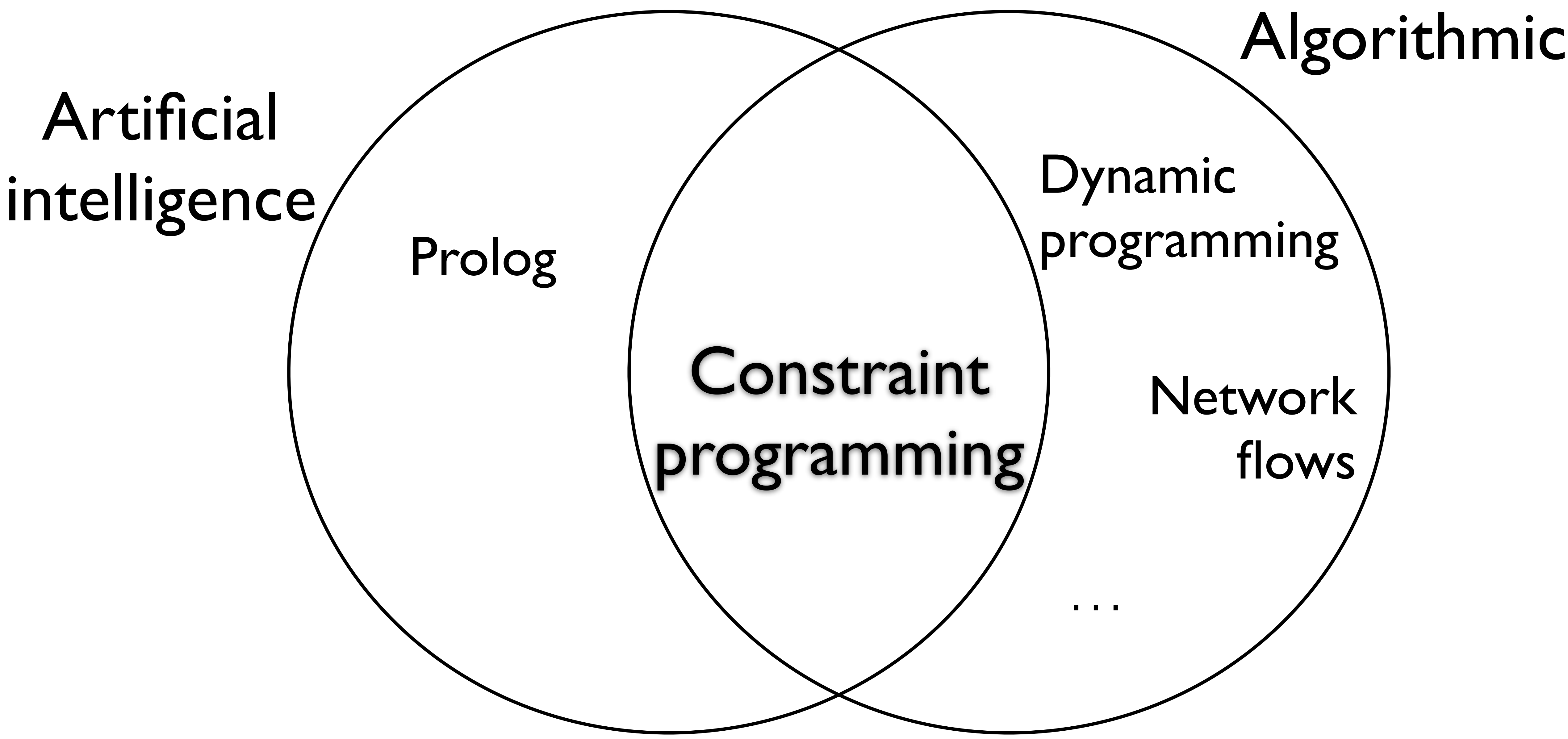
Overview of this tutorial

- What is constraint programming?
- How does it work?
- How can one make it work better?
- How is constraint programming adapted to scheduling problems?
- Presentation of tools

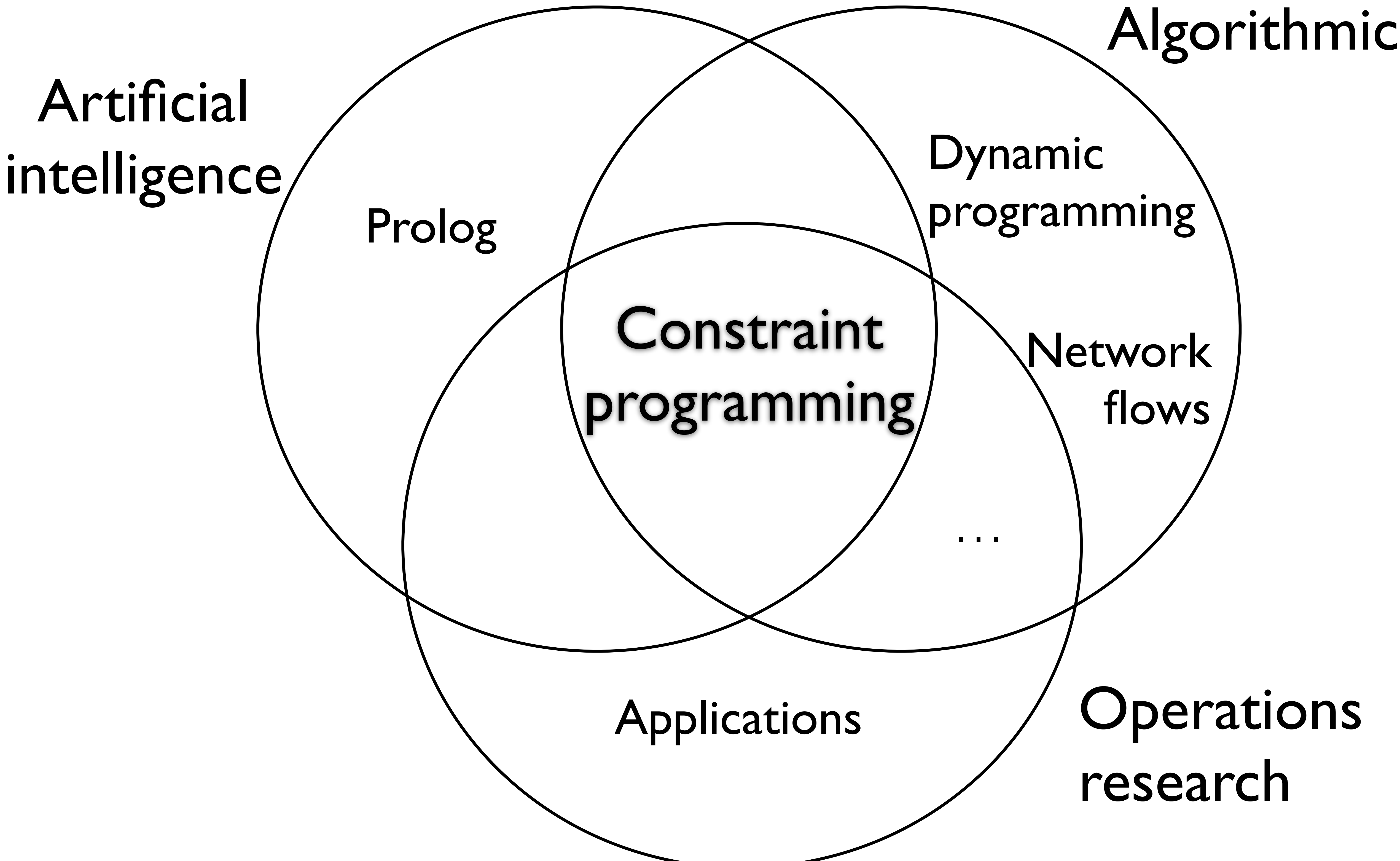
The origins



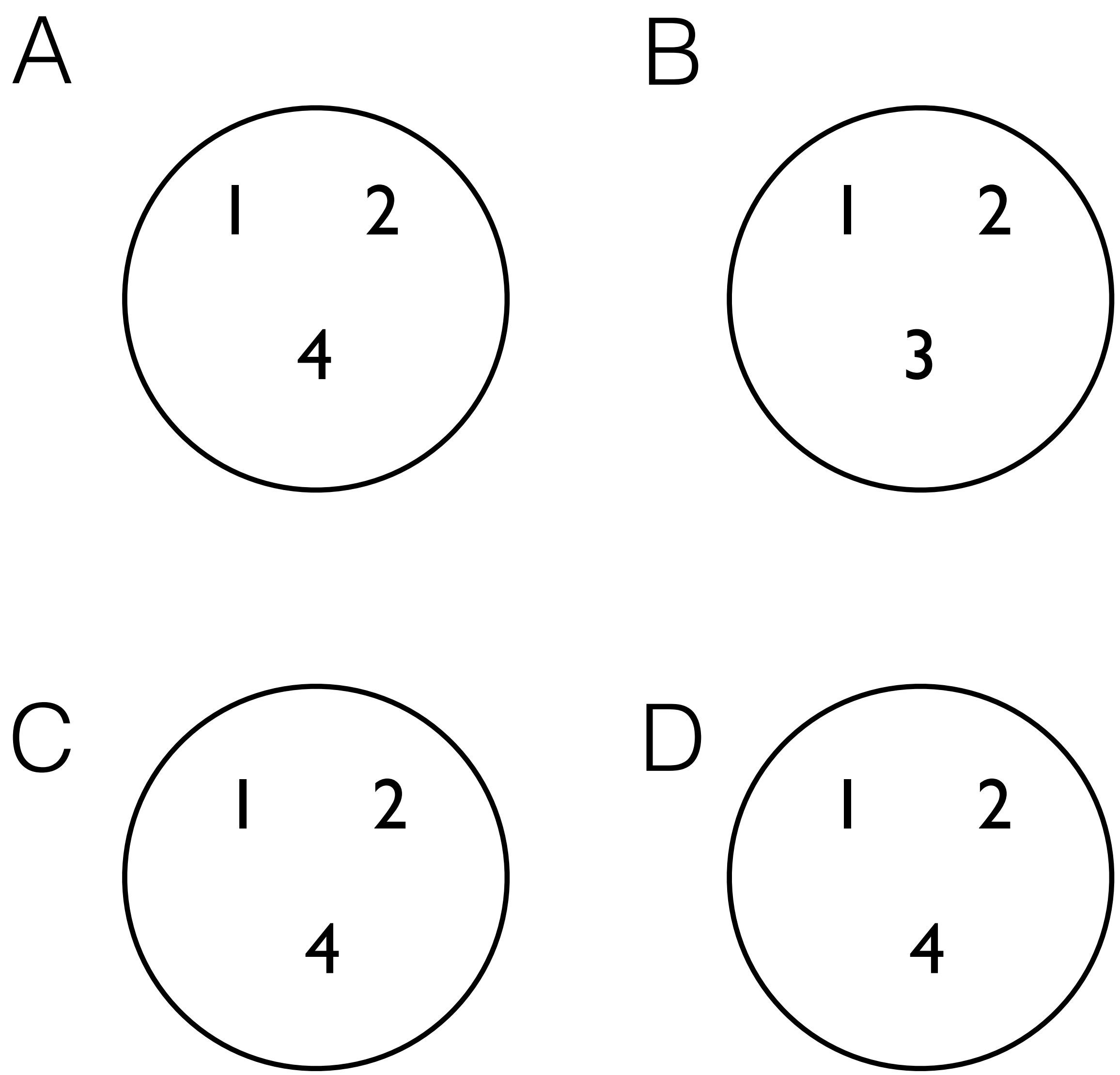
The origins


Artificial intelligence
Algorithmic
Prolog
Constraint programming
Dynamic programming
Network flows
. . .

The origins


Algorithmic
Artificial intelligence
Dynamic programming
Prolog
Constraint programming
Network flows
. . .
Applications
Operations research

Constraint Satisfaction Problem

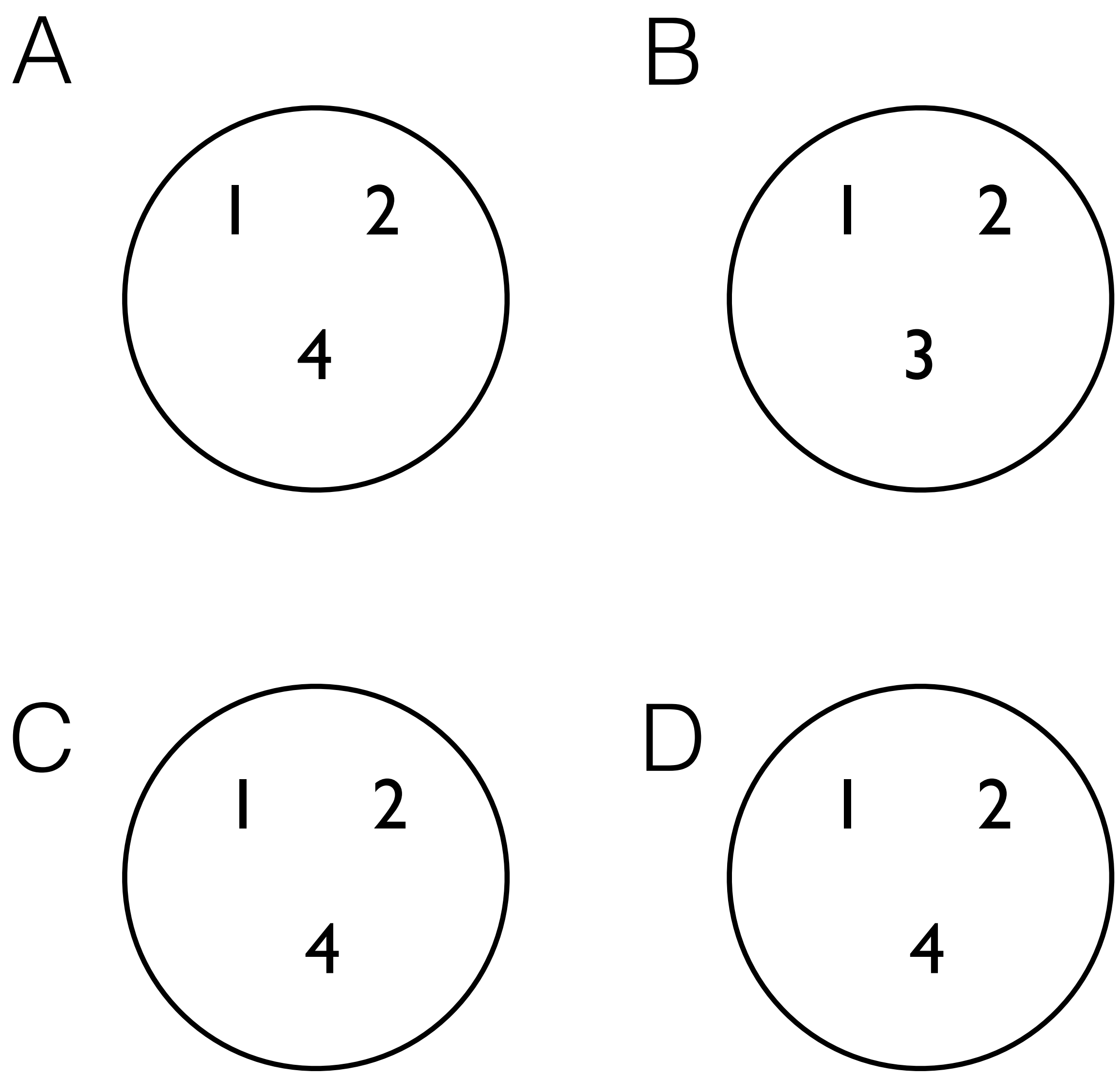


$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$

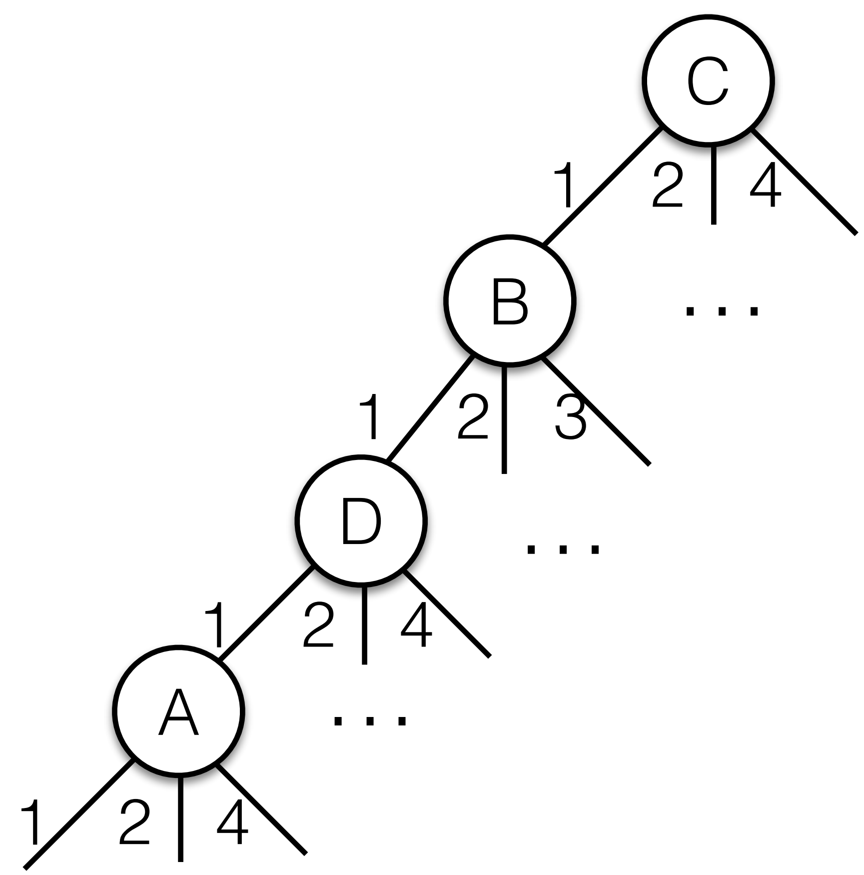
Constraint Satisfaction Problem



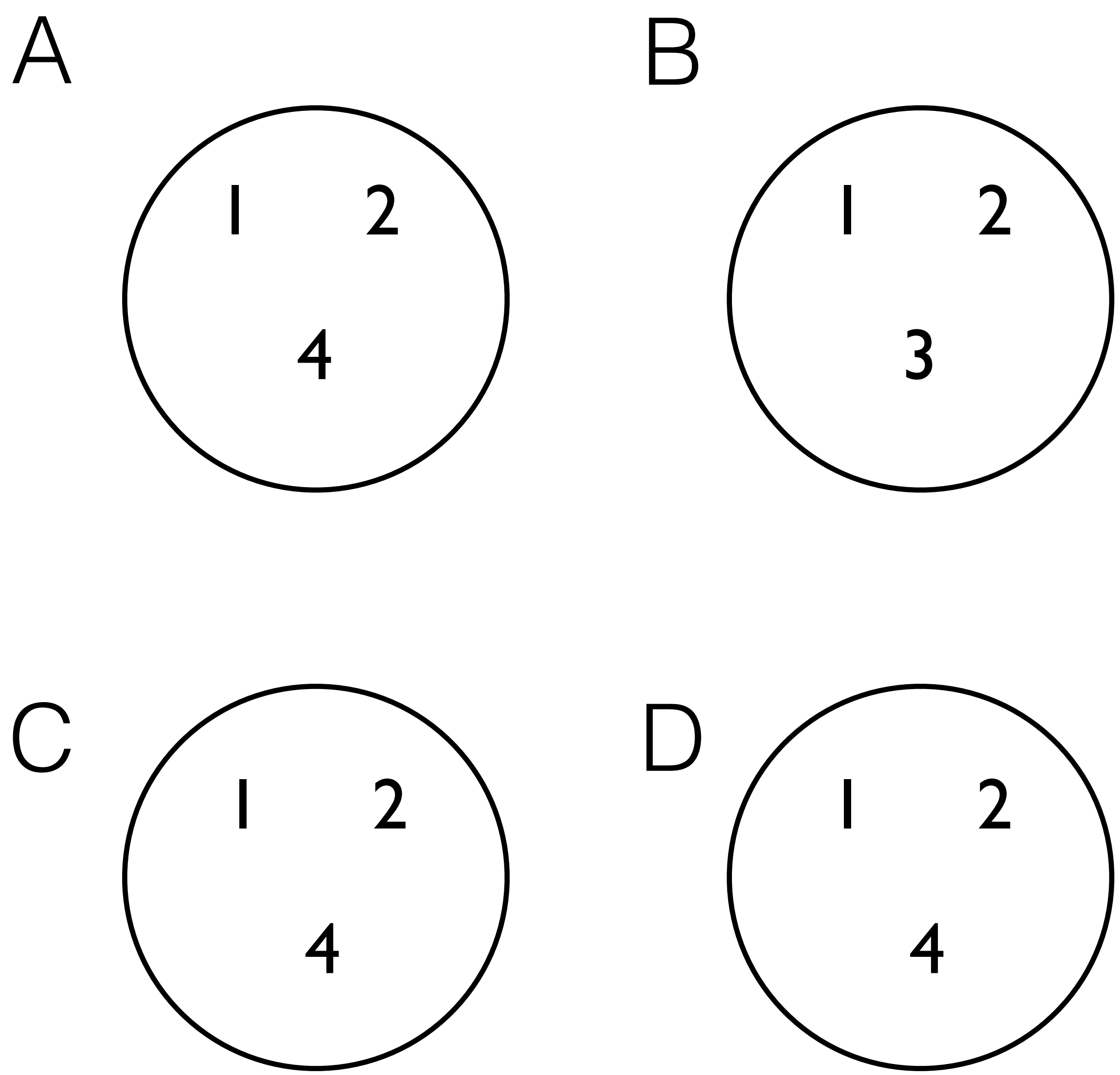
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



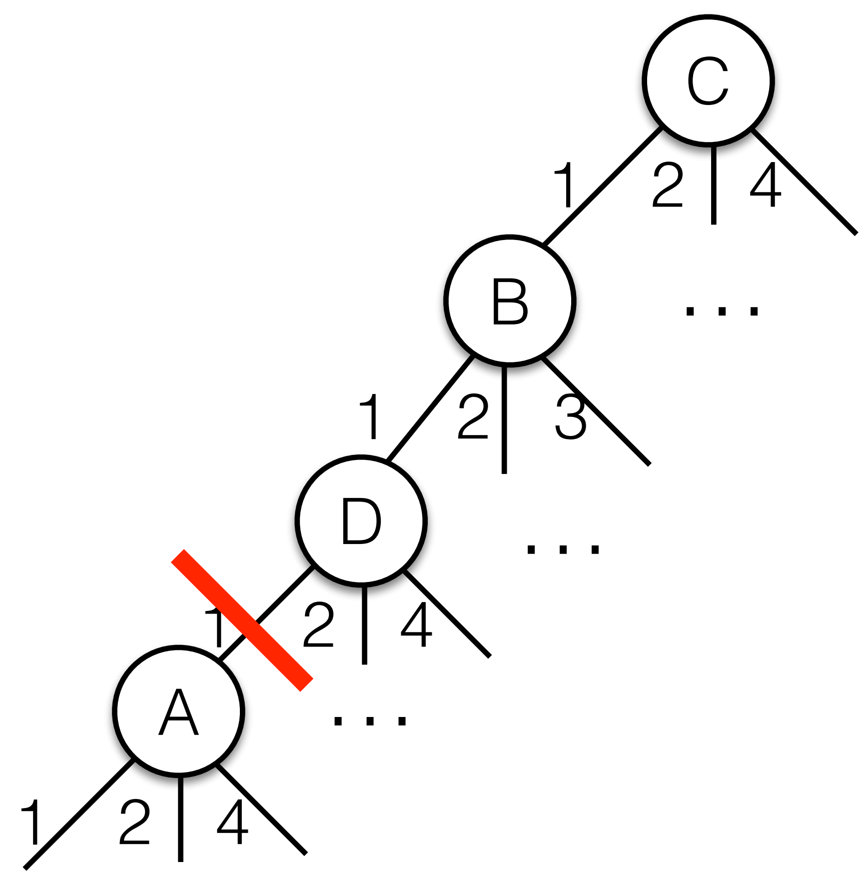
Constraint Satisfaction Problem



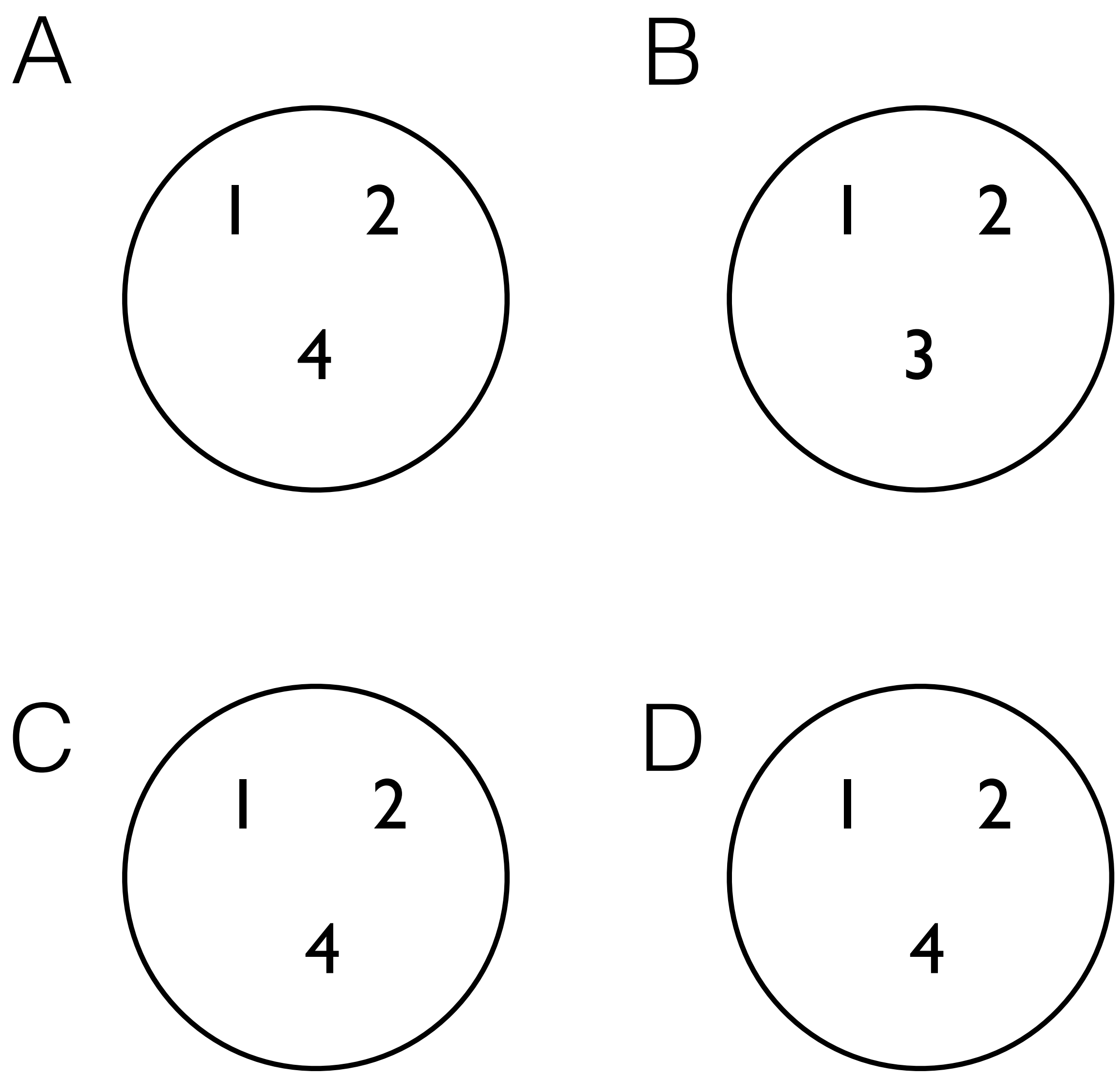
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



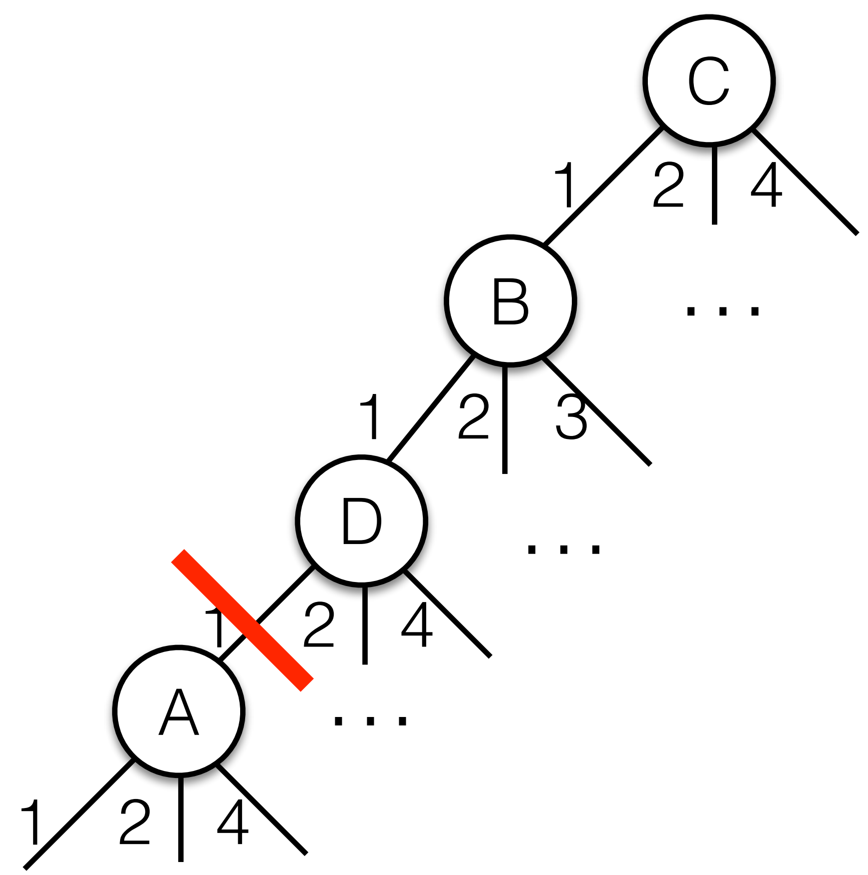
Constraint Satisfaction Problem



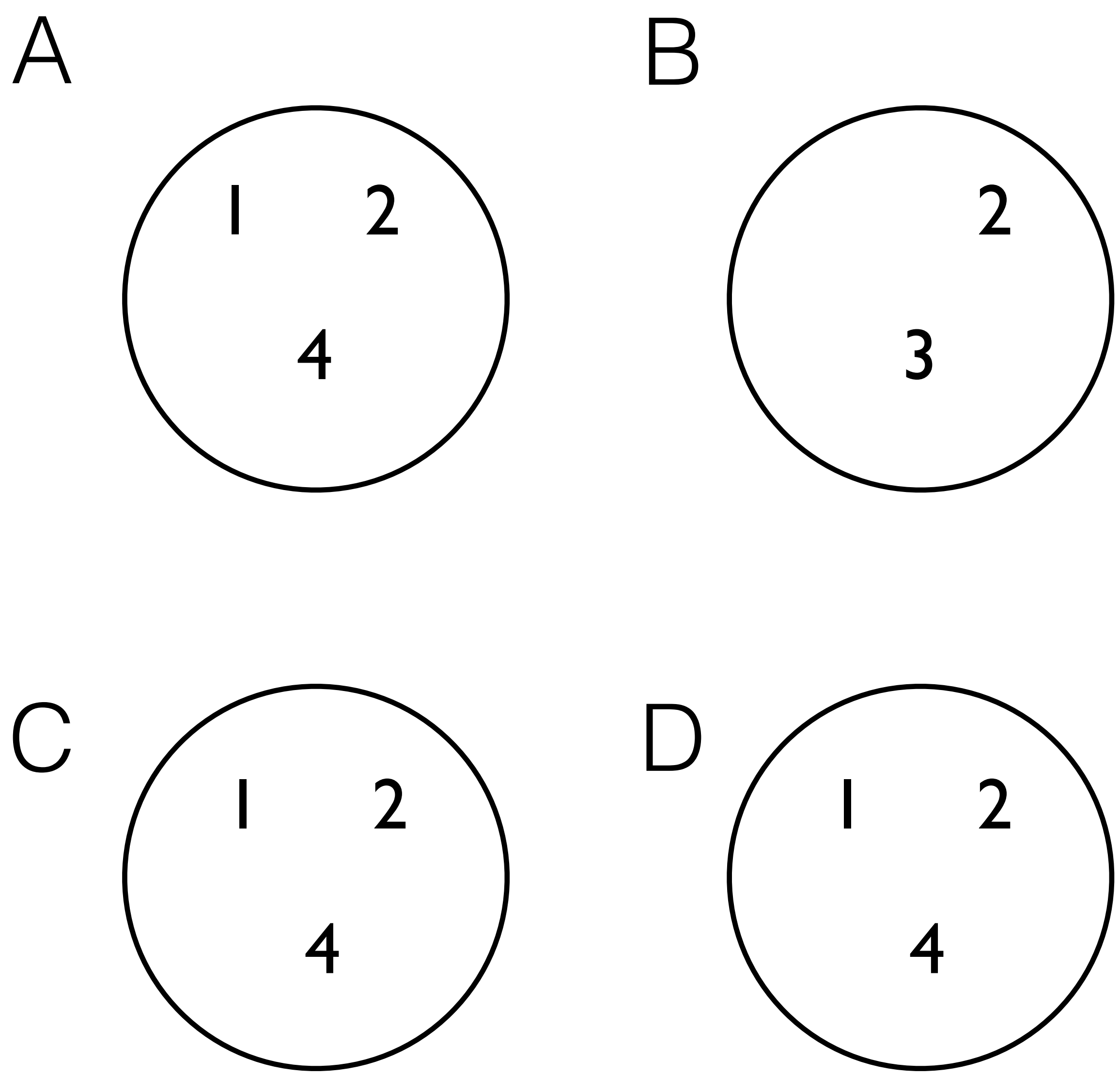
$\boxed{A < B}$ $\quad A \neq D$

$A \neq C \quad C \neq D$

$B + D = 4$



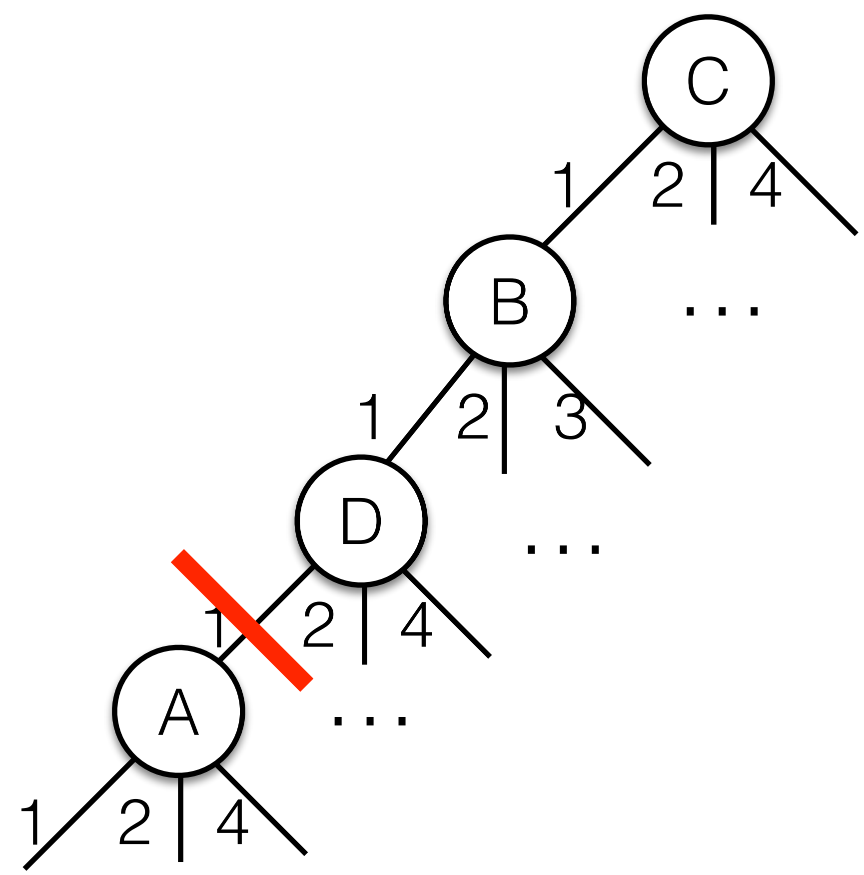
Constraint Satisfaction Problem



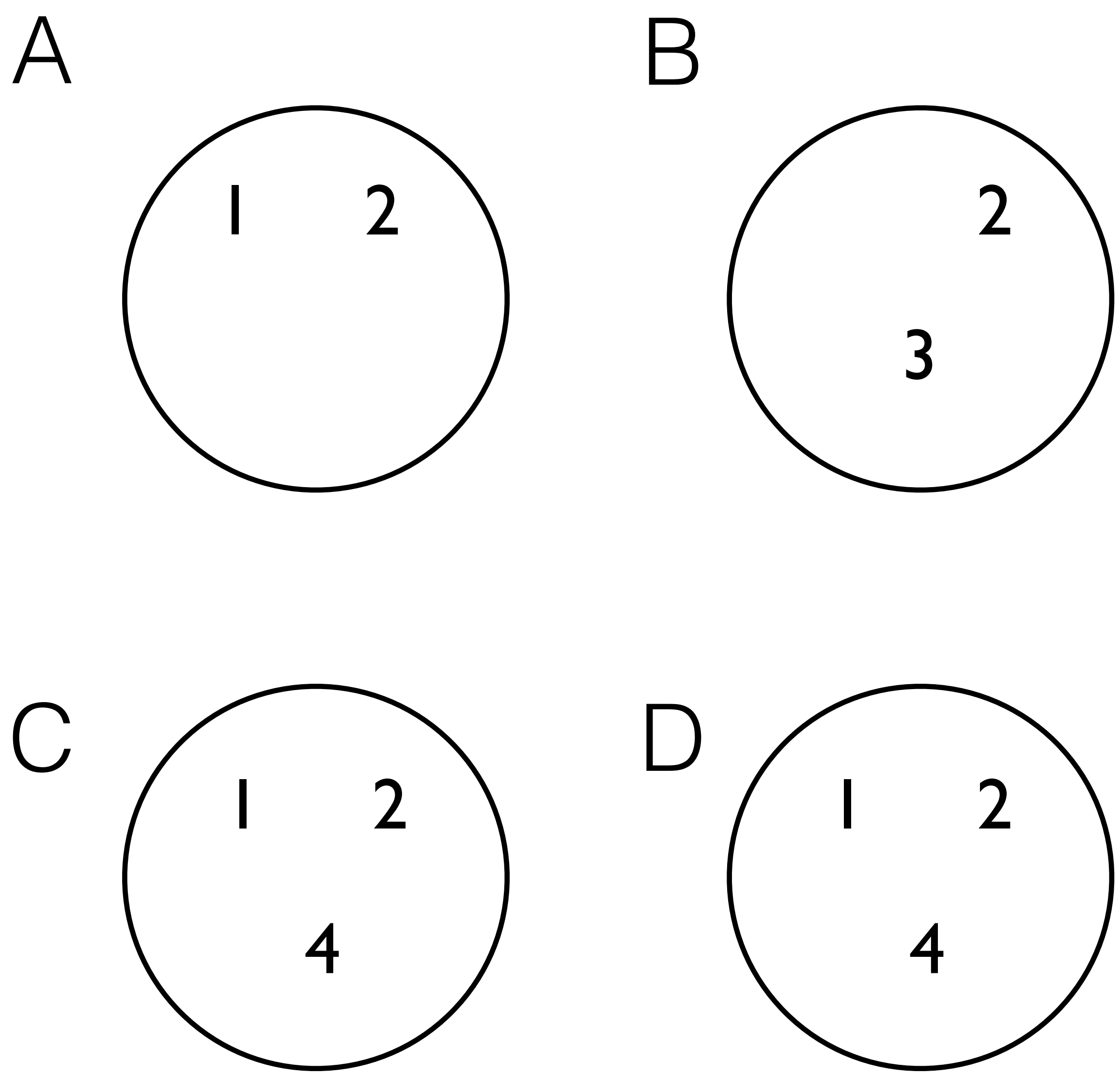
$$\boxed{A < B} \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



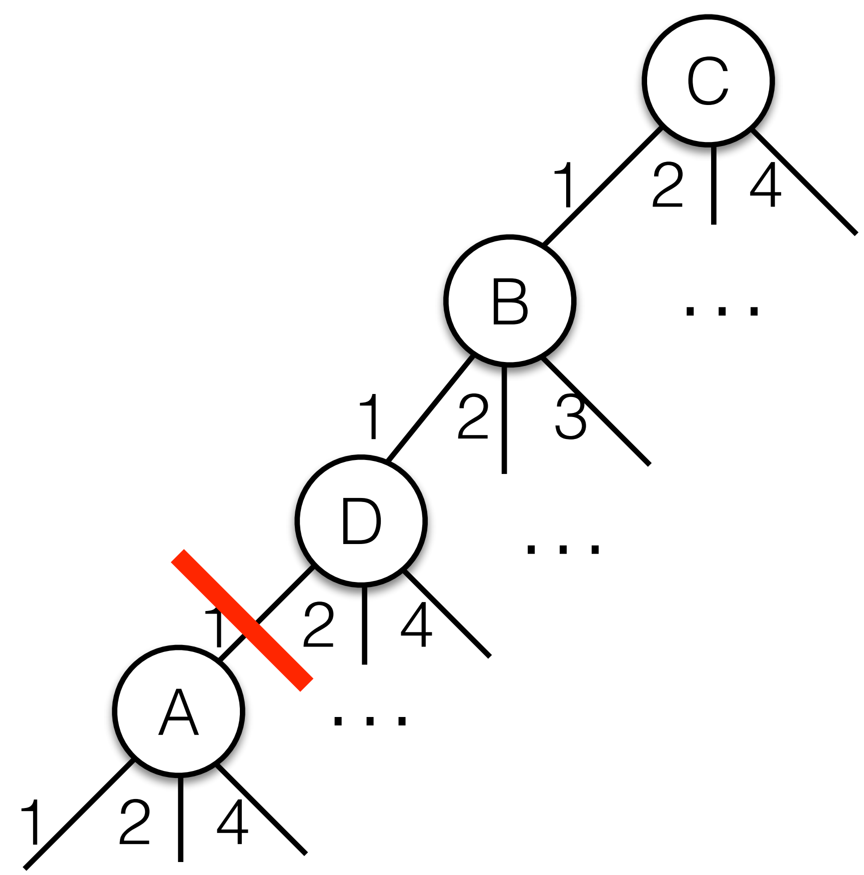
Constraint Satisfaction Problem



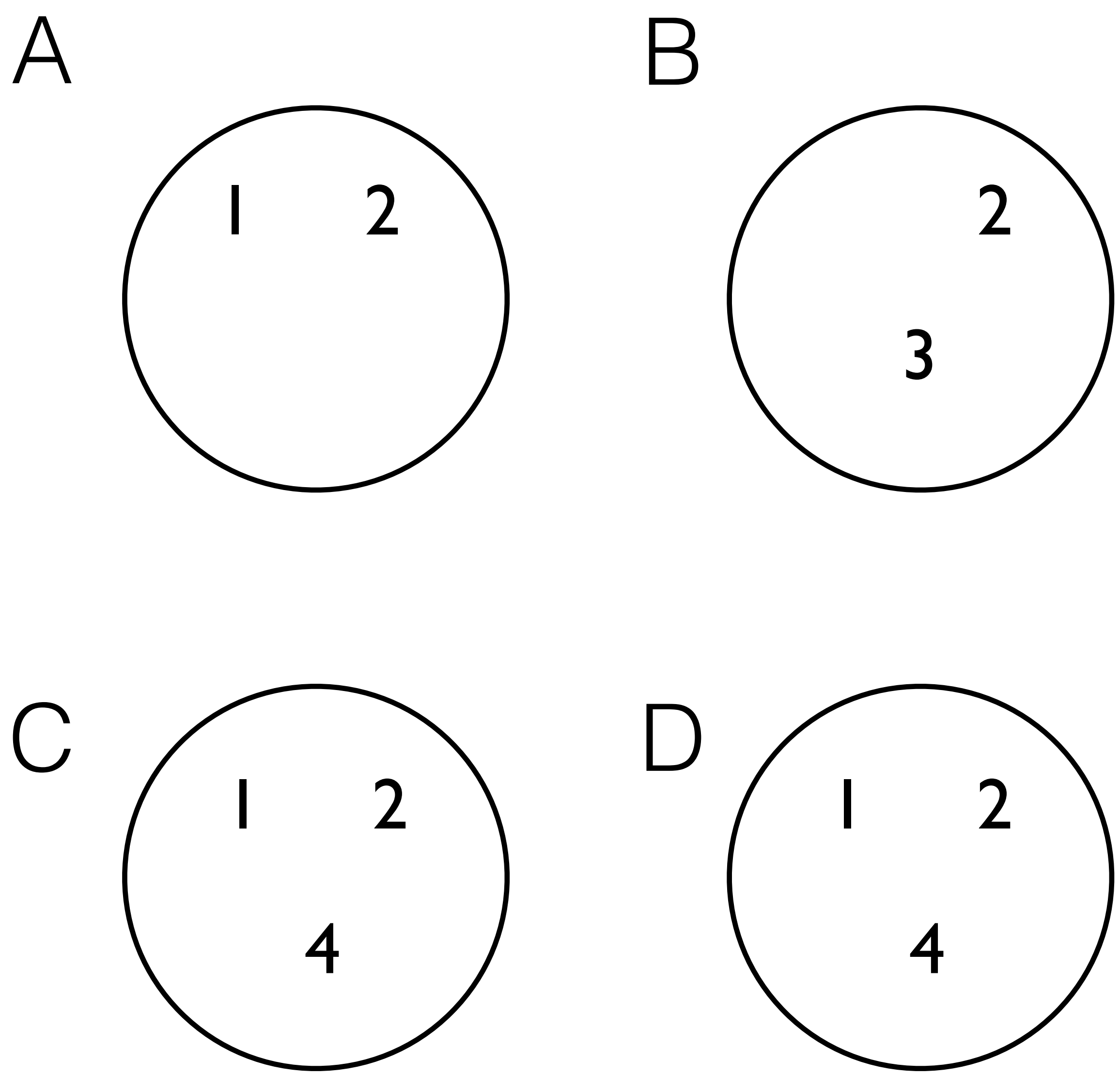
$$\boxed{A < B} \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

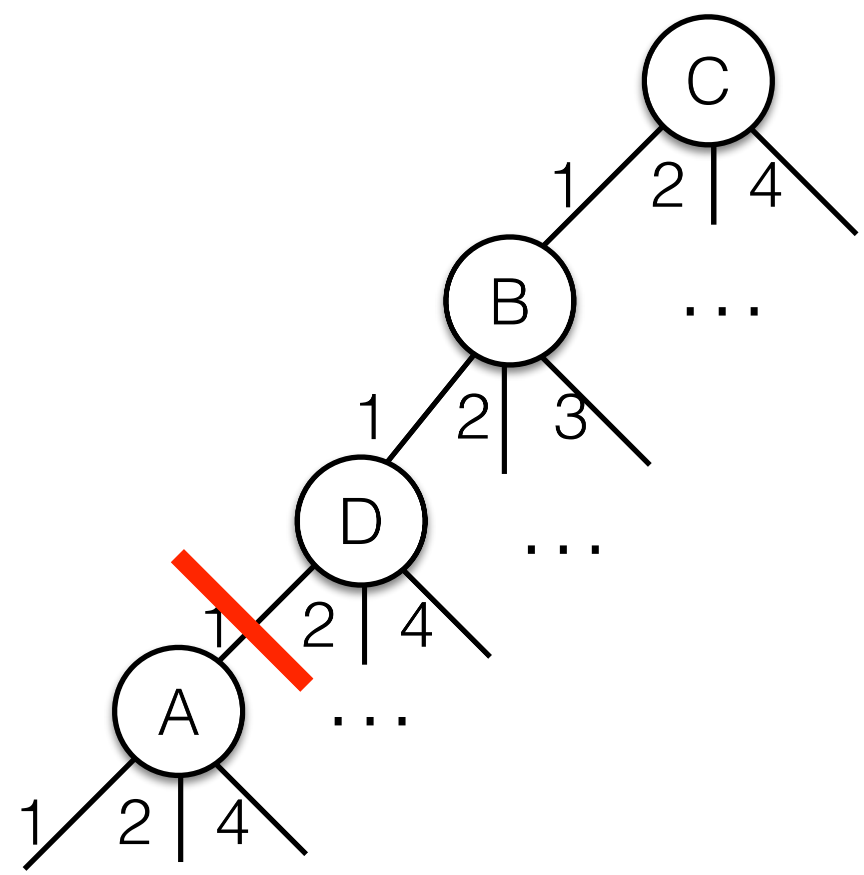
$$B + D = 4$$



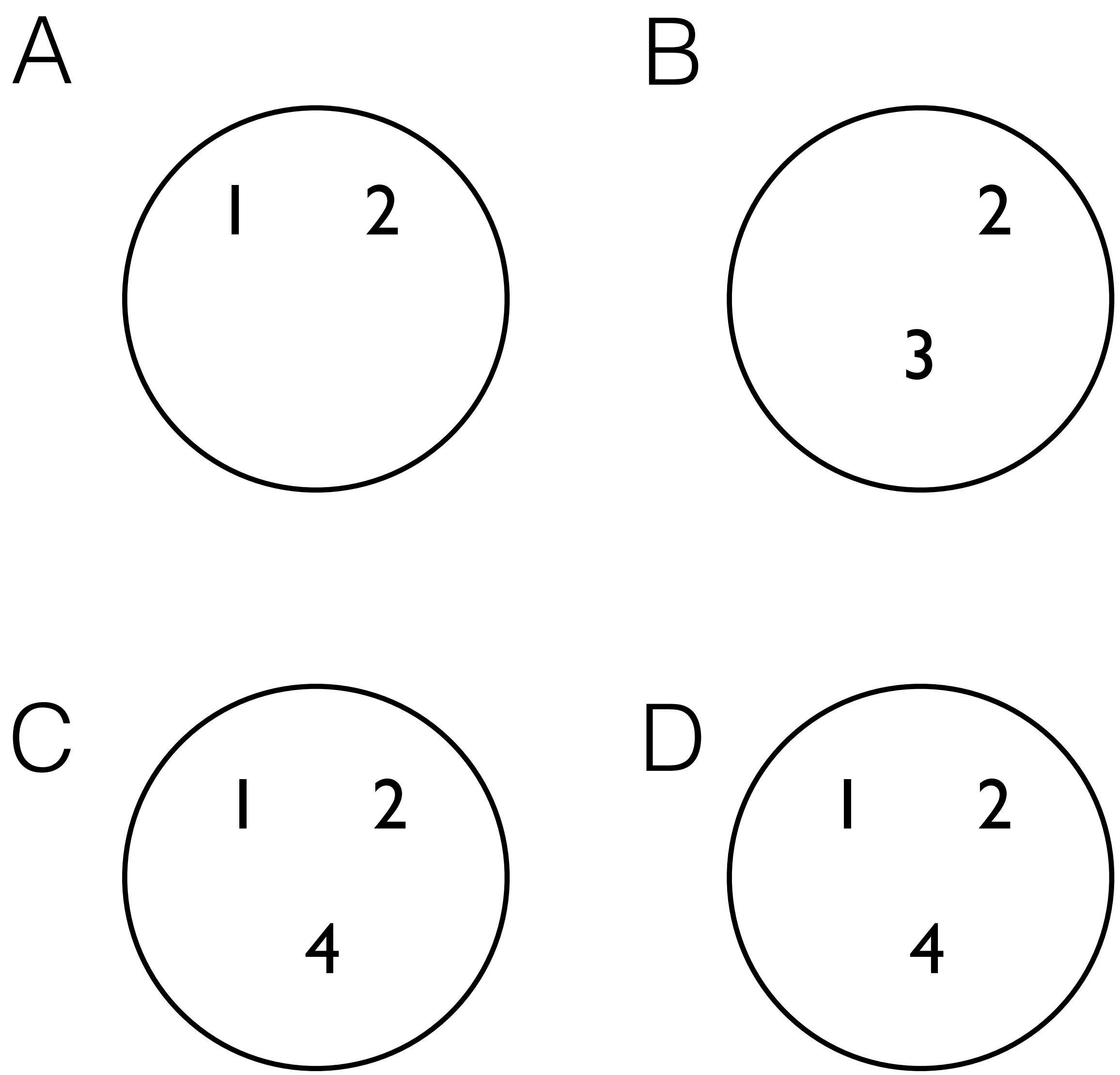
Constraint Satisfaction Problem



$$A < B \qquad A \neq D$$
$$A \neq C \qquad C \neq D$$
$$B + D = 4$$



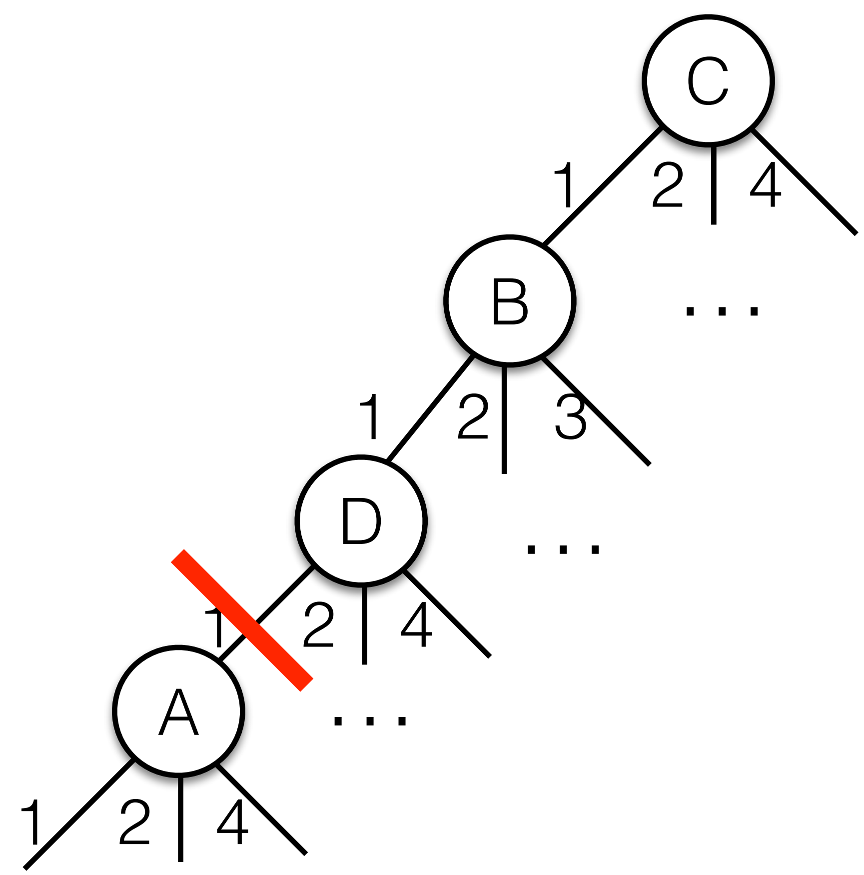
Constraint Satisfaction Problem



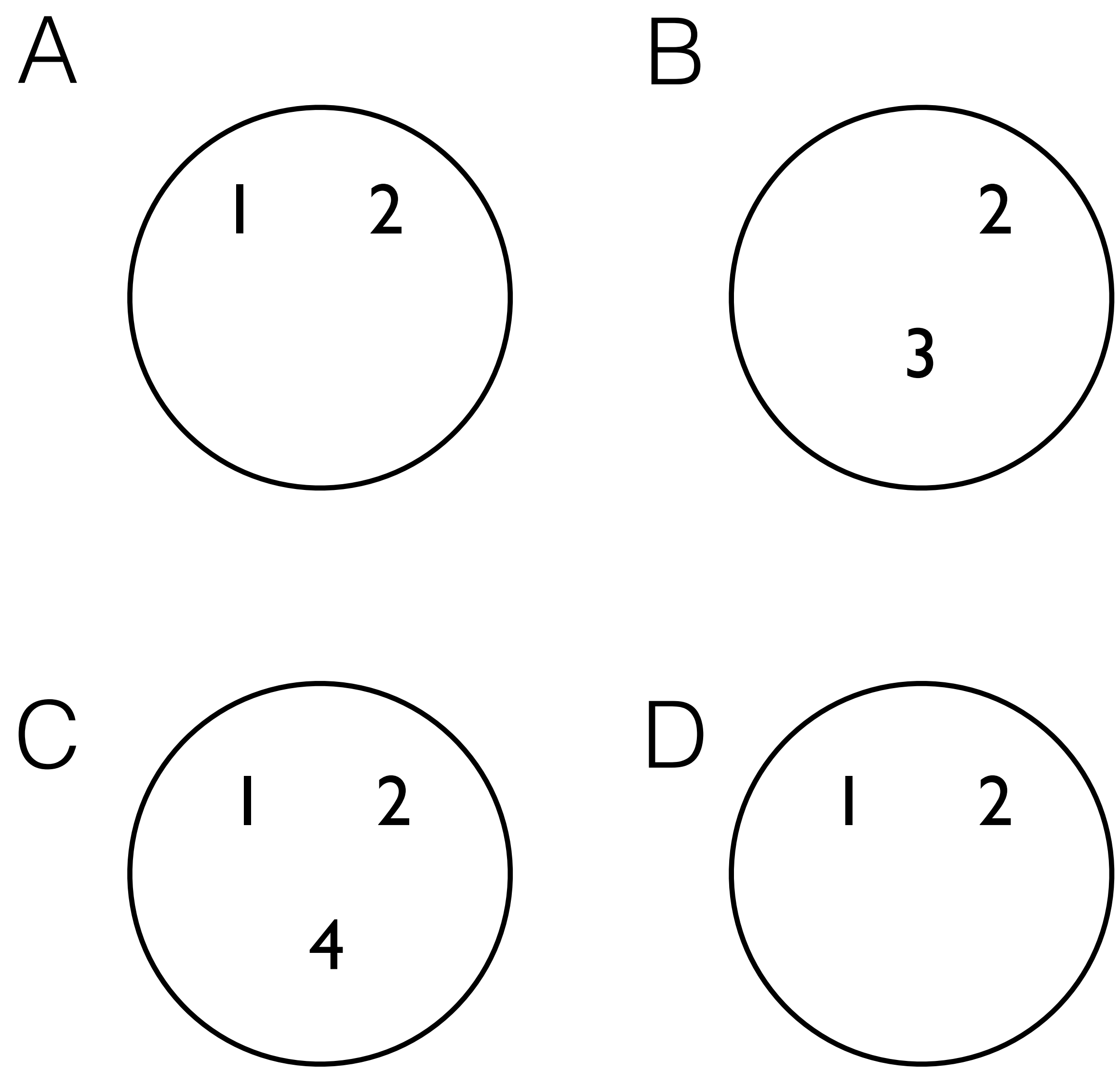
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$\boxed{B + D = 4}$$



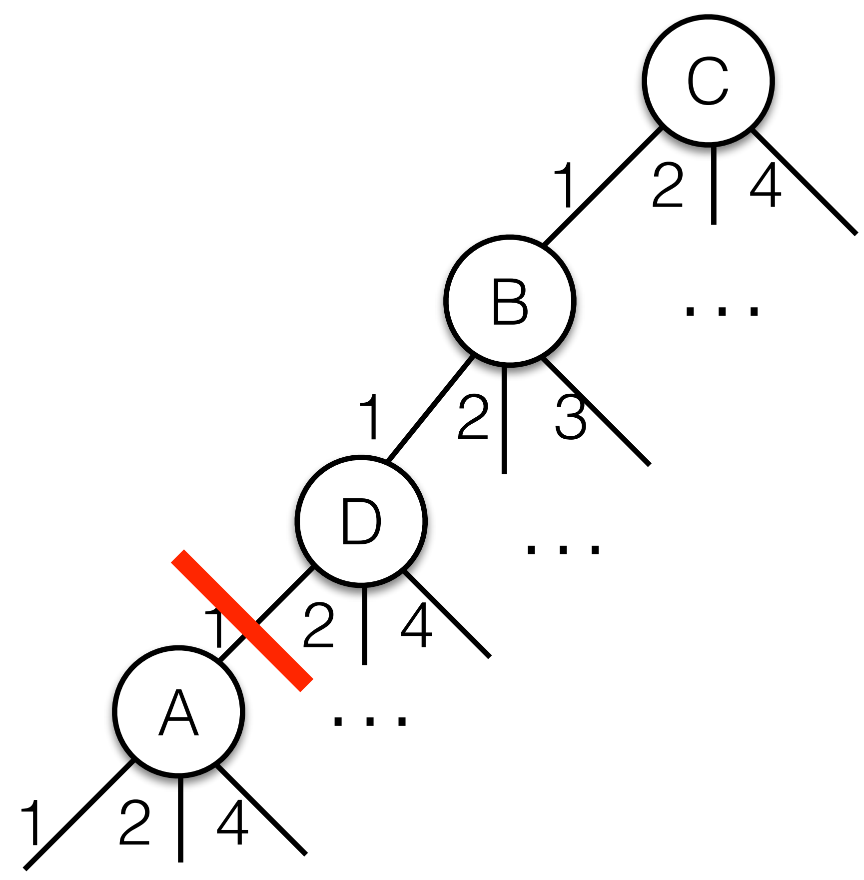
Constraint Satisfaction Problem



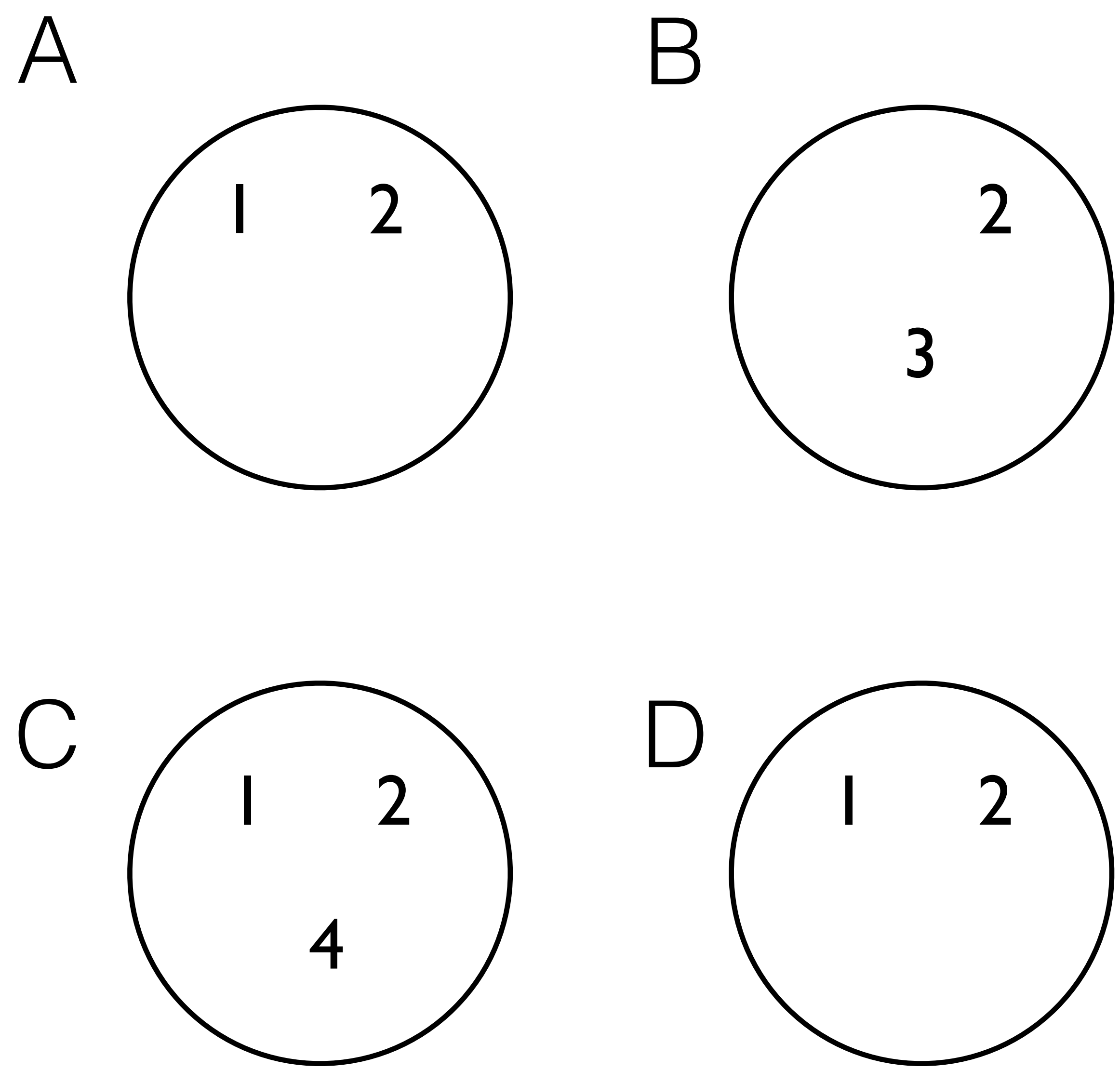
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$\boxed{B + D = 4}$$



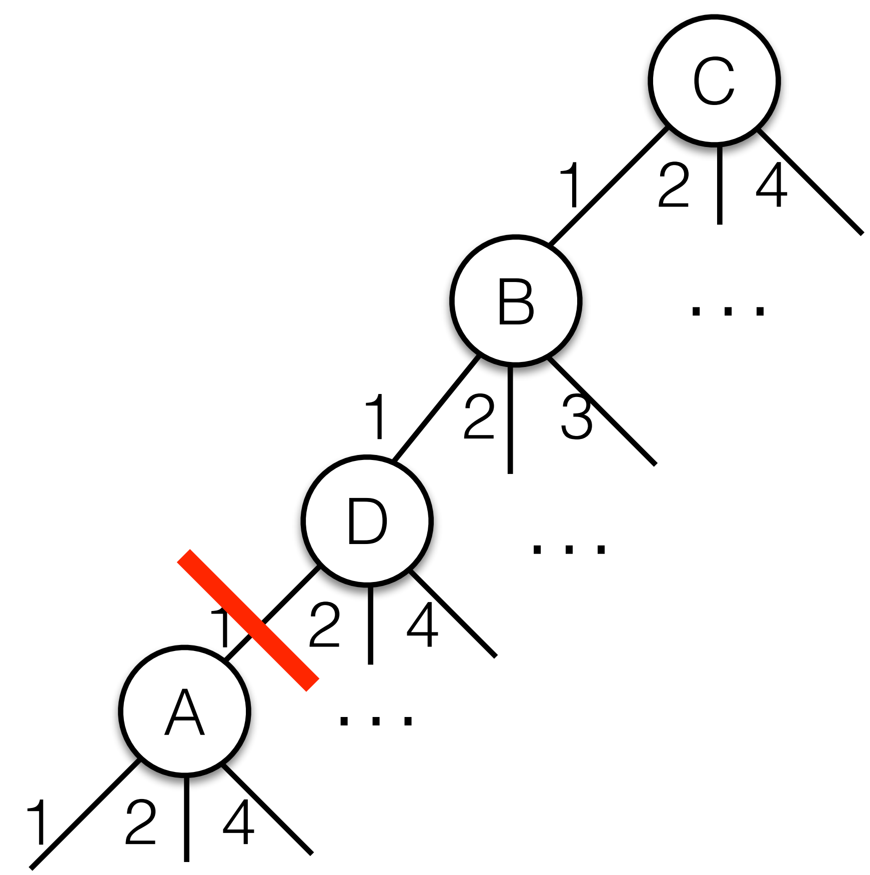
Constraint Satisfaction Problem



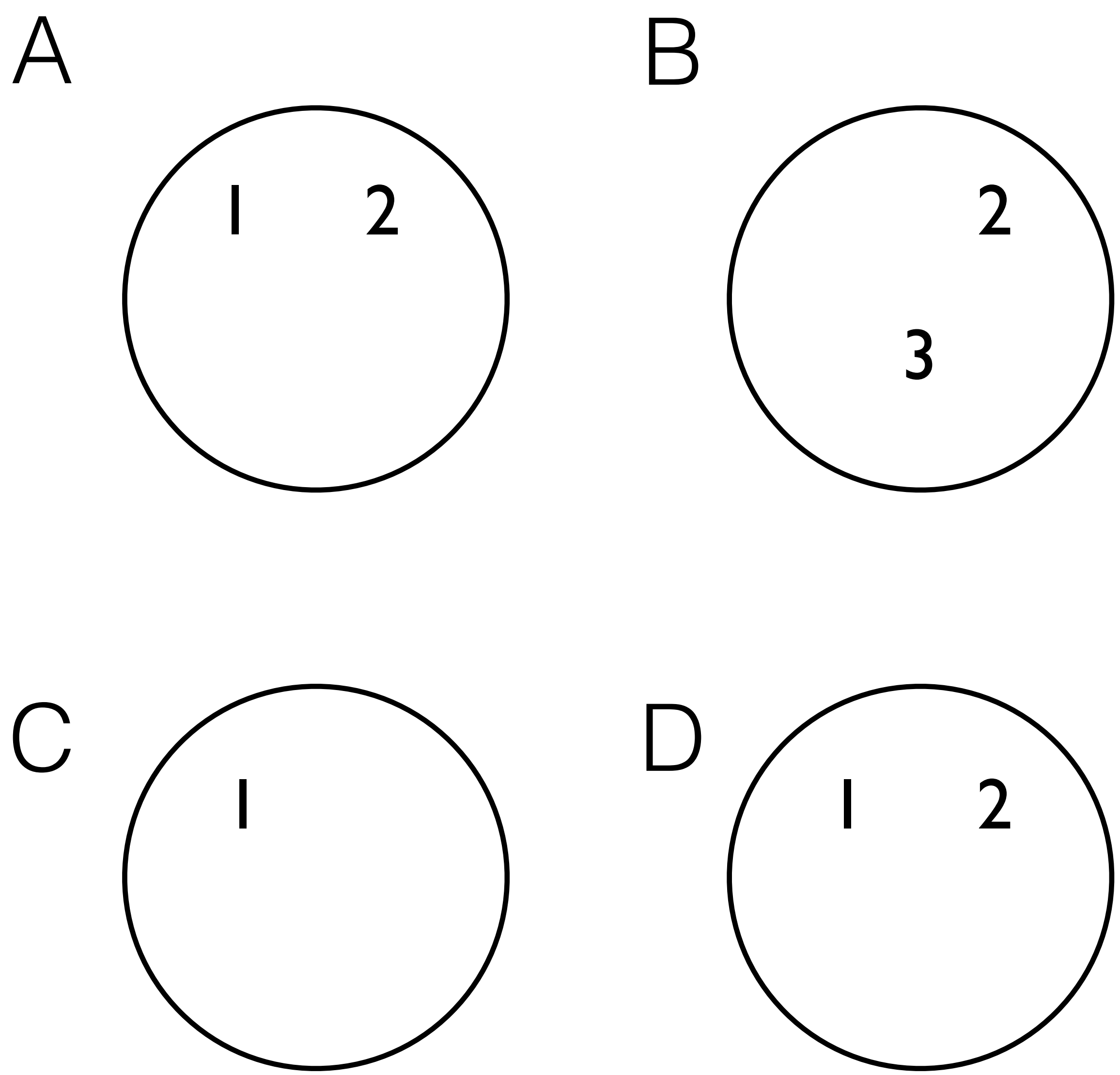
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



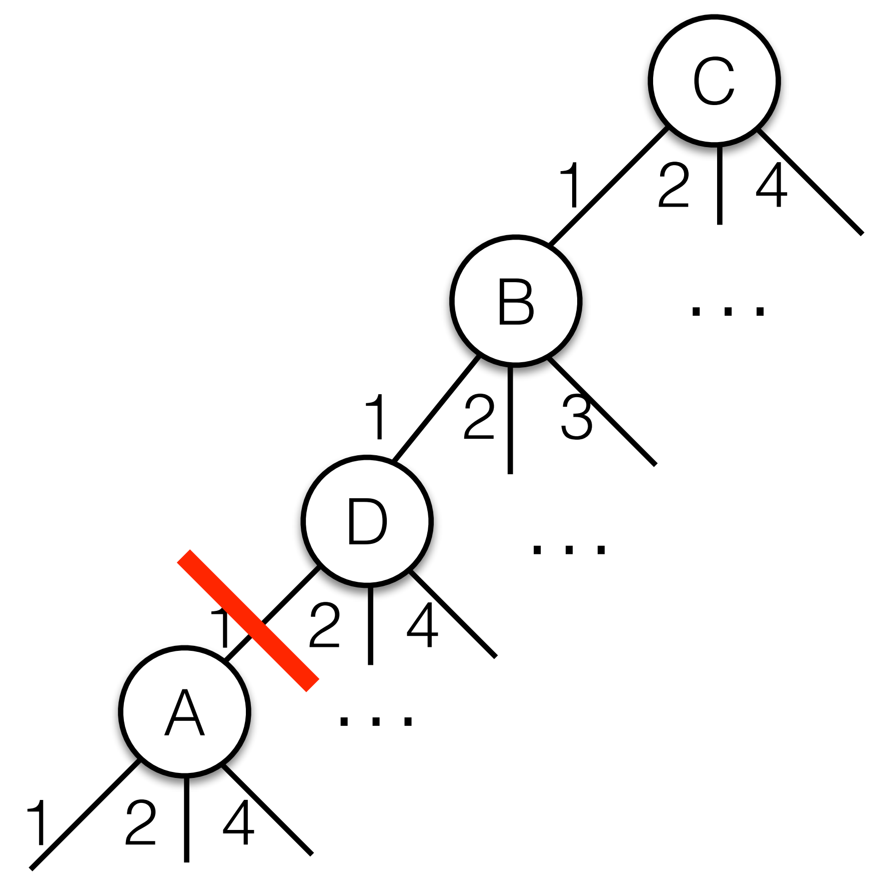
Constraint Satisfaction Problem



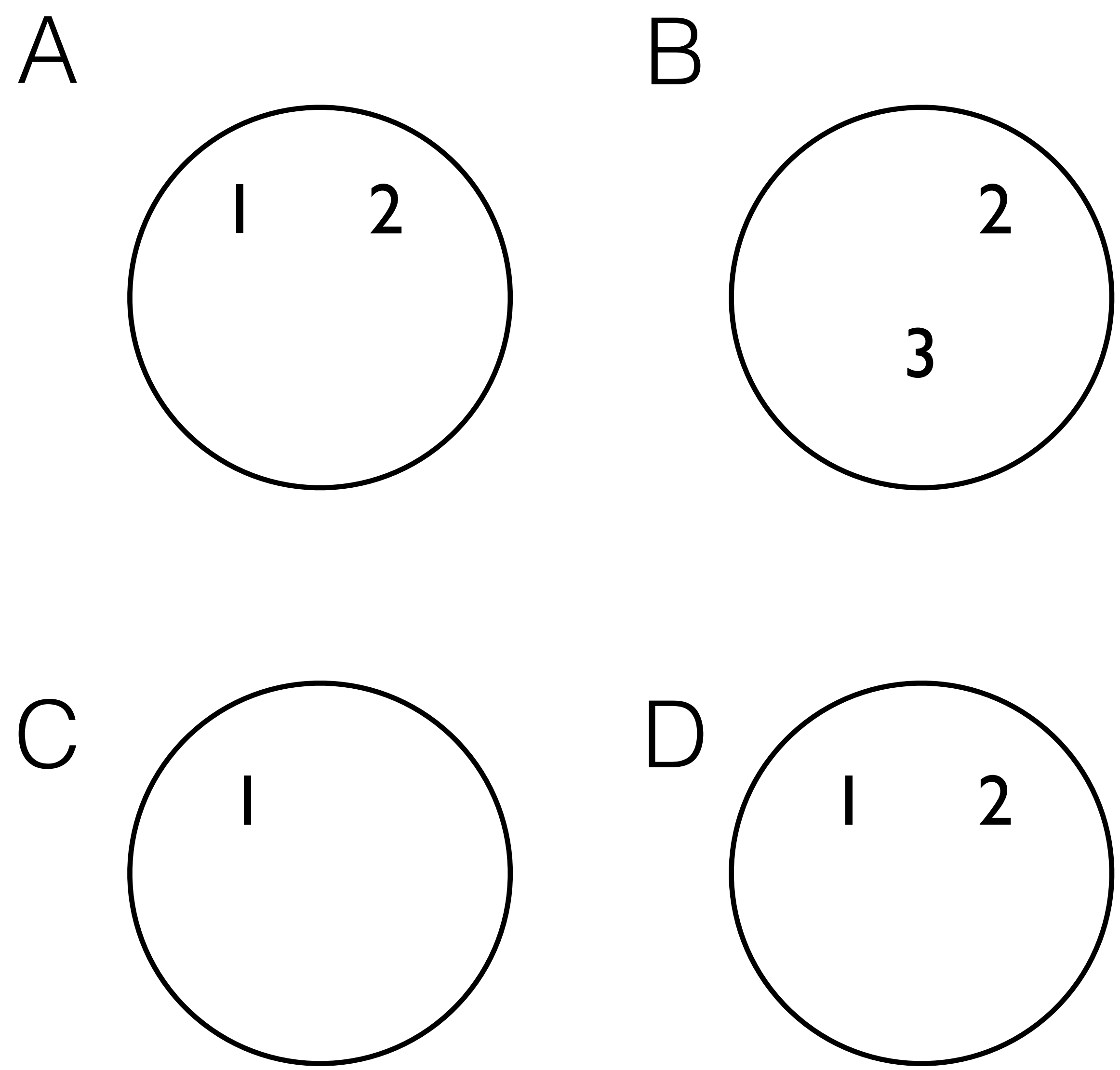
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



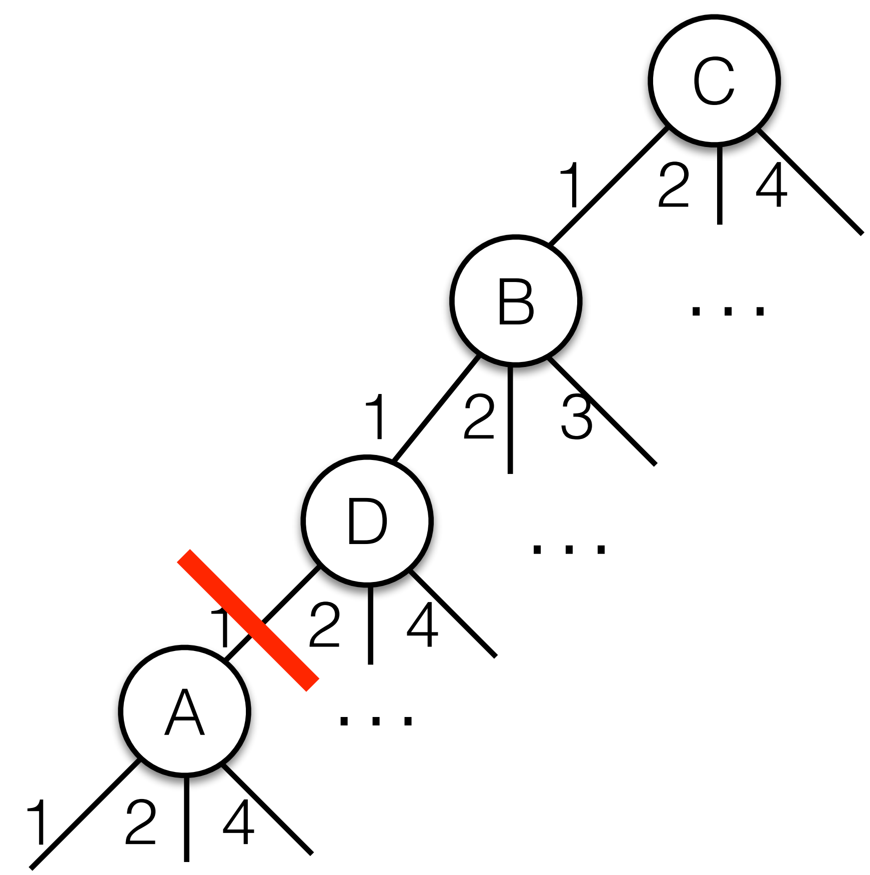
Constraint Satisfaction Problem



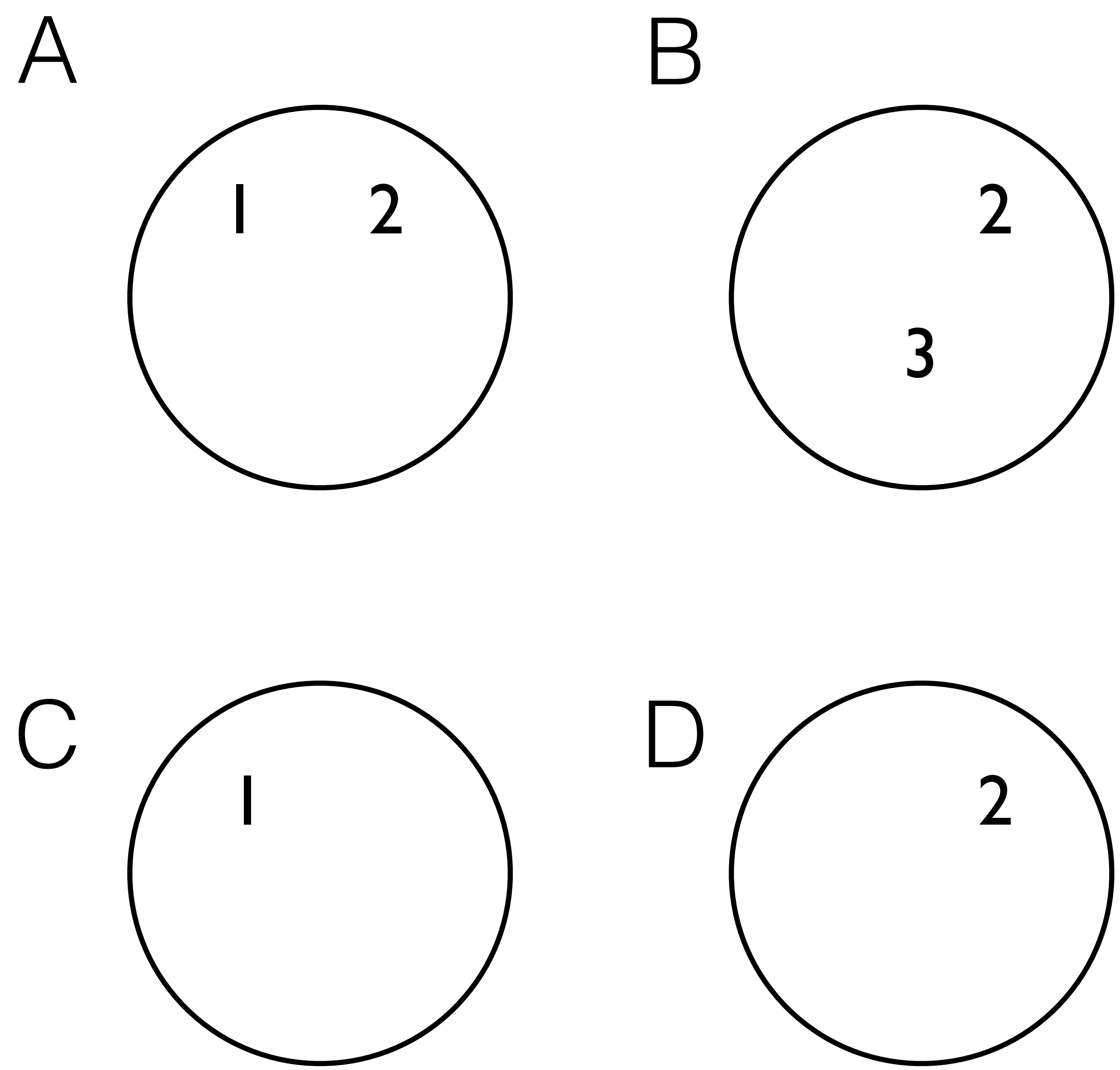
$$A < B \qquad A \neq D$$

$$A \neq C \qquad \boxed{C \neq D}$$

$$B + D = 4$$



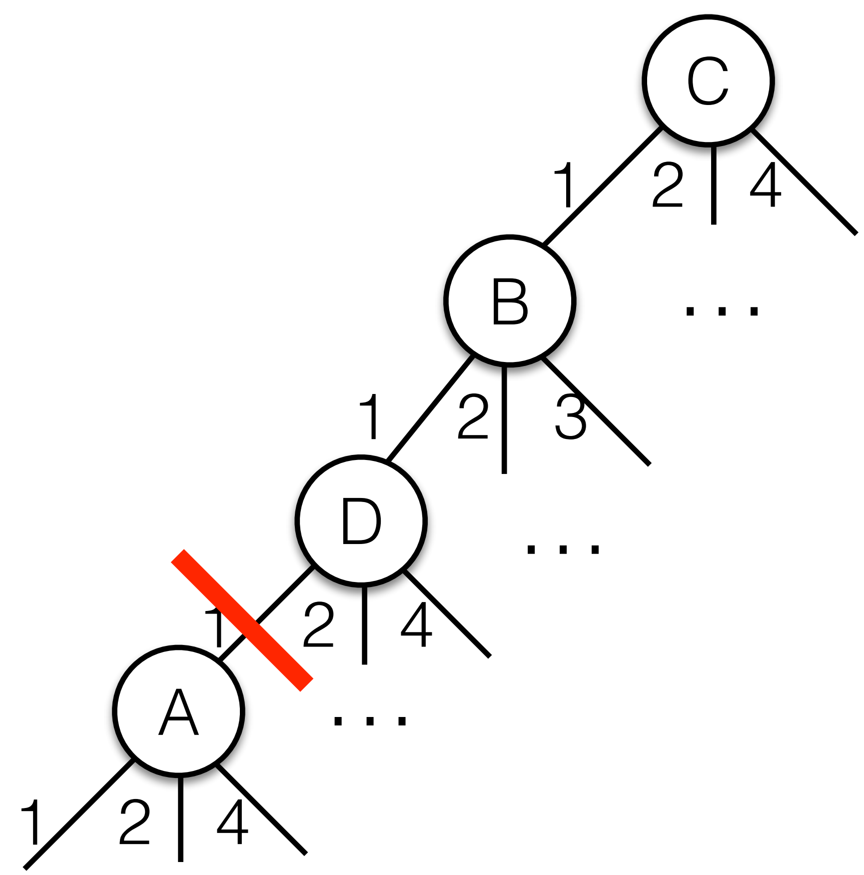
Constraint Satisfaction Problem



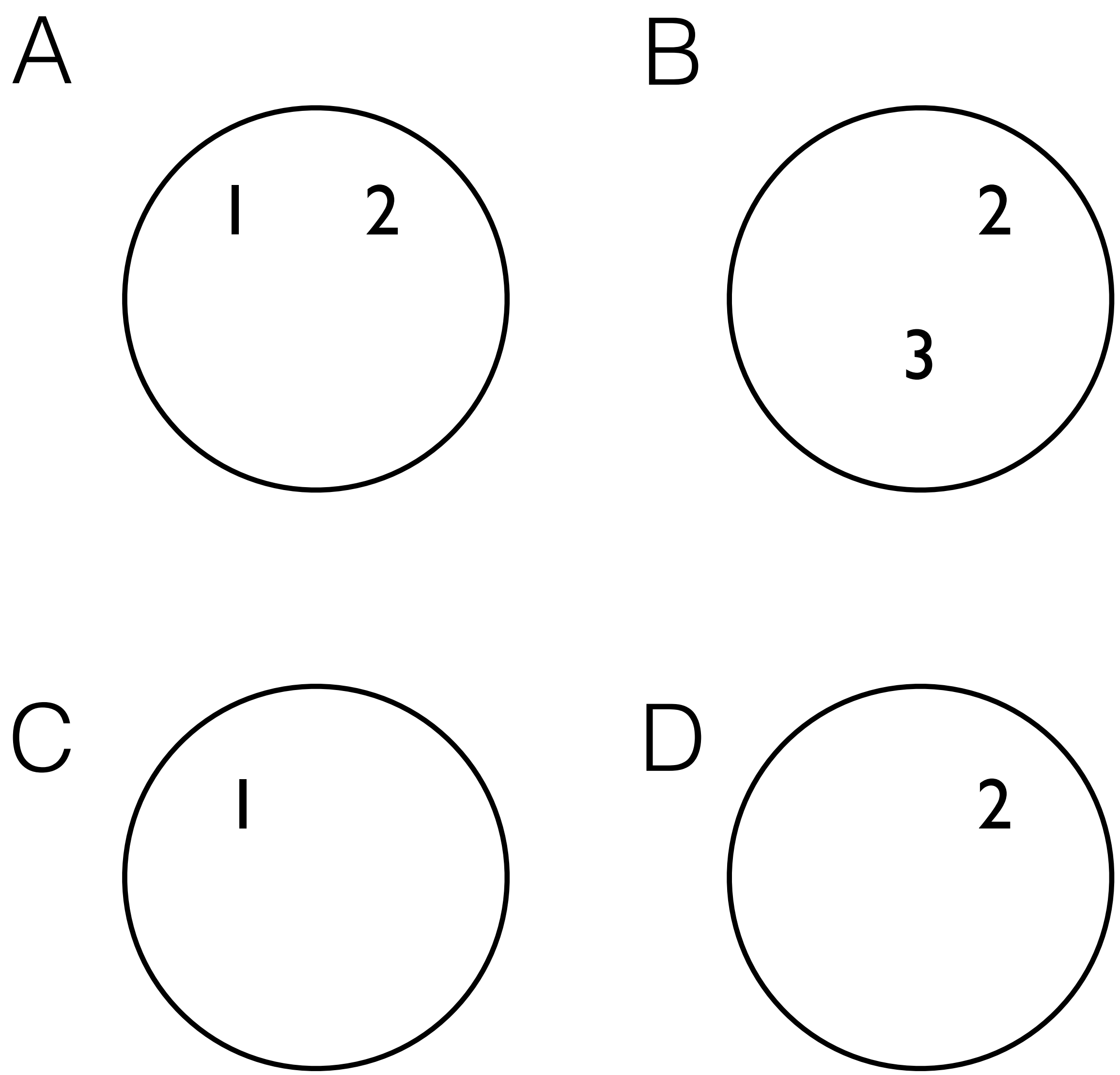
$$A < B \qquad A \neq D$$

$$A \neq C \qquad \boxed{C \neq D}$$

$$B + D = 4$$



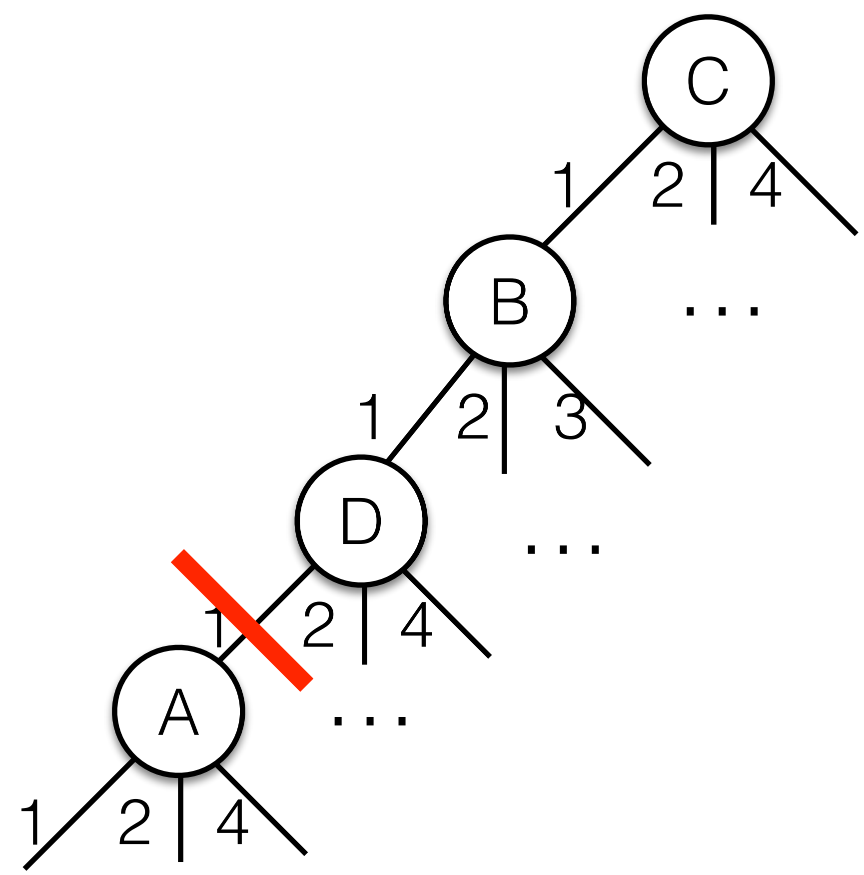
Constraint Satisfaction Problem



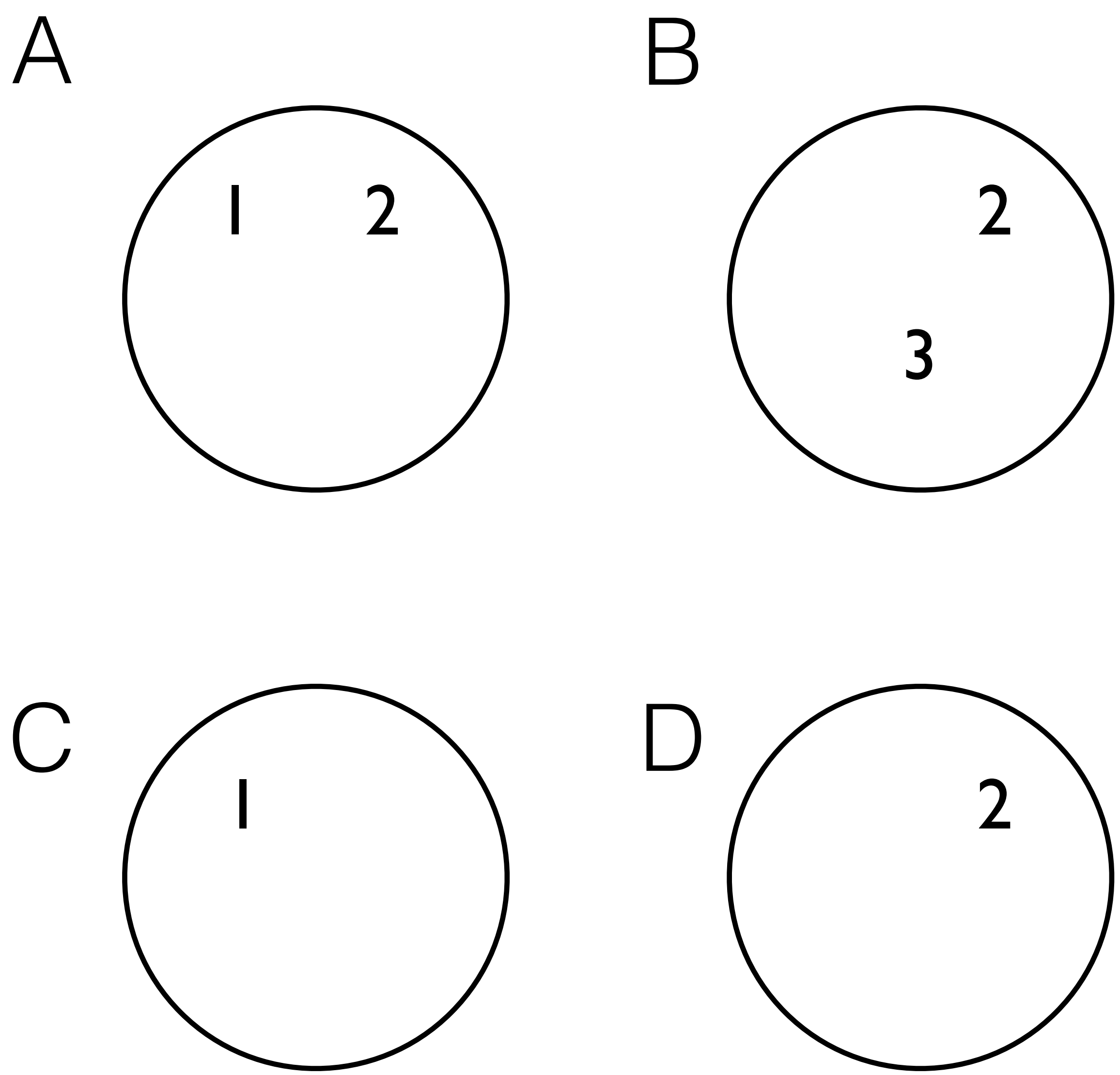
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



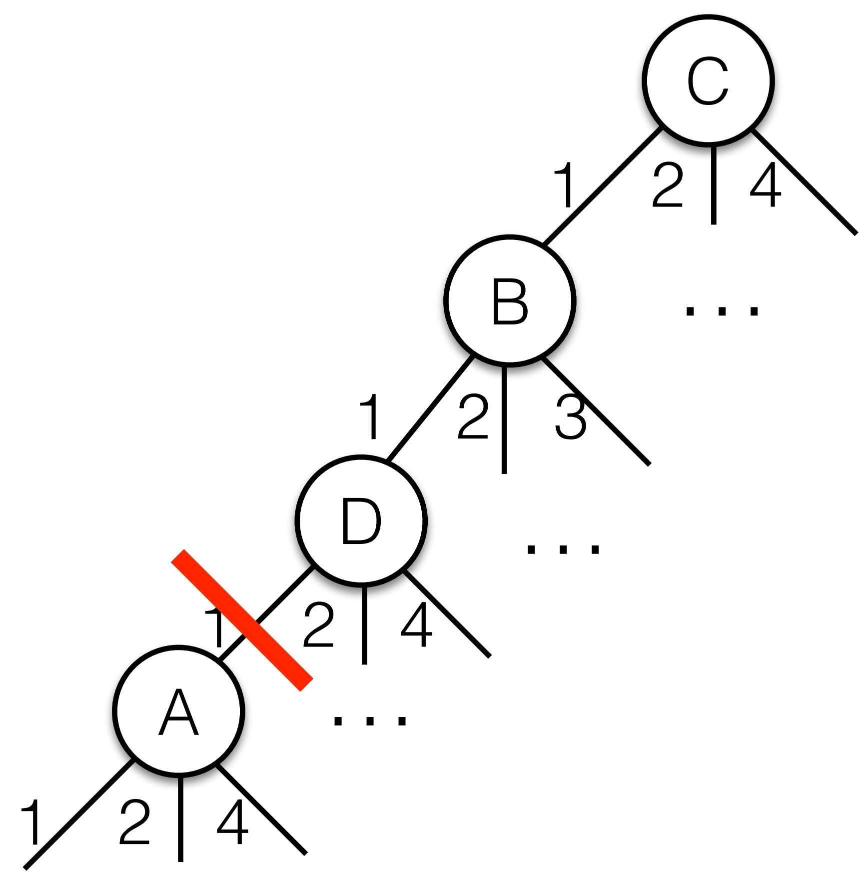
Constraint Satisfaction Problem



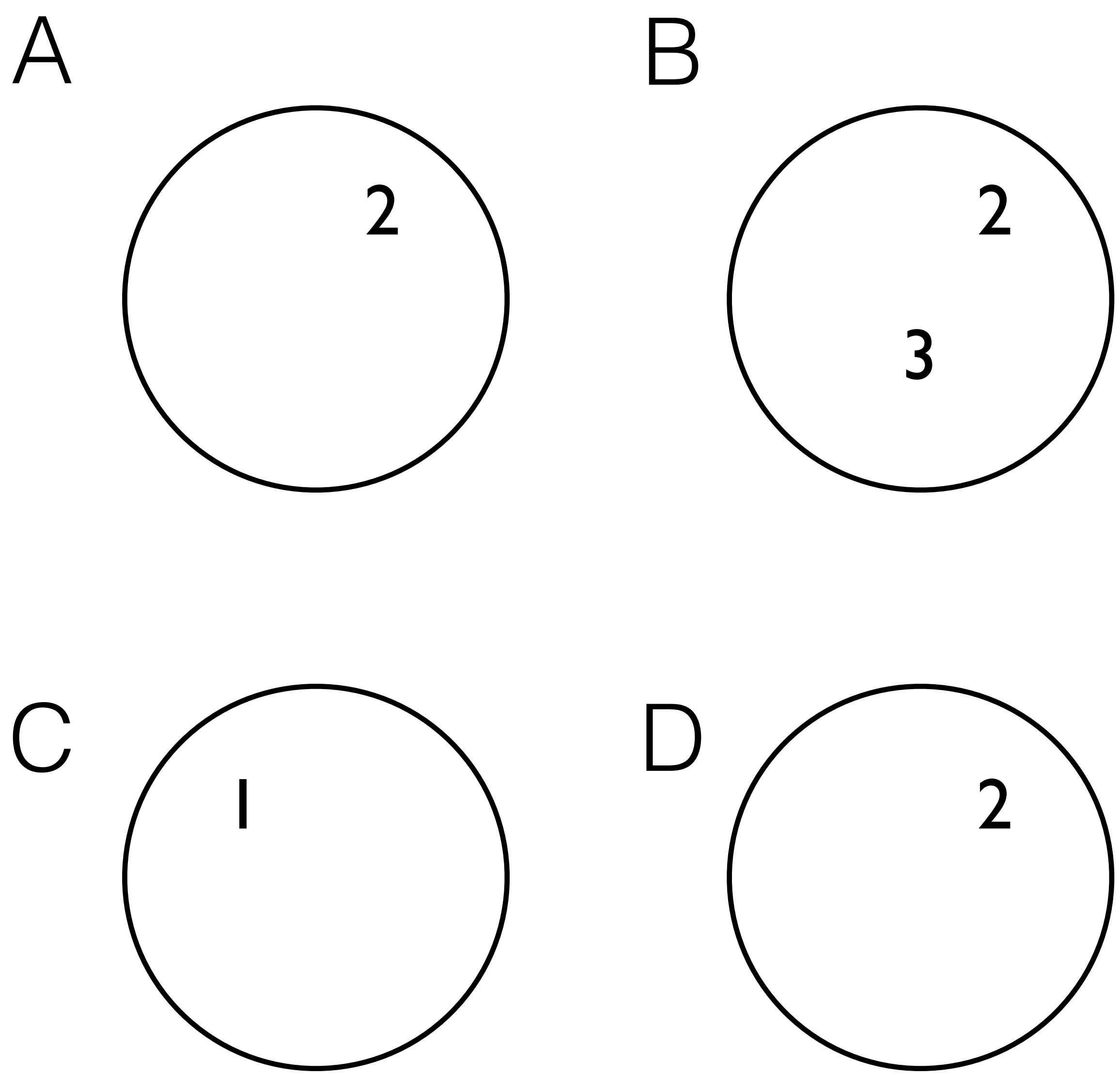
$$A < B \qquad A \neq D$$

$$\boxed{A \neq C} \qquad C \neq D$$

$$B + D = 4$$



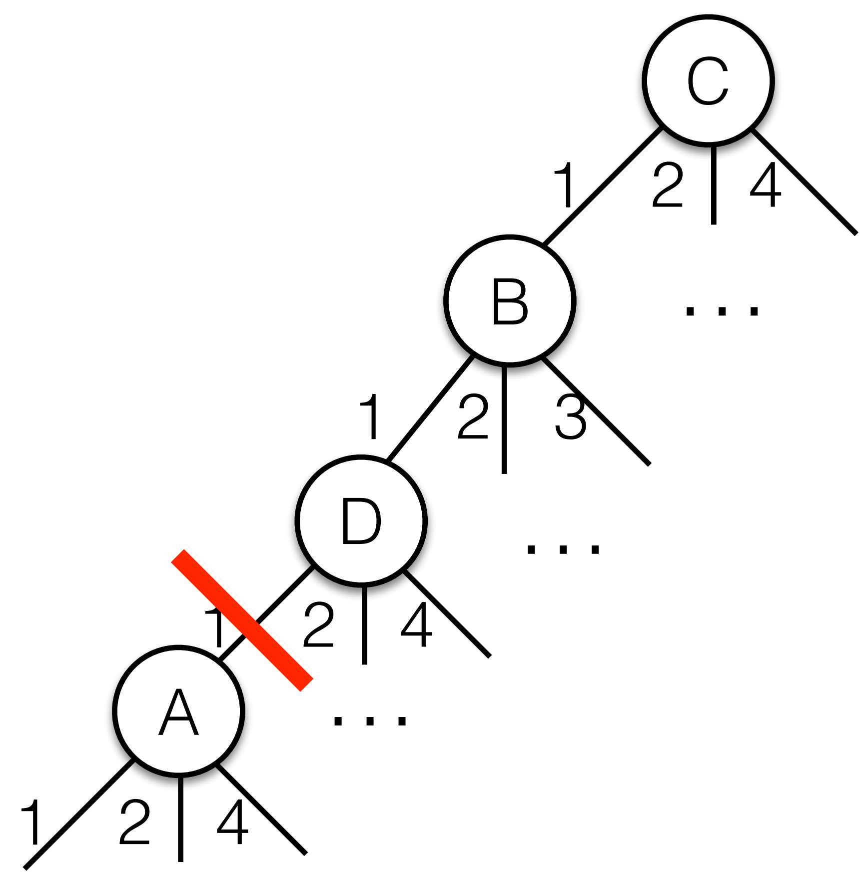
Constraint Satisfaction Problem



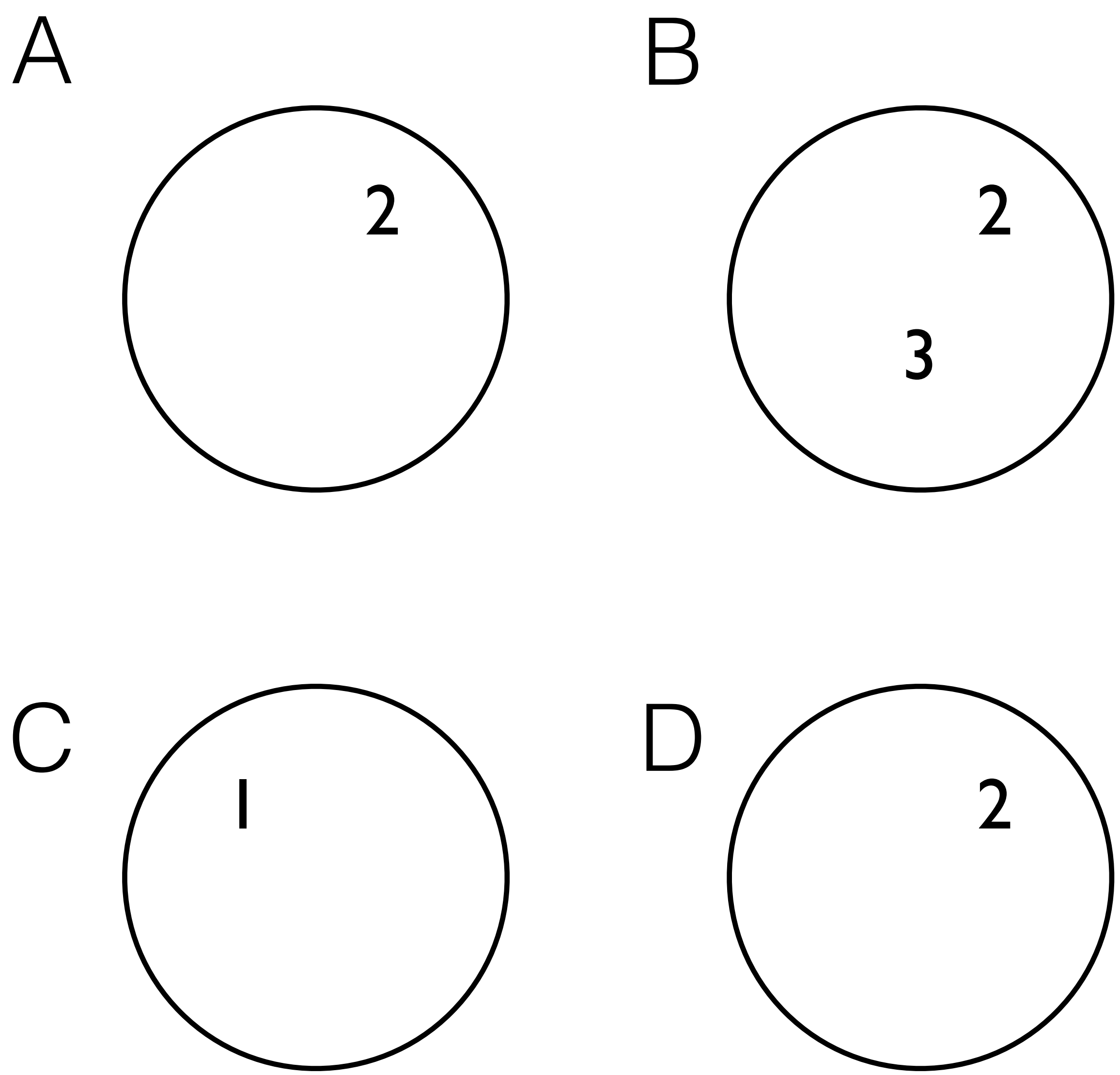
$$A < B \qquad A \neq D$$

$$\boxed{A \neq C} \qquad C \neq D$$

$$B + D = 4$$



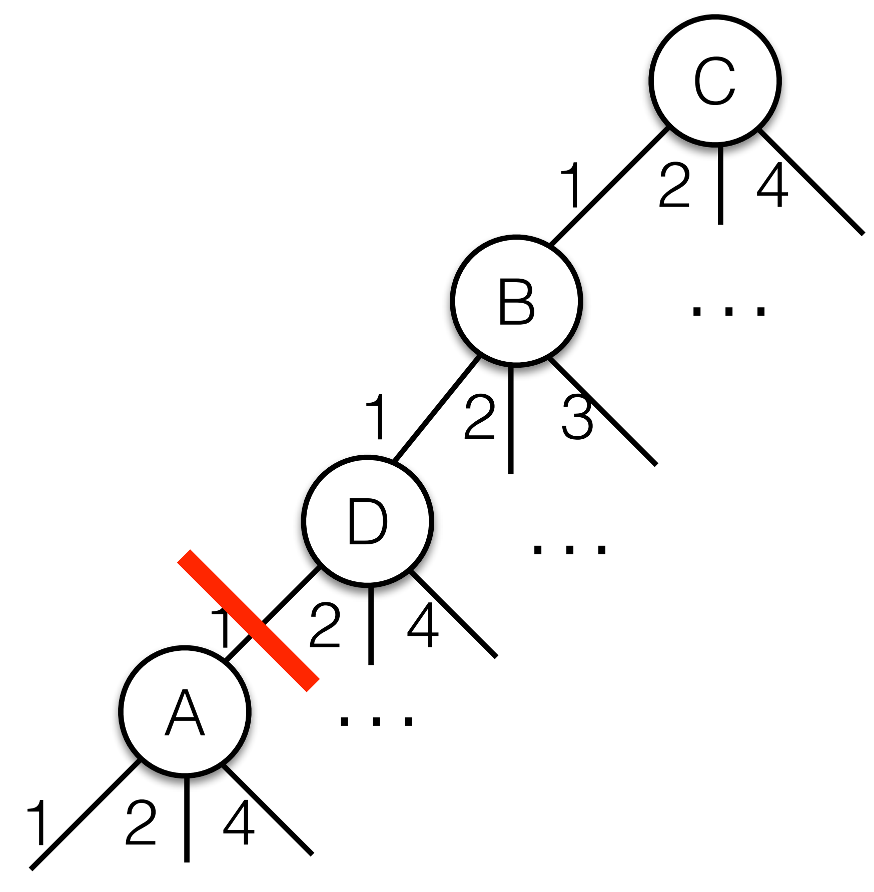
Constraint Satisfaction Problem



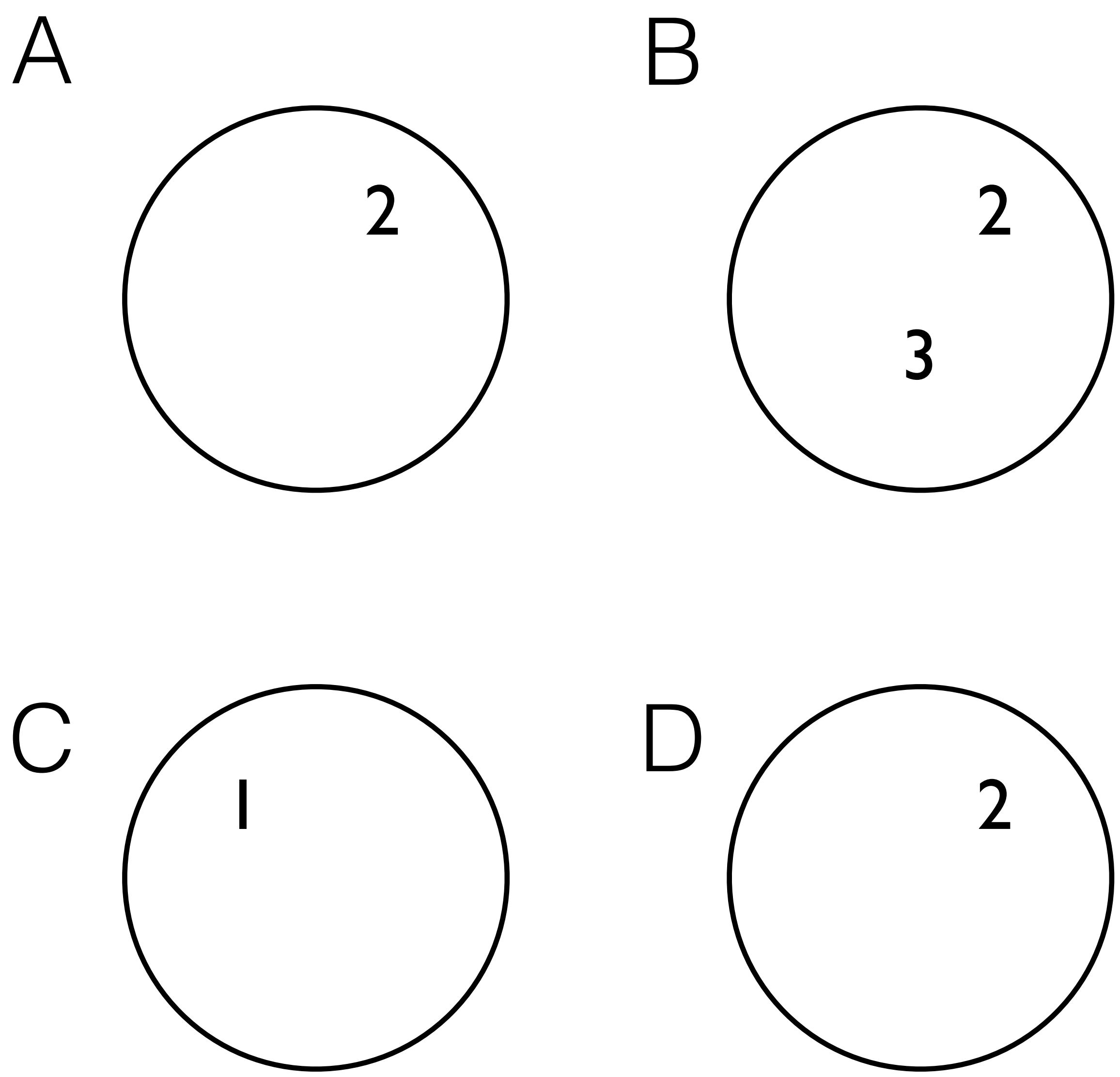
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

$$B + D = 4$$



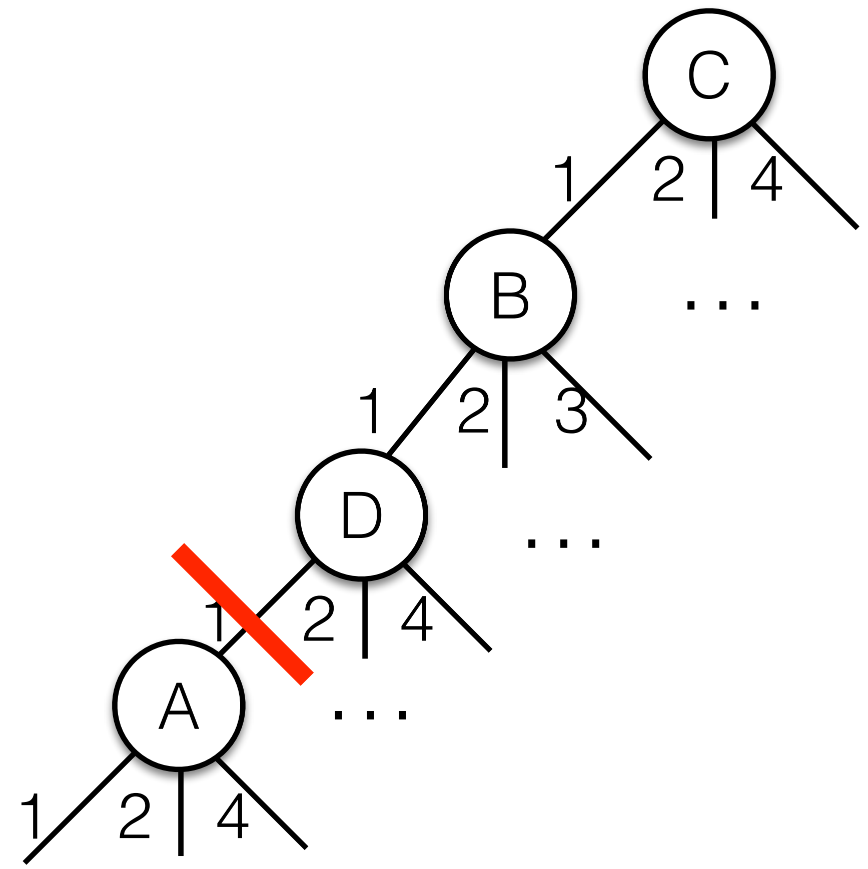
Constraint Satisfaction Problem



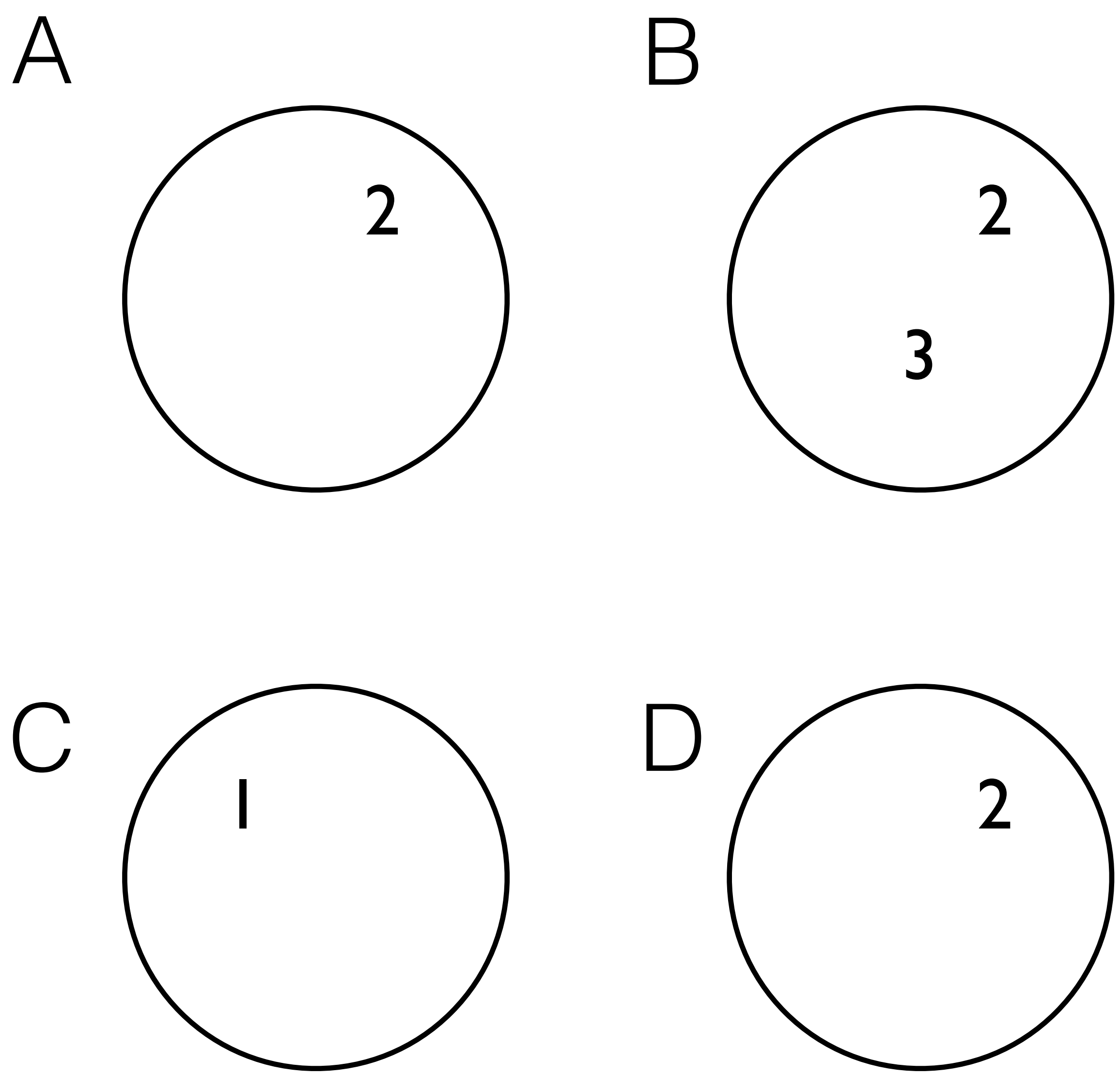
$A < B$ $\quad$ $\boxed{A \neq D}$

$A \neq C$ $\quad$ $C \neq D$

$B + D = 4$



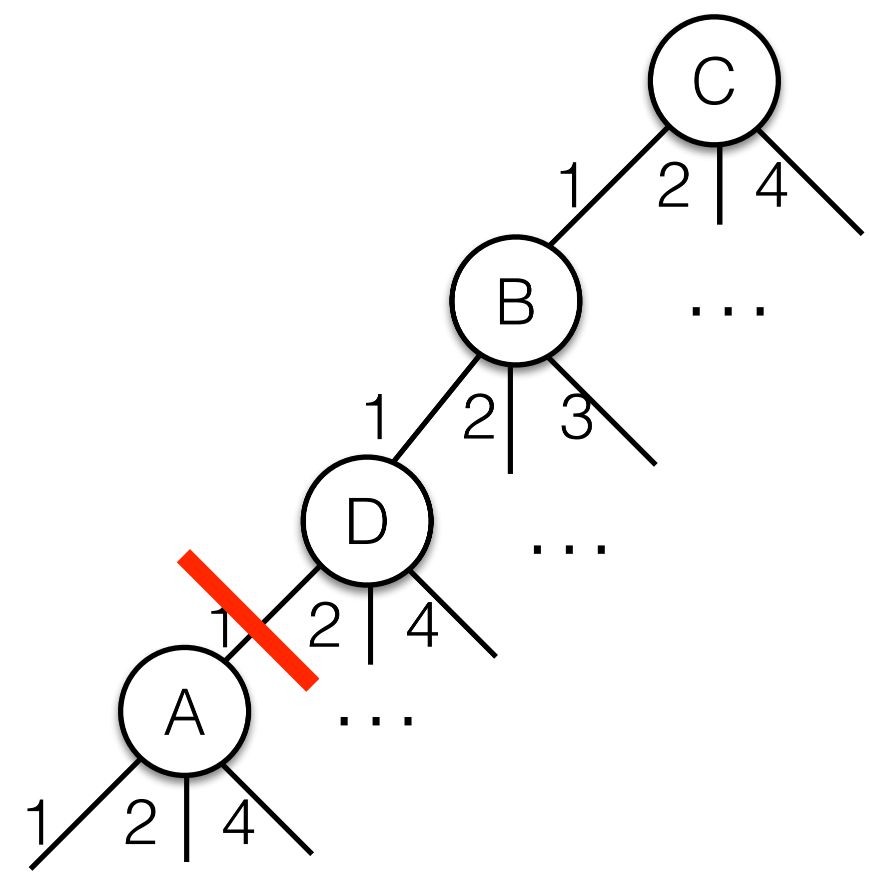
Constraint Satisfaction Problem



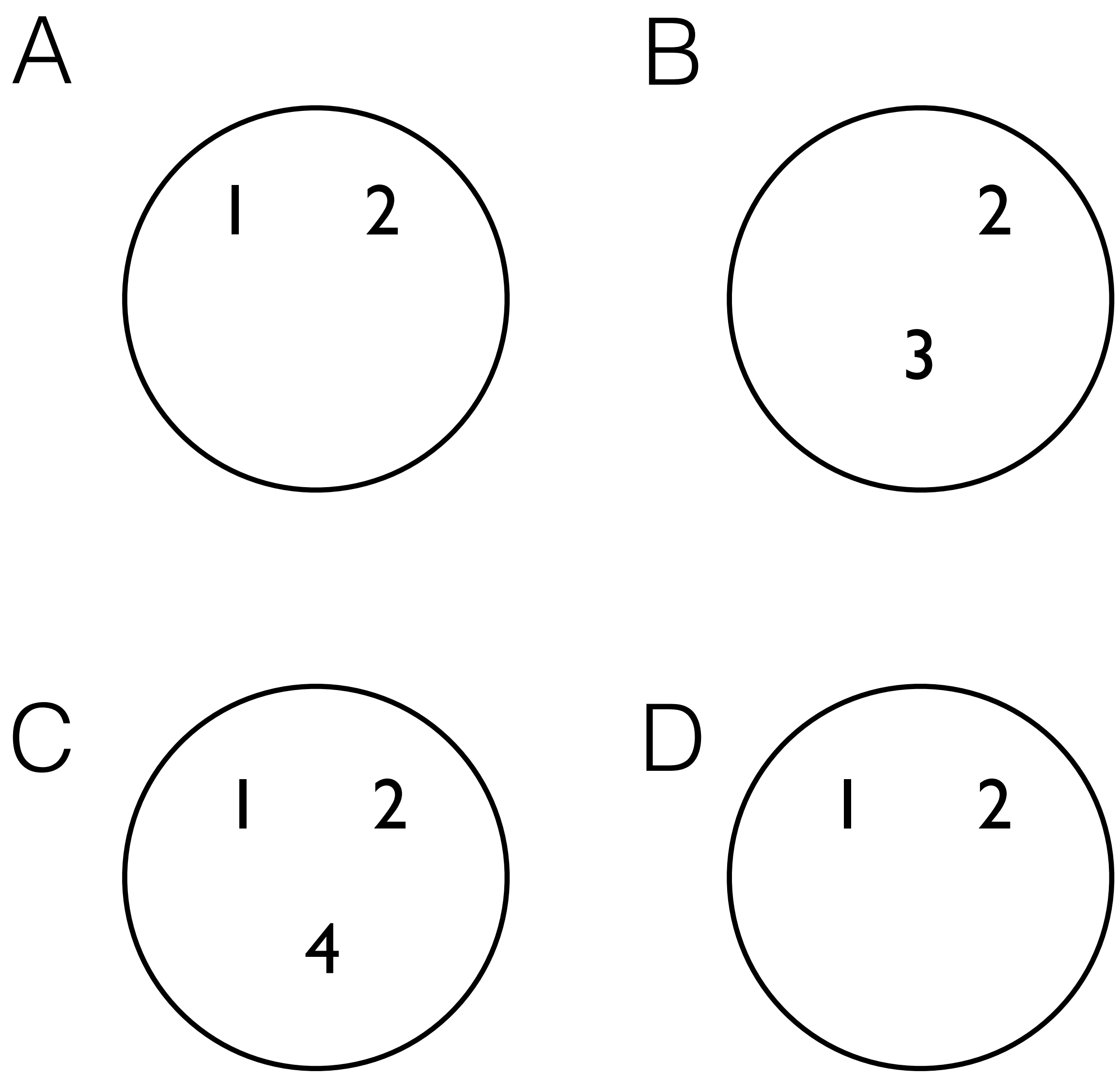
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

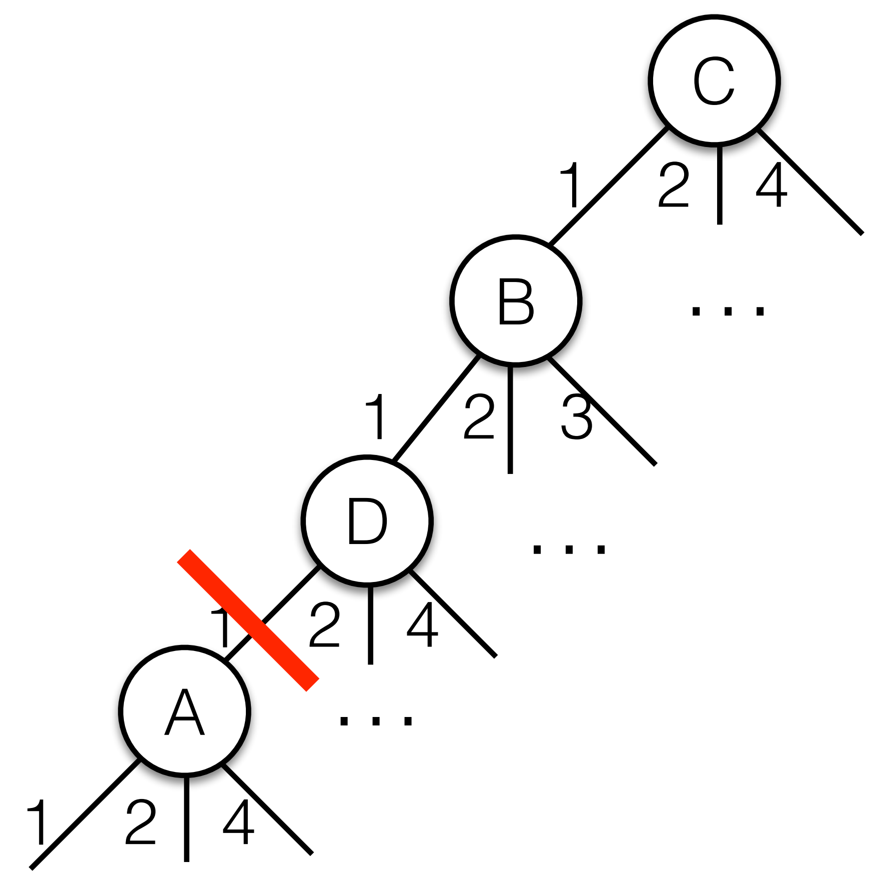
$$B + D = 4$$



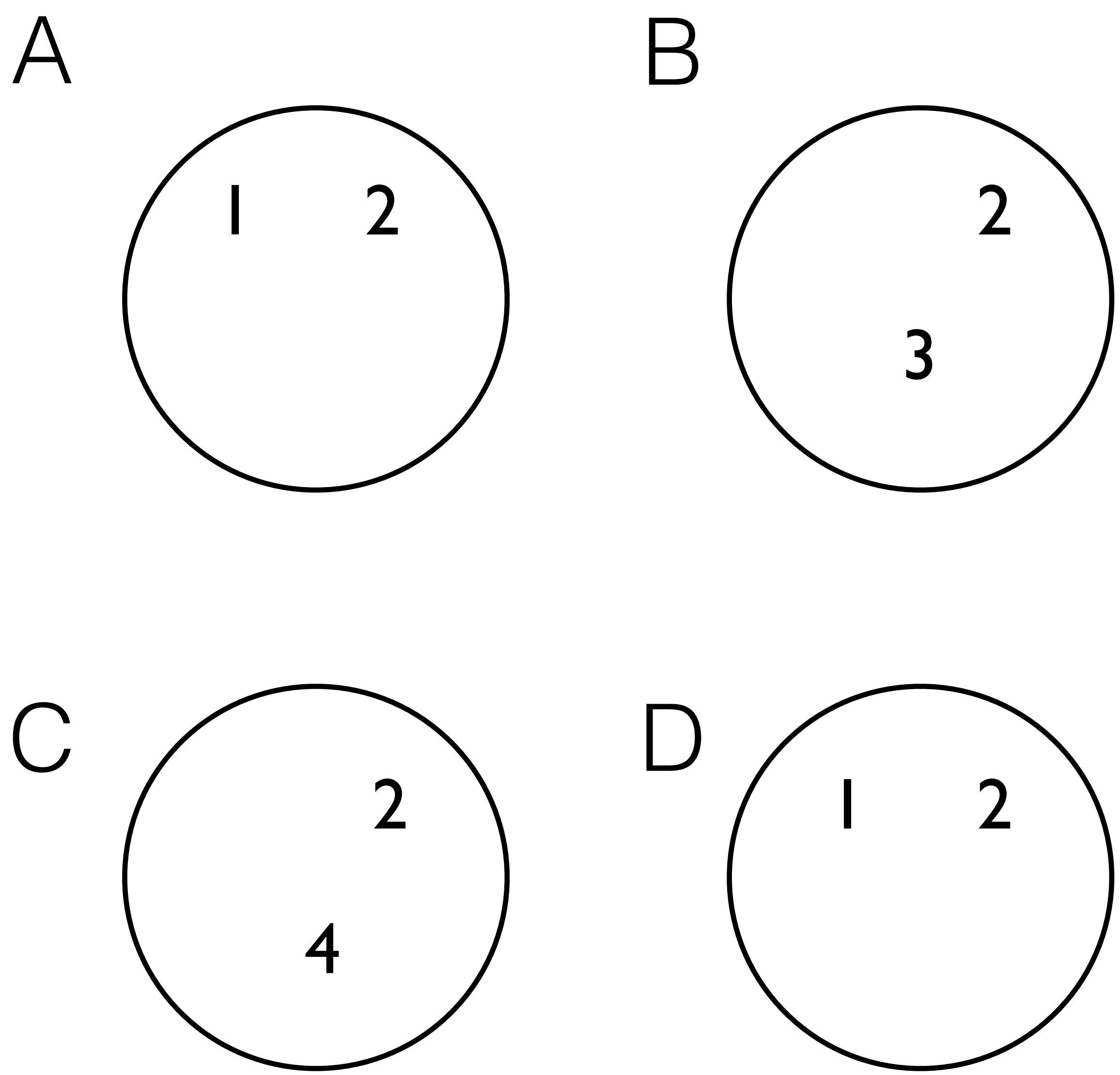
Constraint Satisfaction Problem



$$A < B \qquad A \neq D$$
$$A \neq C \qquad C \neq D$$
$$B + D = 4$$



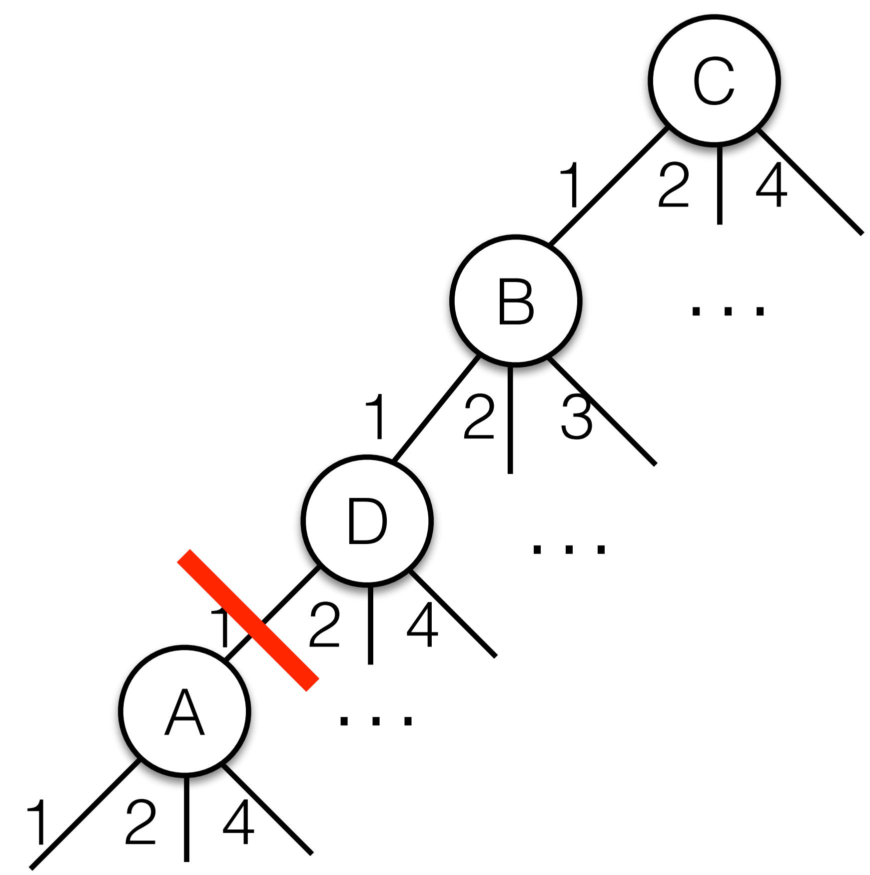
Constraint Satisfaction Problem



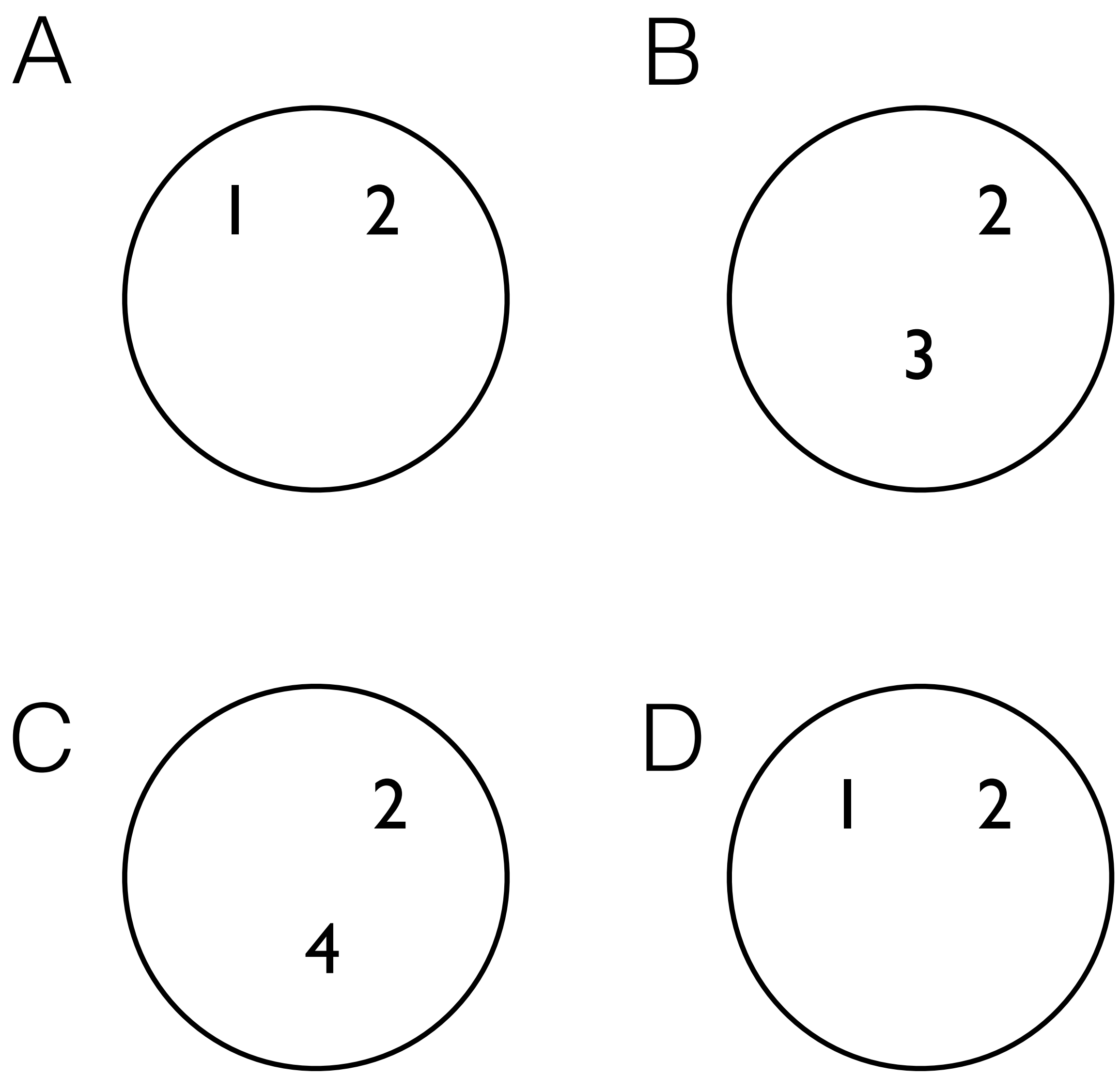
$$A < B \qquad A \neq D$$

$$A \neq C \qquad C \neq D$$

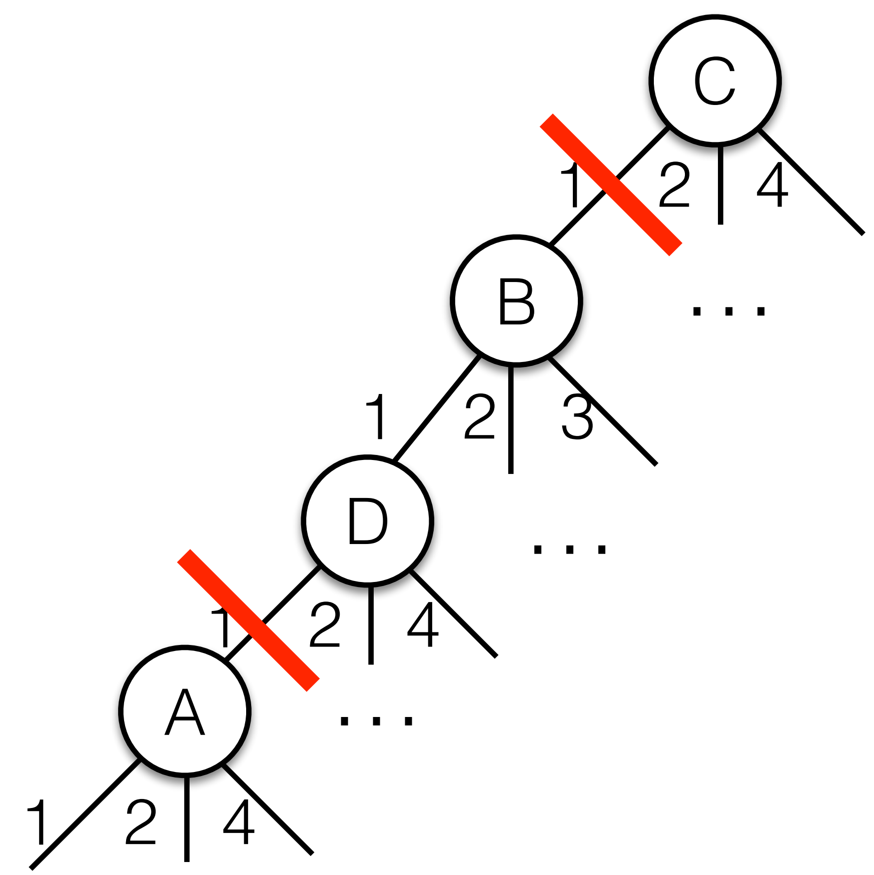
$$B + D = 4$$



Constraint Satisfaction Problem



$$A < B \qquad A \neq D$$
$$A \neq C \qquad C \neq D$$
$$B + D = 4$$



Branch and Bound

- For a minimization problem, the objective is encoded with a variable X.

- Upon finding a solution with objective value v, the solver imposes the constraint $X < v$.

- Filtering algorithms filter the lower bound of the domain of X.

- Filtering algorithms can be quite complex involving Lagrangian relaxation and specialized algorithms.

$\neq$ vs AllDifferent

$$\mathrm{dom}(A) = \{1, 2\}$$
$$\mathrm{dom}(B) = \{1, 2\}$$
$$\mathrm{dom}(C) = \{1, 2, 3\}$$

$\neq$ vs AllDifferent

$$\mathrm{dom}(A) = \{1, 2\}$$
$$\mathrm{dom}(B) = \{1, 2\}$$
$$\mathrm{dom}(C) = \{1, 2, 3\}$$

First model: Use elementary constraints

$$A \neq B, A \neq C, B \neq C$$

$\neq$ vs AllDifferent

$$\mathrm{dom}(A) = \{1, 2\}$$
$$\mathrm{dom}(B) = \{1, 2\}$$
$$\mathrm{dom}(C) = \{1, 2, 3\}$$

First model: Use elementary constraints

$$A \neq B, A \neq C, B \neq C$$

Second model: Use a global constraint

$$\textsc{All-Different}(A, B, C)$$

$\neq$ vs AllDifferent

$$\mathrm{dom}(A) = \{1, 2\}$$
$$\mathrm{dom}(B) = \{1, 2\}$$
$$\mathrm{dom}(C) = \{\cancel{1}, \cancel{2}, 3\}$$

First model: Use elementary constraints

$$A \neq B, A \neq C, B \neq C$$

Second model: Use a global constraint

$$\textsc{All-Different}(A, B, C)$$

Hall Interval

$$\begin{aligned} \mathrm{dom}(X_1) &= [3,4] \\ \mathrm{dom}(X_2) &= [3,4] \\ \mathrm{dom}(X_3) &= [1,3] \end{aligned}$$

Hall Interval

$$\left.\begin{aligned} \mathrm{dom}(X_1) &= [3,4] \\ \mathrm{dom}(X_2) &= [3,4] \end{aligned}\right\} \quad \text{Hall interval}$$

$$\mathrm{dom}(X_3) = [1,3]$$

- A Hall interval is an in an interval of *k* consecutive values that contain the domains of *k* variables.

Hall Interval

$$\left.\begin{array}{rcl} \mathrm{dom}(X_1) & = & [3,4] \\ \mathrm{dom}(X_2) & = & [3,4] \end{array}\right\} \quad \text{Hall interval}$$
$$\begin{array}{rcl} \mathrm{dom}(X_3) & = & [1,2] \end{array}$$

- A Hall interval is an in an interval of *k* consecutive values that contain the domains of *k* variables.

Filtering Algorithm

$$\begin{aligned} \mathrm{dom}(X_1) &= [2, 3] \\ \mathrm{dom}(X_2) &= [2, 3] \\ \mathrm{dom}(X_3) &= [3, 4] \\ \mathrm{dom}(X_4) &= [2, 6] \end{aligned}$$

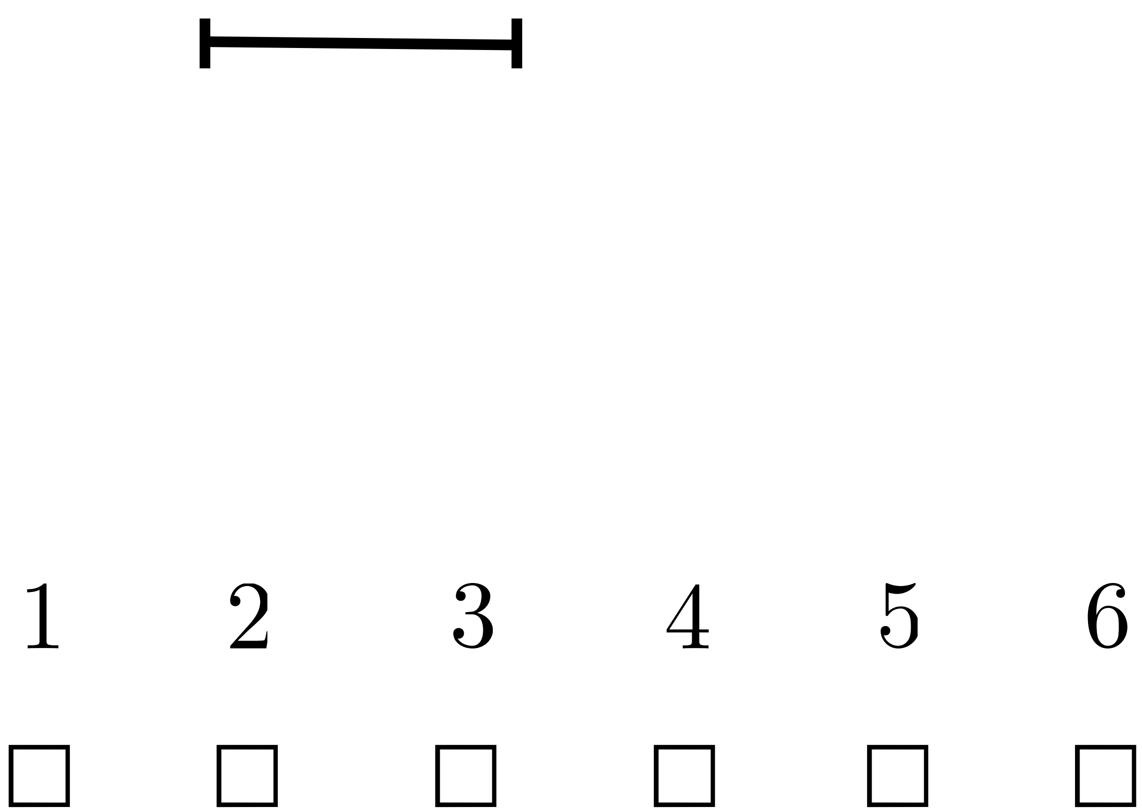
1	2	3	4	5	6
$\square$	$\square$	$\square$	$\square$	$\square$	$\square$

Filtering Algorithm

$$\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [3,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}$$

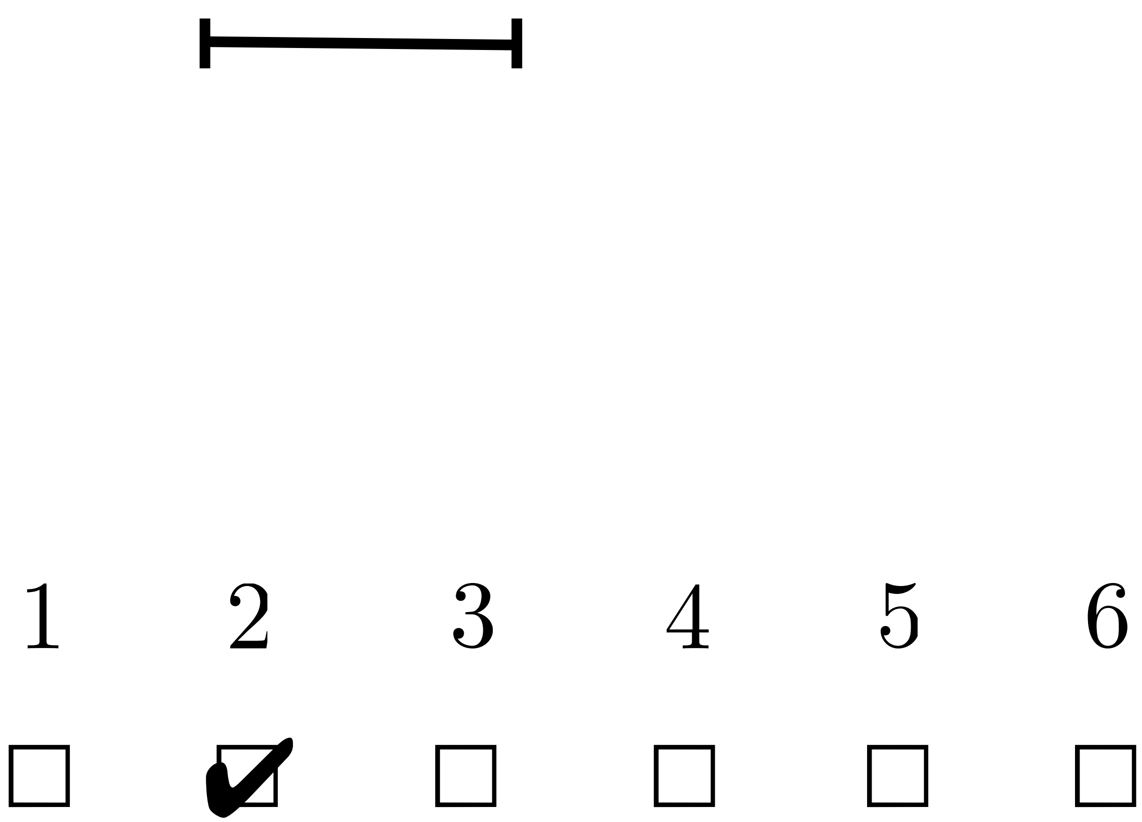
1	2	3	4	5	6
$\square$	$\square$	$\square$	$\square$	$\square$	$\square$

Filtering Algorithm



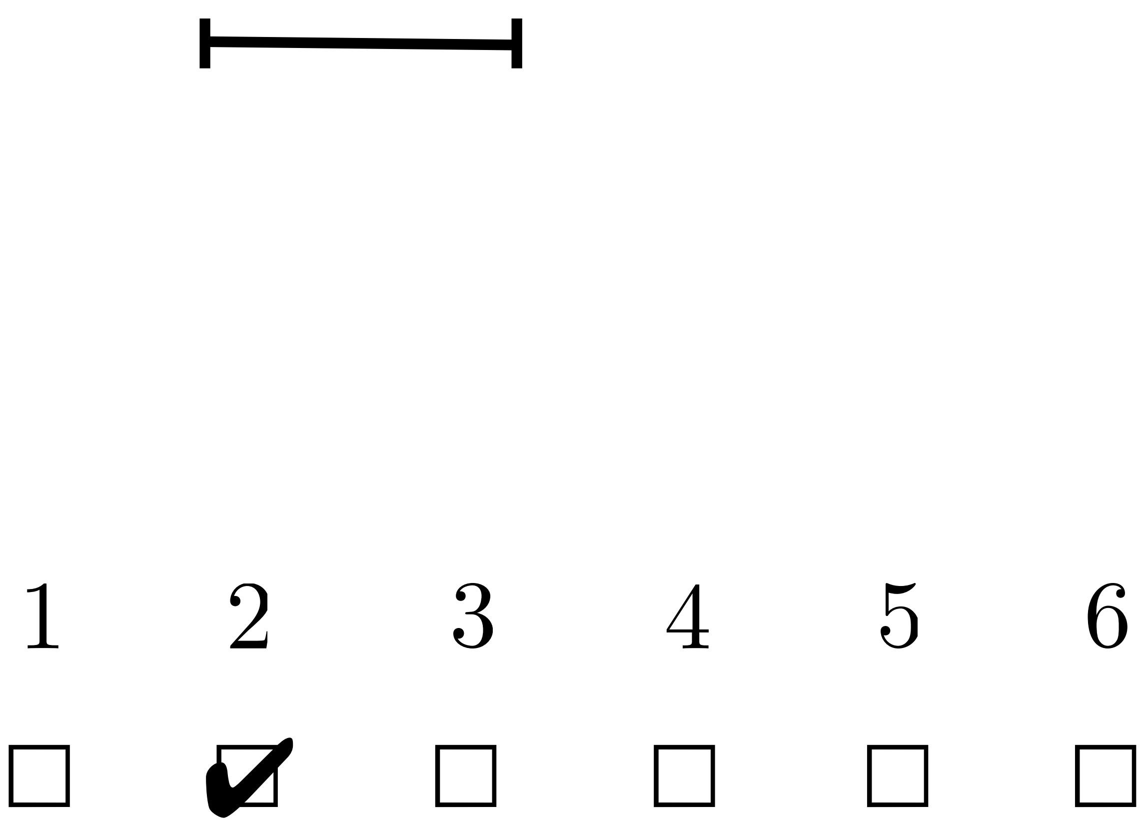
$$\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [3,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}$$

Filtering Algorithm



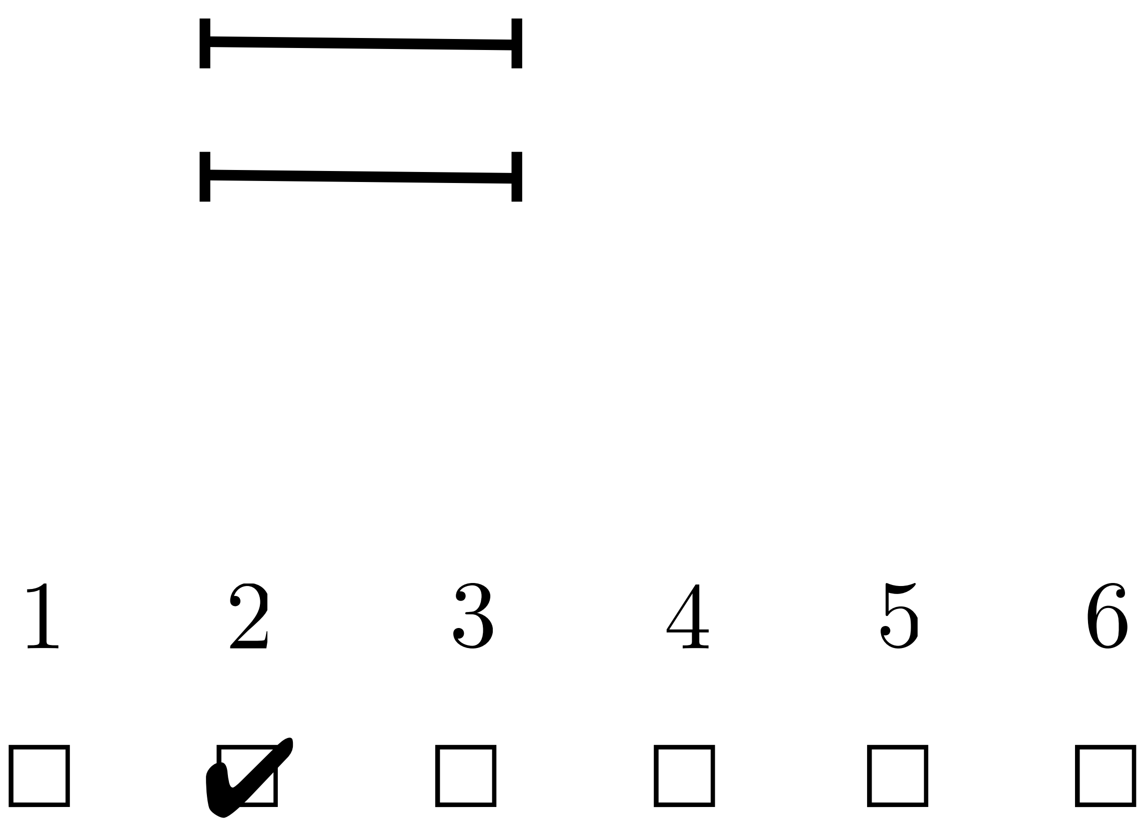
$$\begin{aligned}
\mathrm{dom}(X_1) &= [2, 3] \\
\mathrm{dom}(X_2) &= [2, 3] \\
\mathrm{dom}(X_3) &= [3, 4] \\
\mathrm{dom}(X_4) &= [2, 6]
\end{aligned}$$

Filtering Algorithm



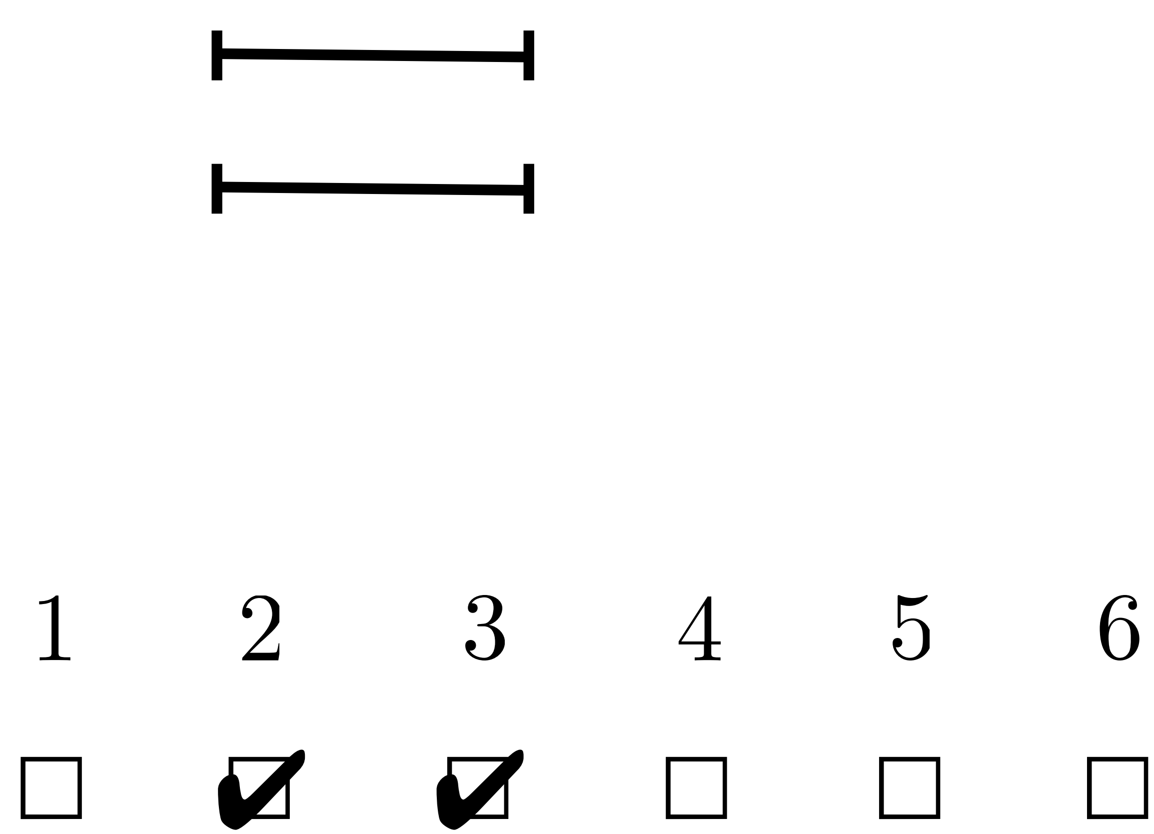
$$\begin{aligned} \mathrm{dom}(X_1) &= [2,3] \\ \mathrm{dom}(X_2) &= [2,3] \\ \mathrm{dom}(X_3) &= [3,4] \\ \mathrm{dom}(X_4) &= [2,6] \end{aligned}$$

Filtering Algorithm



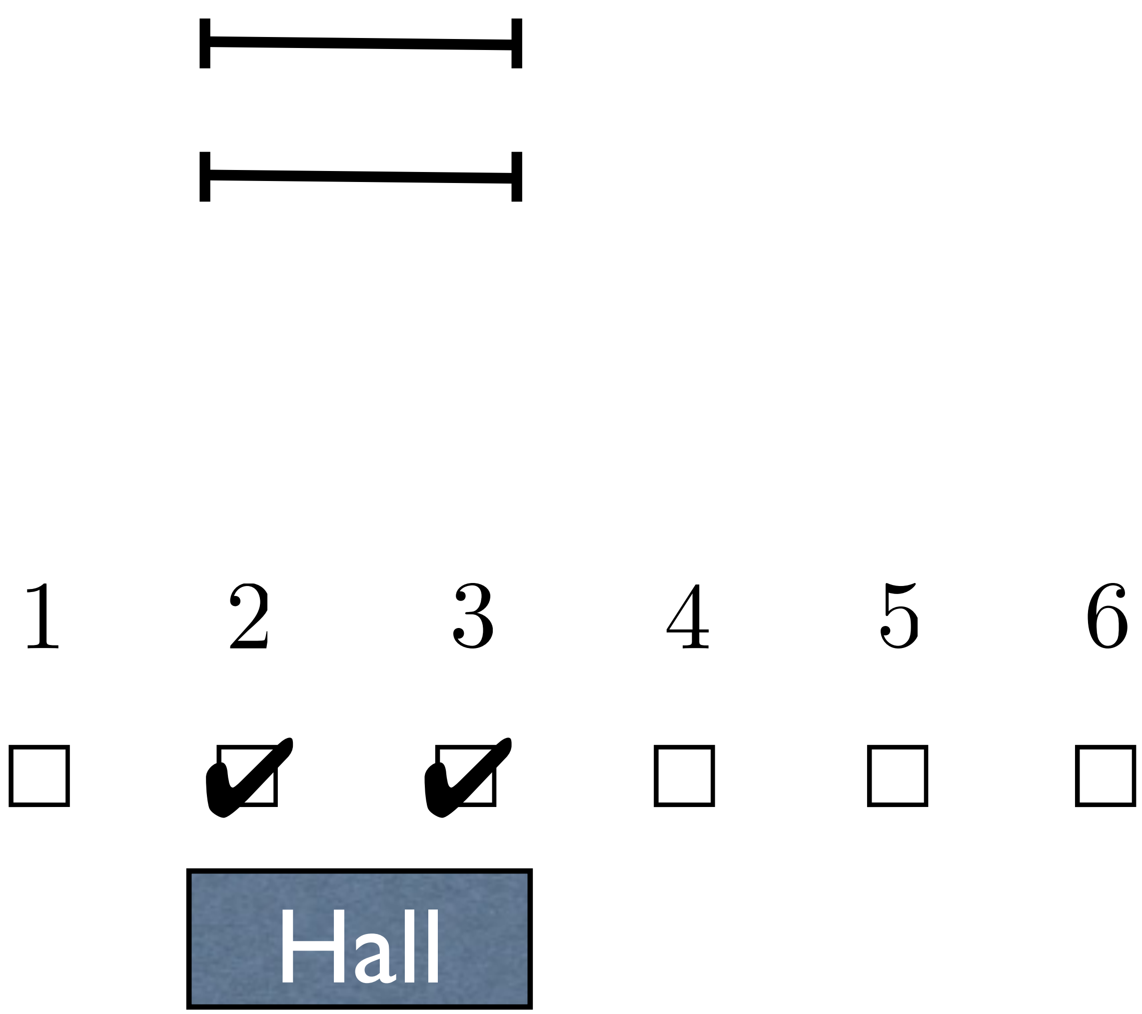
$$\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [3,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}$$

Filtering Algorithm



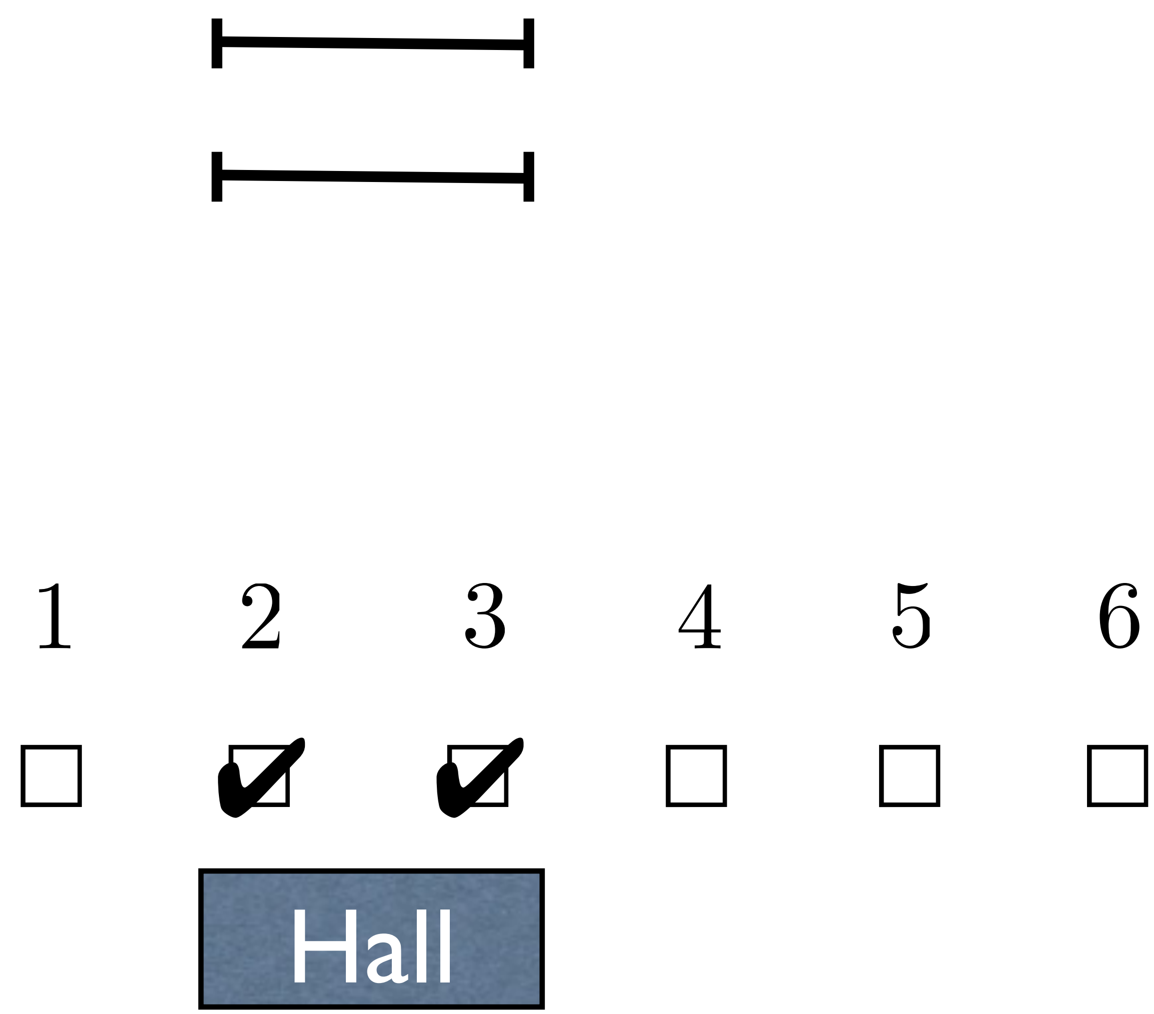
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3]\\
\mathrm{dom}(X_2) &= [2,3]\\
\mathrm{dom}(X_3) &= [3,4]\\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



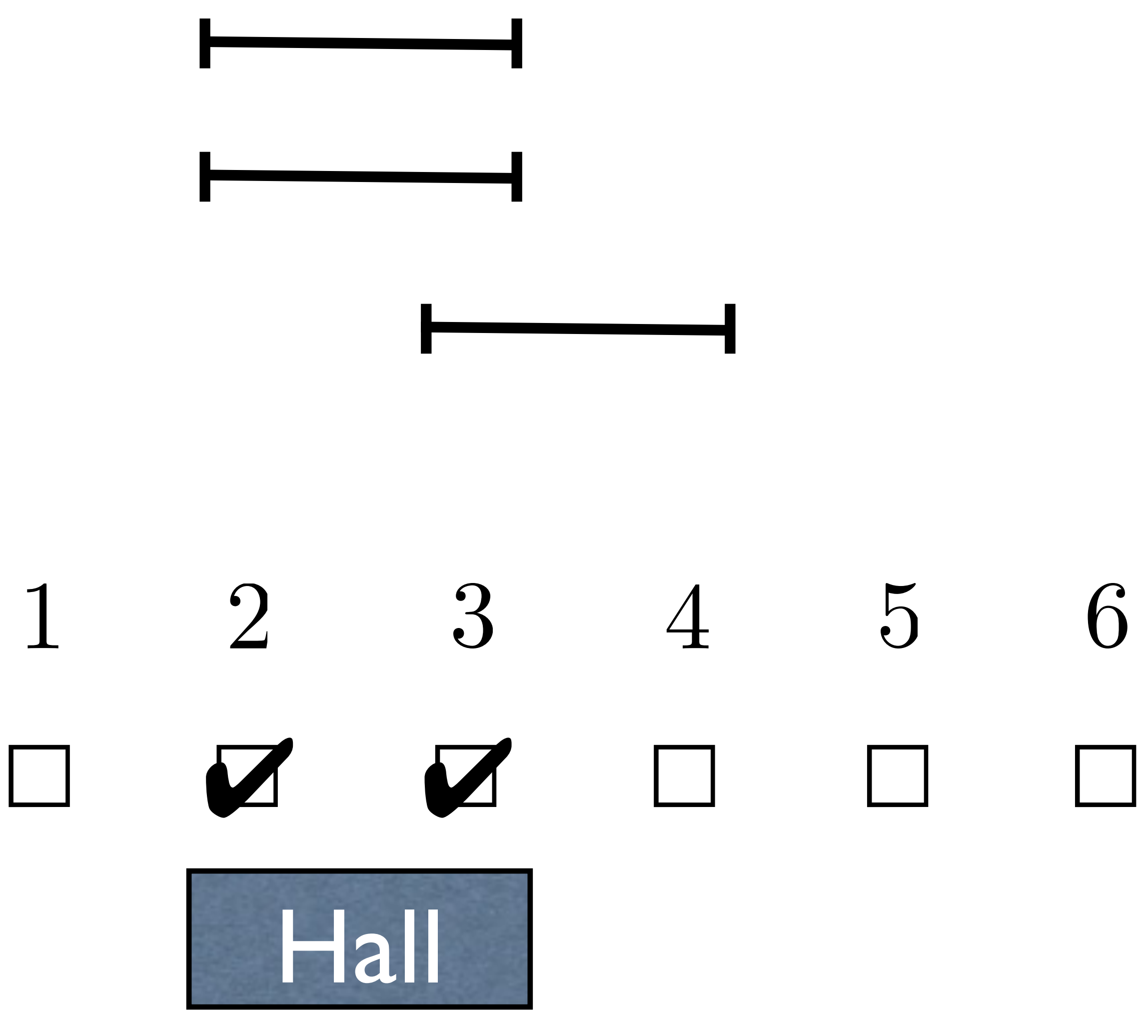
$$\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [3,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}$$

Filtering Algorithm



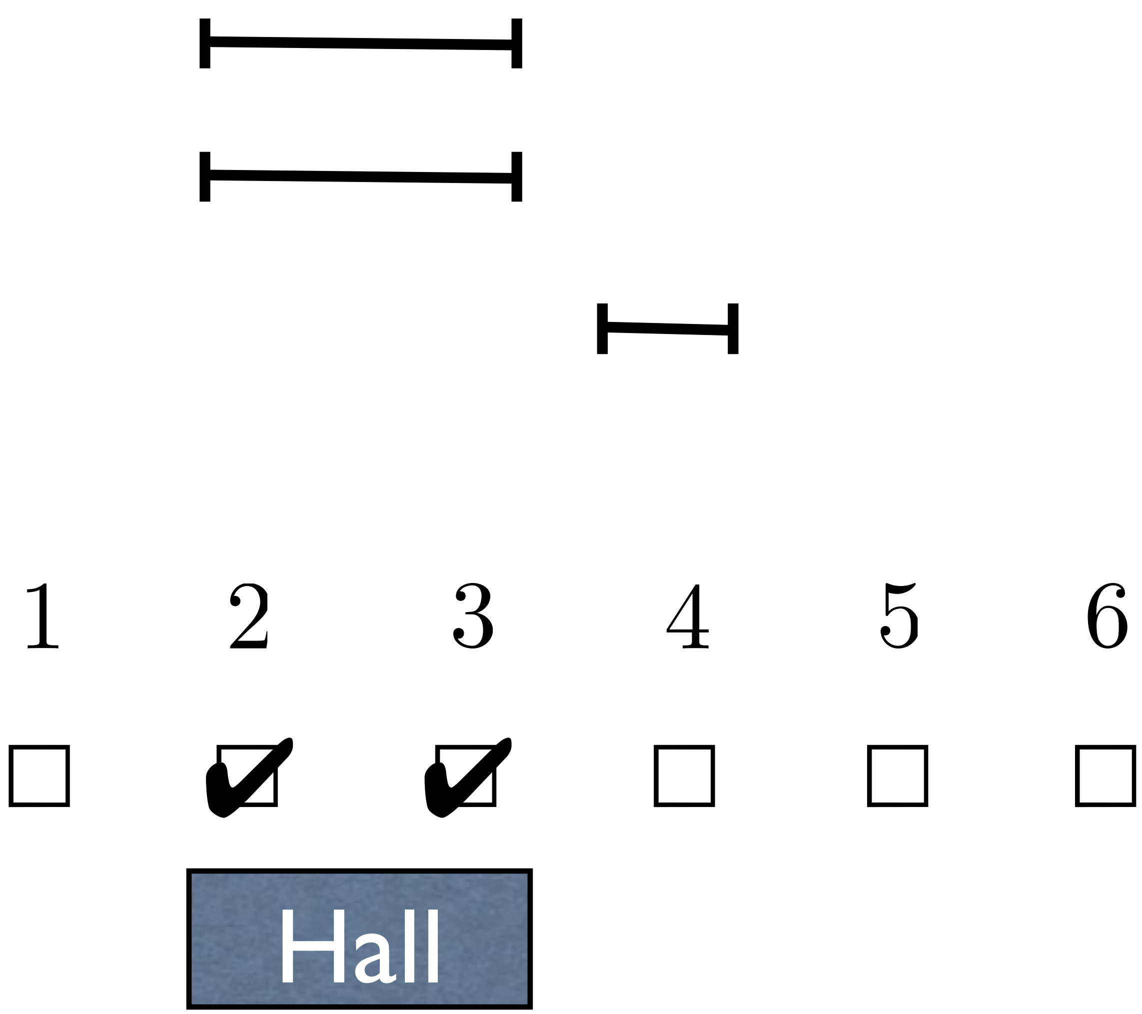
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [3,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



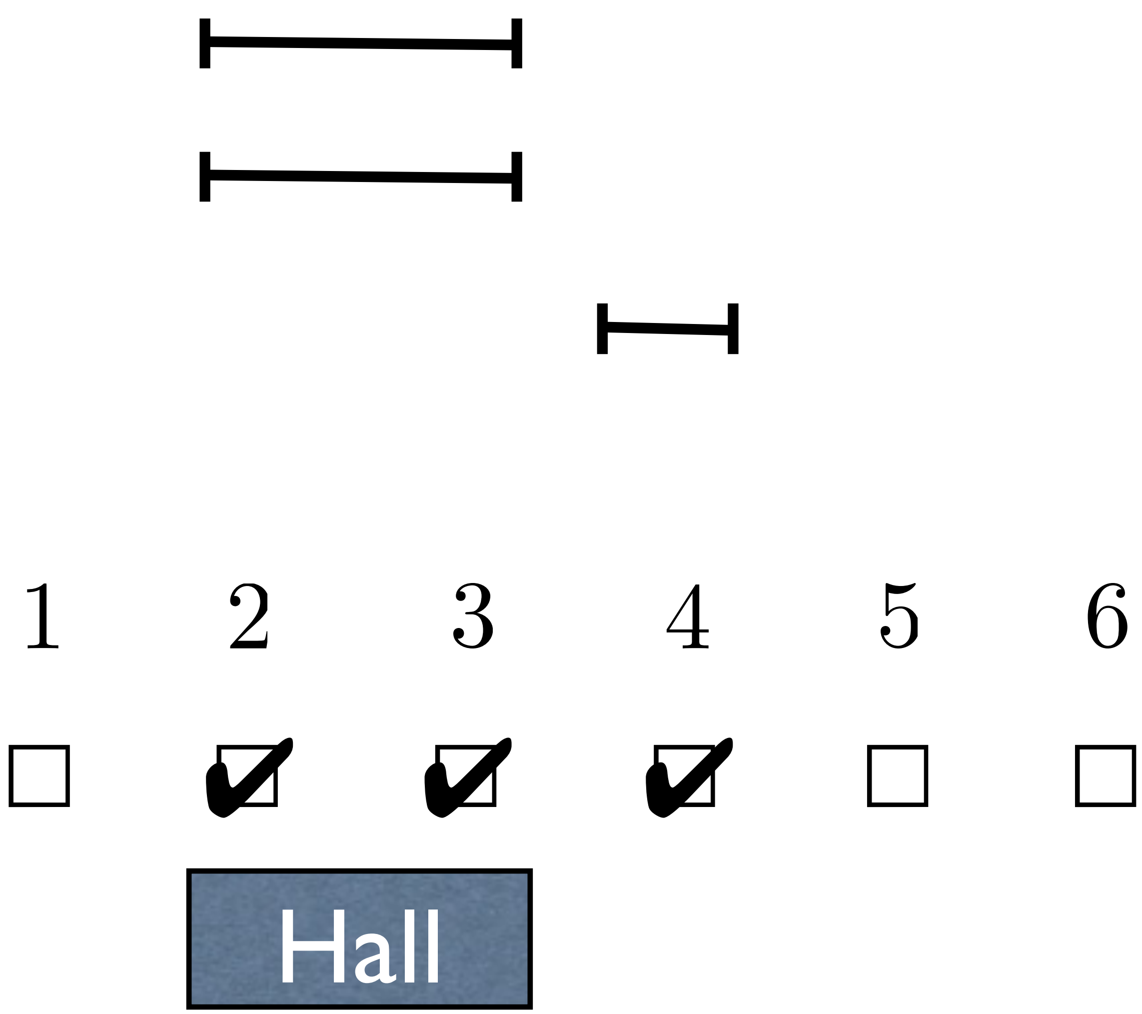
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3]\\
\mathrm{dom}(X_2) &= [2,3]\\
\mathrm{dom}(X_3) &= [3,4]\\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



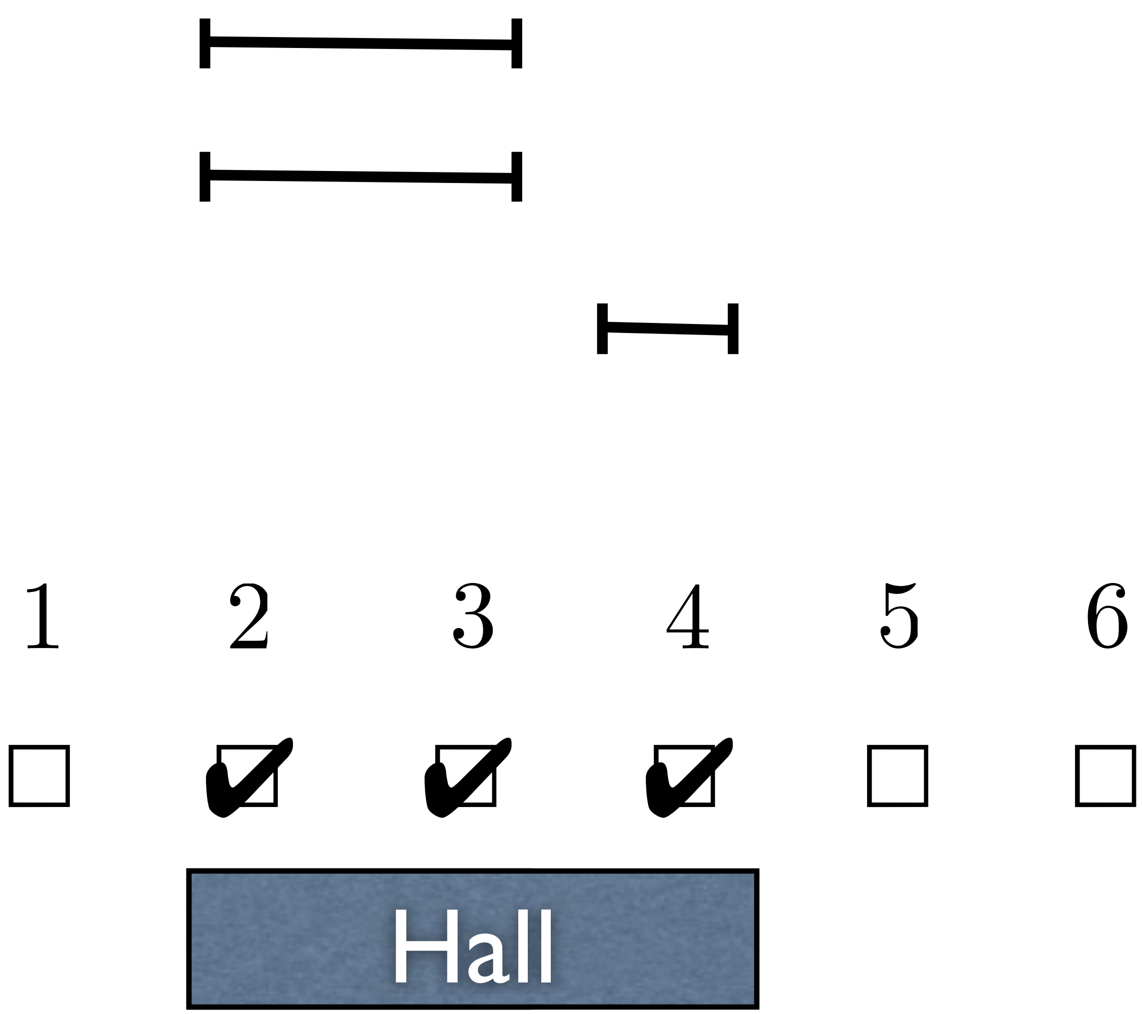
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [4,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



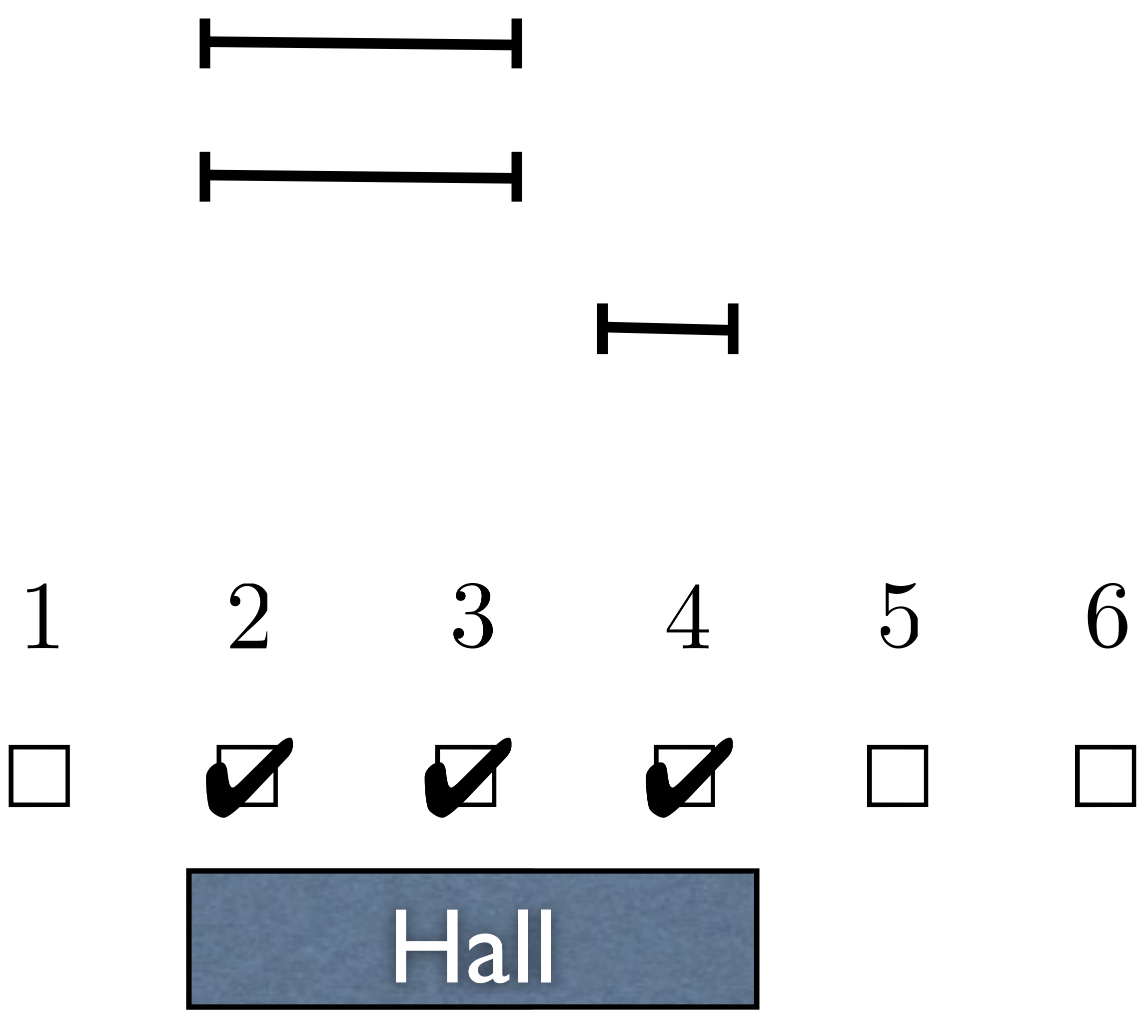
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [4,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



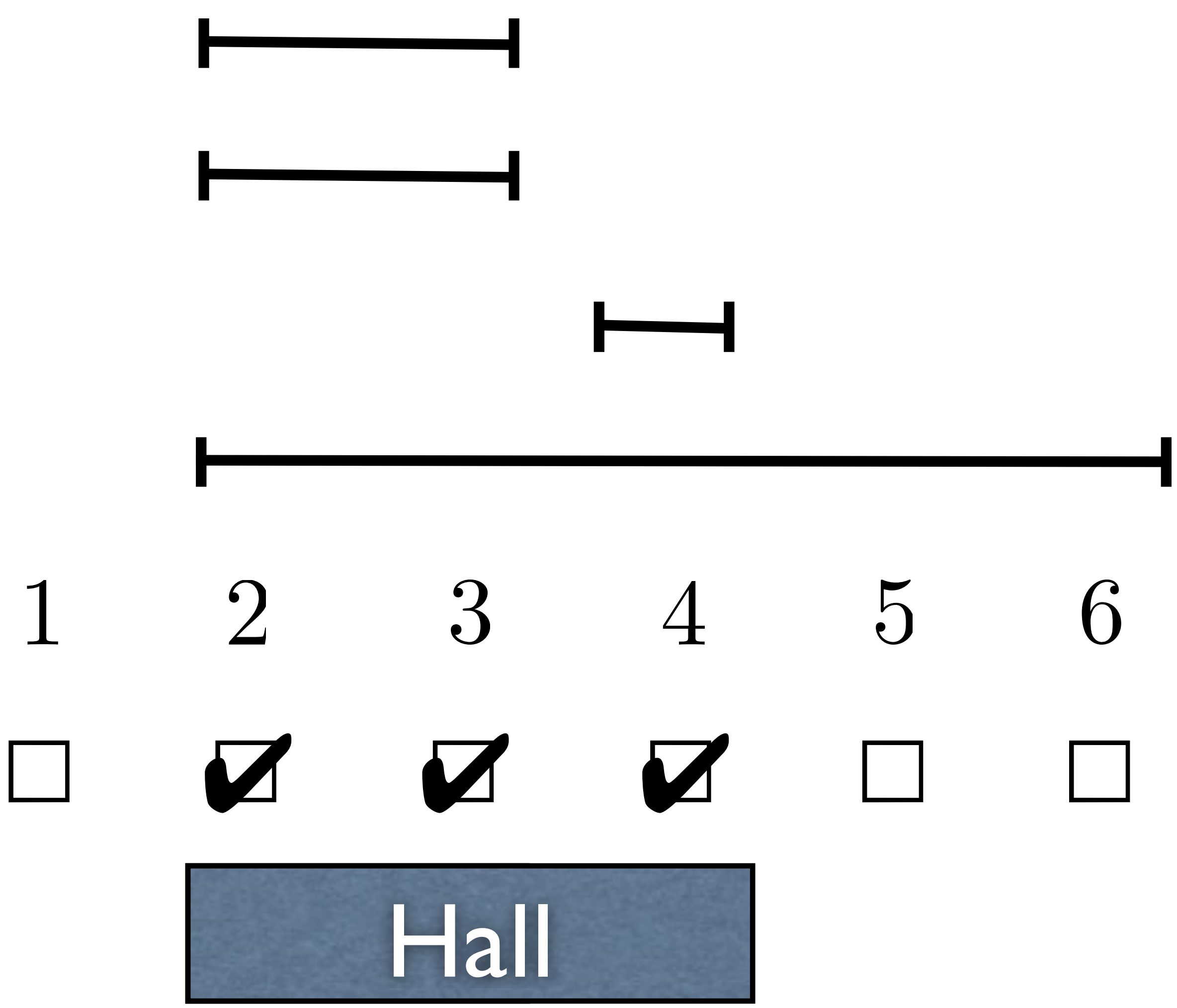
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [4,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



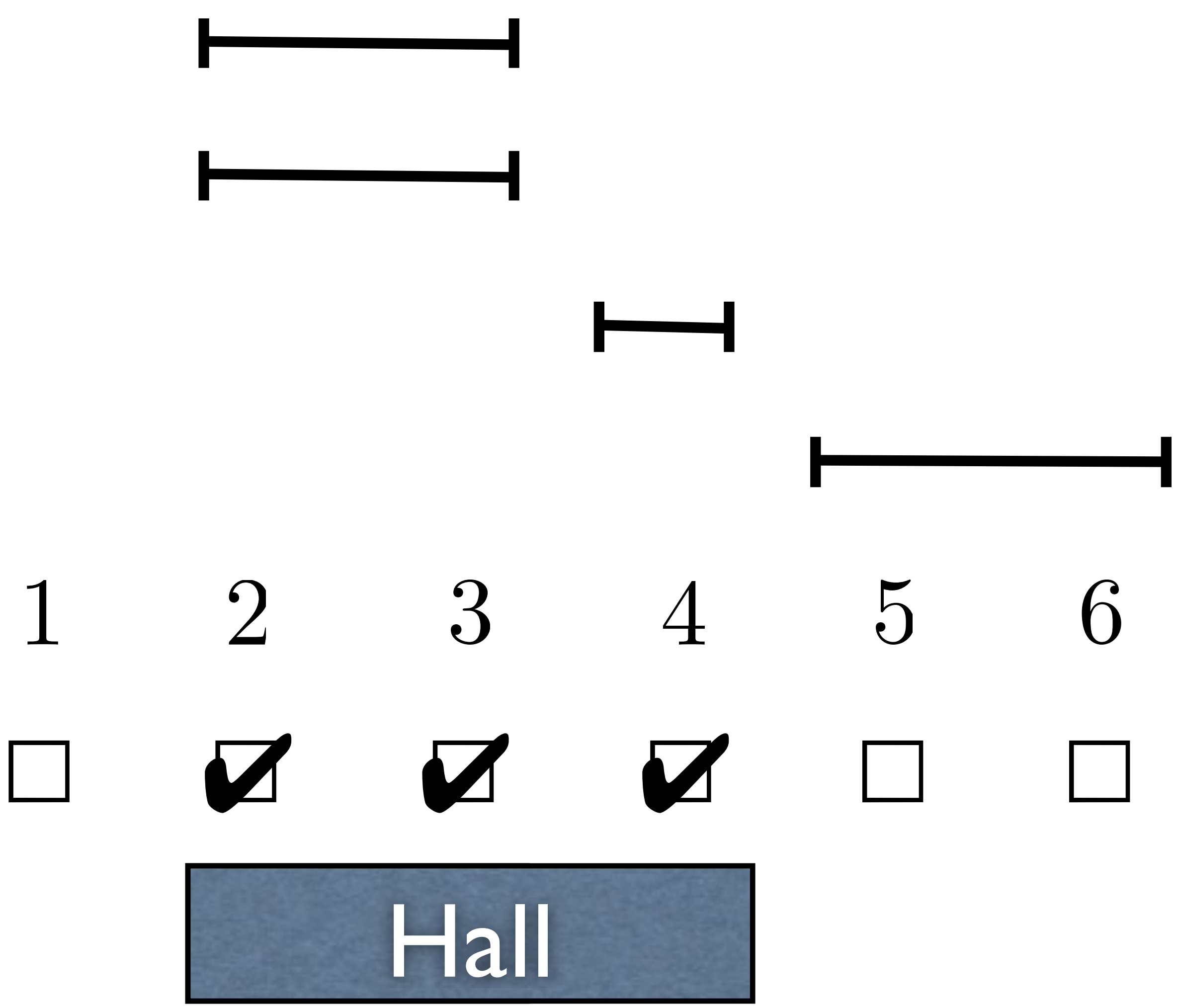
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [4,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



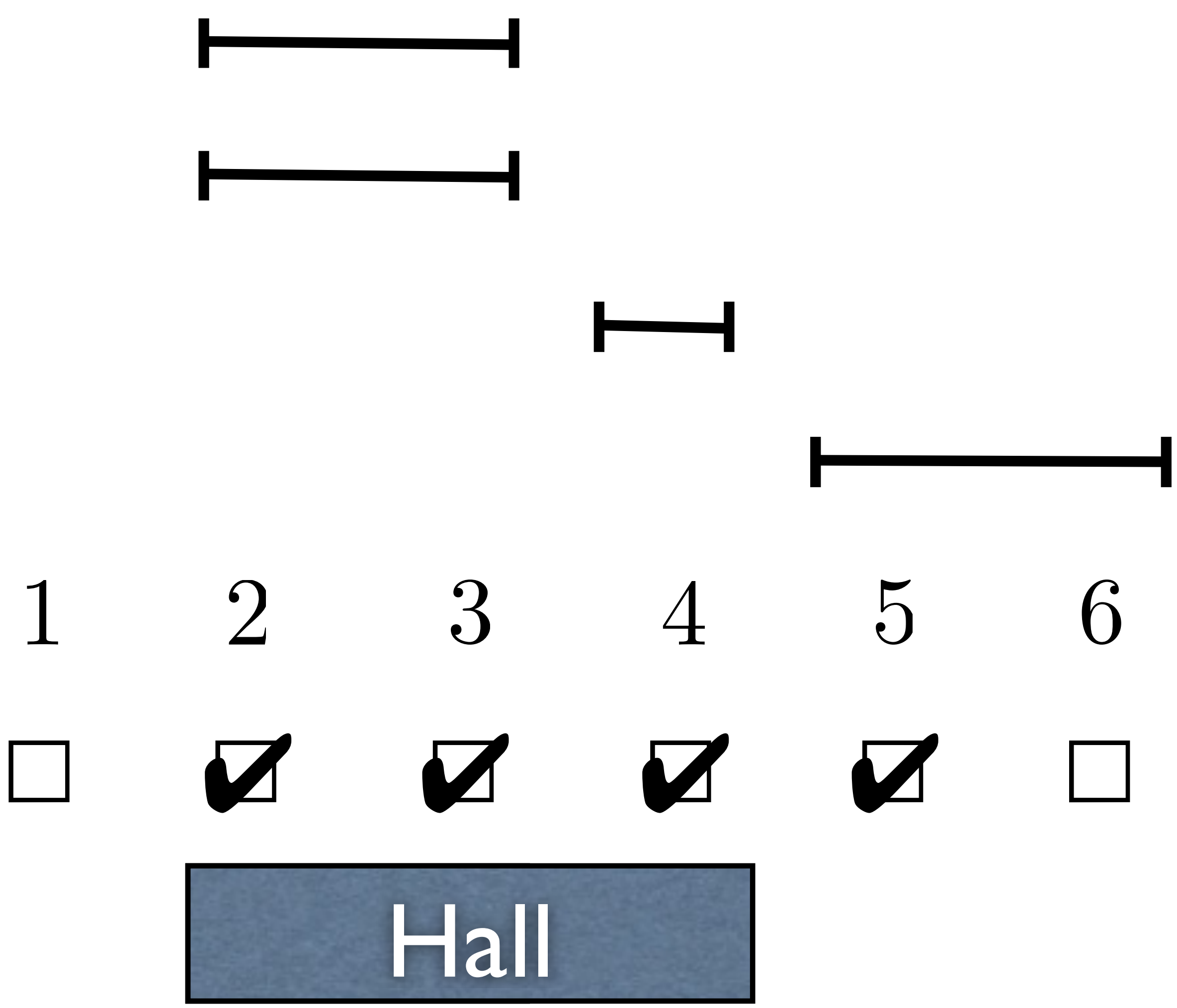
$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2,3] \\
\mathrm{dom}(X_2) &= [2,3] \\
\mathrm{dom}(X_3) &= [4,4] \\
\mathrm{dom}(X_4) &= [2,6]
\end{aligned}
$$

Filtering Algorithm



$$
\begin{aligned}
\mathrm{dom}(X_1) &= [2, 3] \\
\mathrm{dom}(X_2) &= [2, 3] \\
\mathrm{dom}(X_3) &= [4, 4] \\
\mathrm{dom}(X_4) &= [5, 6]
\end{aligned}
$$

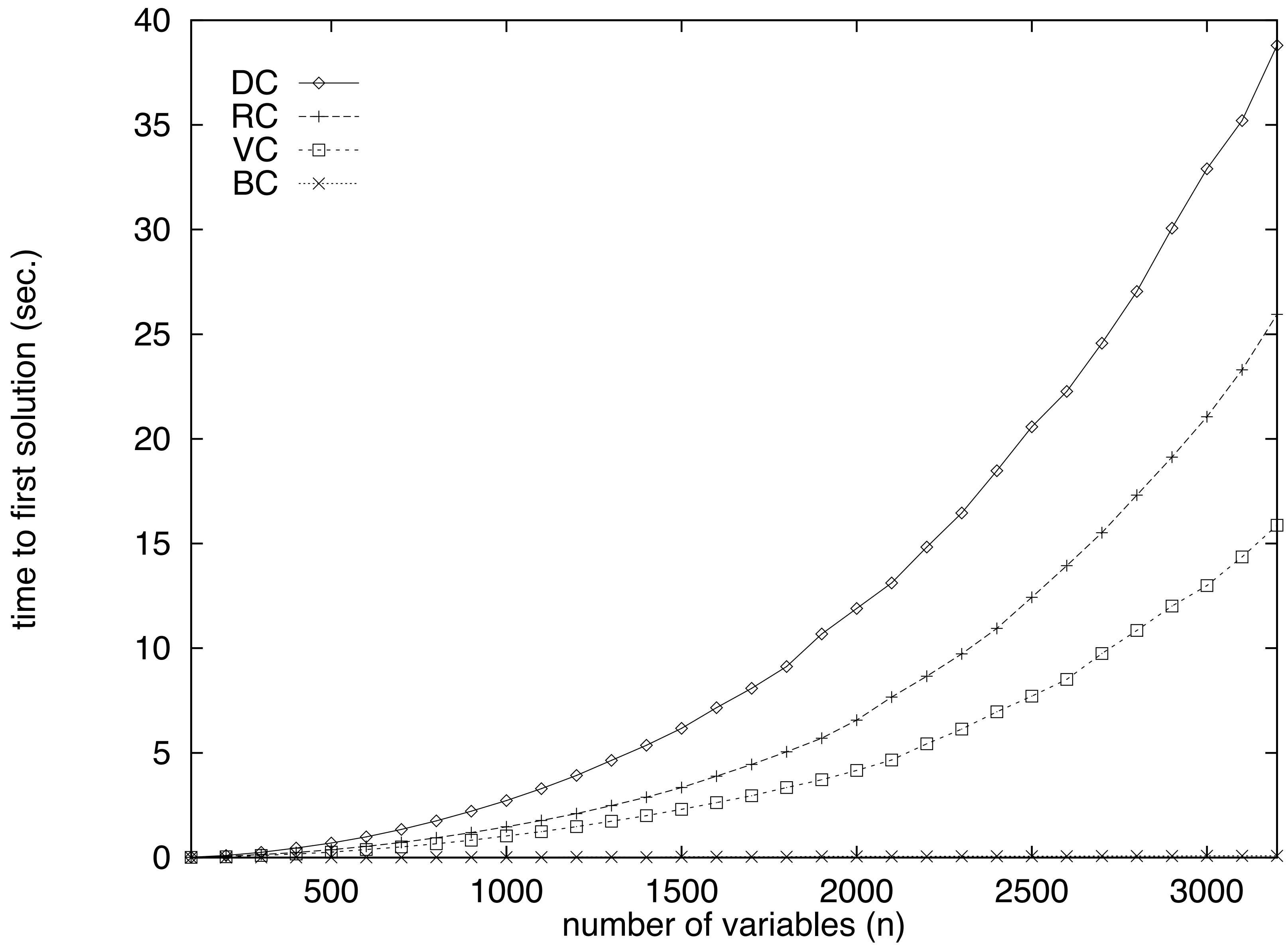
Filtering Algorithm



$$\begin{aligned}\mathrm{dom}(X_1) &= [2,3]\\ \mathrm{dom}(X_2) &= [2,3]\\ \mathrm{dom}(X_3) &= [4,4]\\ \mathrm{dom}(X_4) &= [5,6]\end{aligned}$$

Execution Time

- Using the right data structure, the algorithm runs in linear time.



Global Constraints

- There exist 423 global constraints in the literature [Global Constraint Catalog, Beldiceanu et al.]
- In practice, it is only useful to know about a dozen of them.

Scheduling Constraints

- Some constraints are specifically designed for scheduling problems.

Scheduling Constraints

- Some constraints are specifically designed for scheduling problems.
- The Disjunctive constraint prevents tasks from overlapping.

$$\textsc{Disjunctive}([S_1, \ldots, S_n], [p_1, \ldots, p_n])$$
$$\iff$$
$$\forall i, j, \quad S_i + p_i \leq S_j \vee S_j + p_j \leq S_i$$

Scheduling Constraints

- Some constraints are specifically designed for scheduling problems.
- The Disjunctive constraint prevents tasks from overlapping.

$$\textsc{Disjunctive}([S_1, \ldots, S_n], [p_1, \ldots, p_n])$$
$$\iff$$
$$\forall i, j, \quad S_i + p_i \leq S_j \lor S_j + p_j \leq S_i$$

- The Cumulative constraint prevents too many tasks to execute simultaneously.

$$\textsc{Cumulative}([S_1, \ldots, S_n], [p_1, \ldots, p_n], [h_1, \ldots, h_n], C)$$
$$\iff$$
$$\forall t, \sum_{S_i \leq t \leq S_i + p_i} h_i \leq C$$

Filtering Scheduling Constraints

- Given variable domains, deciding whether DISJUNCTIVE or CUMULATIVE is satisfiable is NP-Complete.

Filtering Scheduling Constraints

- Given variable domains, deciding whether DISJUNCTIVE or CUMULATIVE is satisfiable is NP-Complete.
- Fully filtering these constraints is NP-Hard.

Filtering Scheduling Constraints

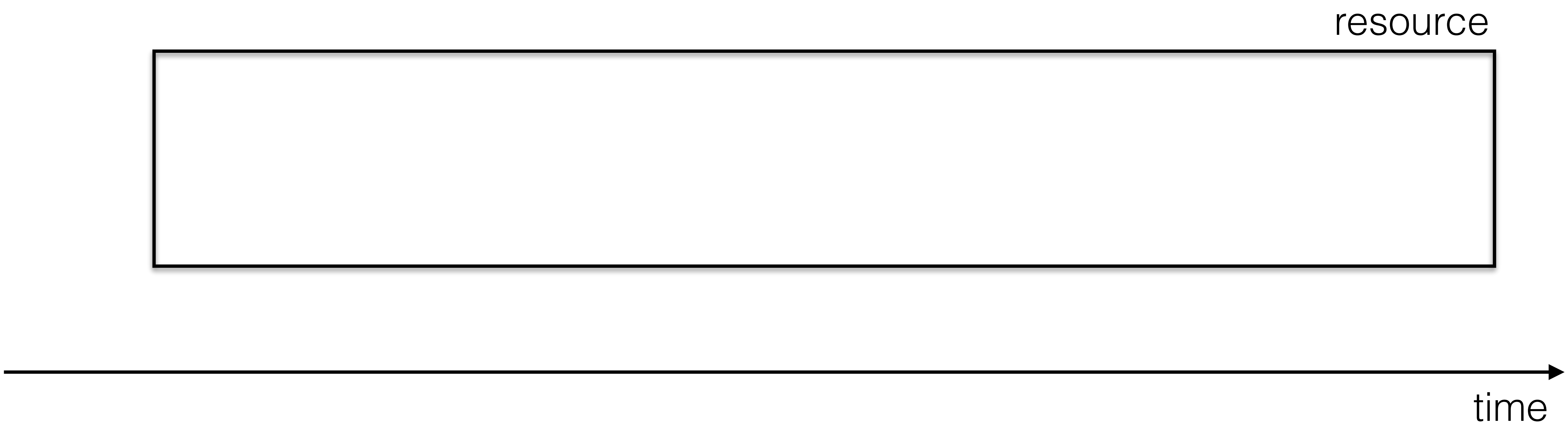
- Given variable domains, deciding whether DISJUNCTIVE or CUMULATIVE is satisfiable is NP-Complete.
- Fully filtering these constraints is NP-Hard.
- Constraint solvers use filtering rules that filter some values, but not all values.

Filtering Scheduling Constraints

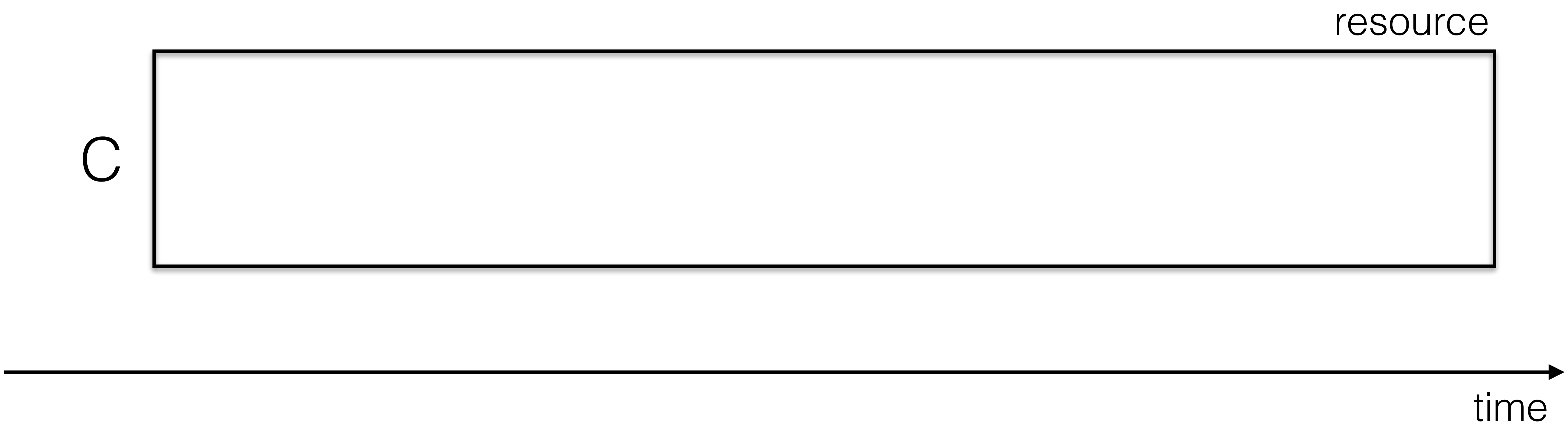
- Given variable domains, deciding whether DISJUNCTIVE or CUMULATIVE is satisfiable is NP-Complete.
- Fully filtering these constraints is NP-Hard.
- Constraint solvers use filtering rules that filter some values, but not all values.
- These rules are designed to offer a good trade-off between filtering power and computation time.

Definitions

Definitions

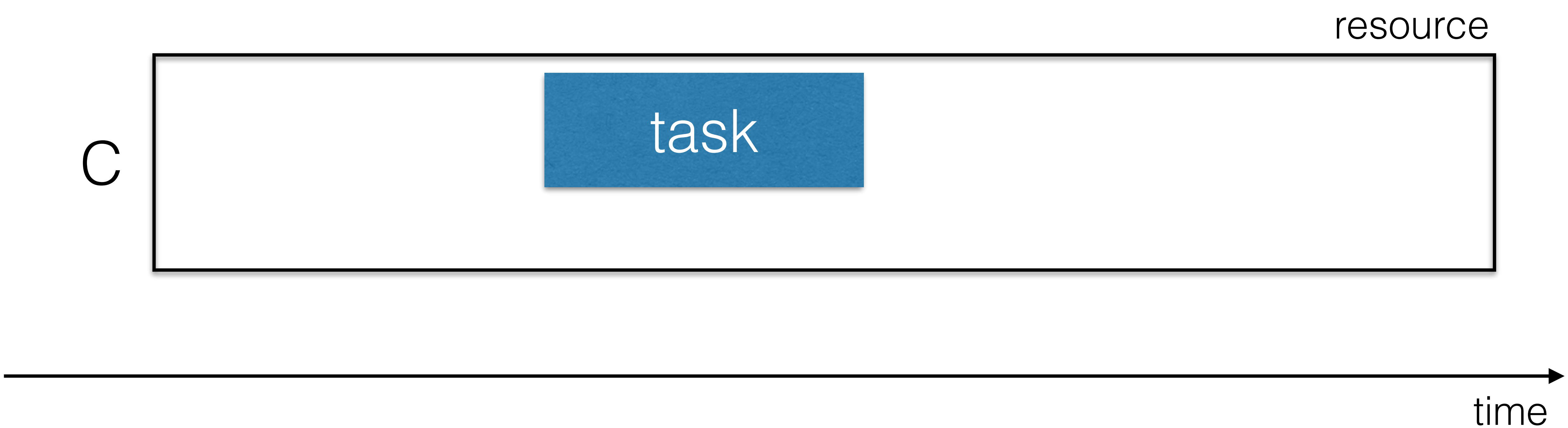


Definitions



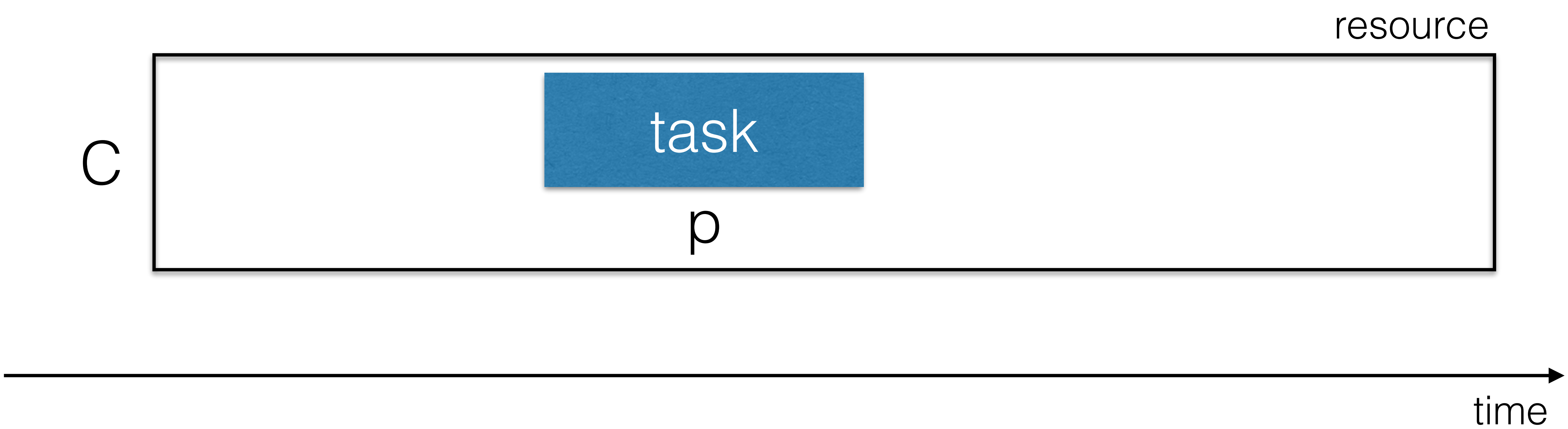
- C: resource capacity

Definitions



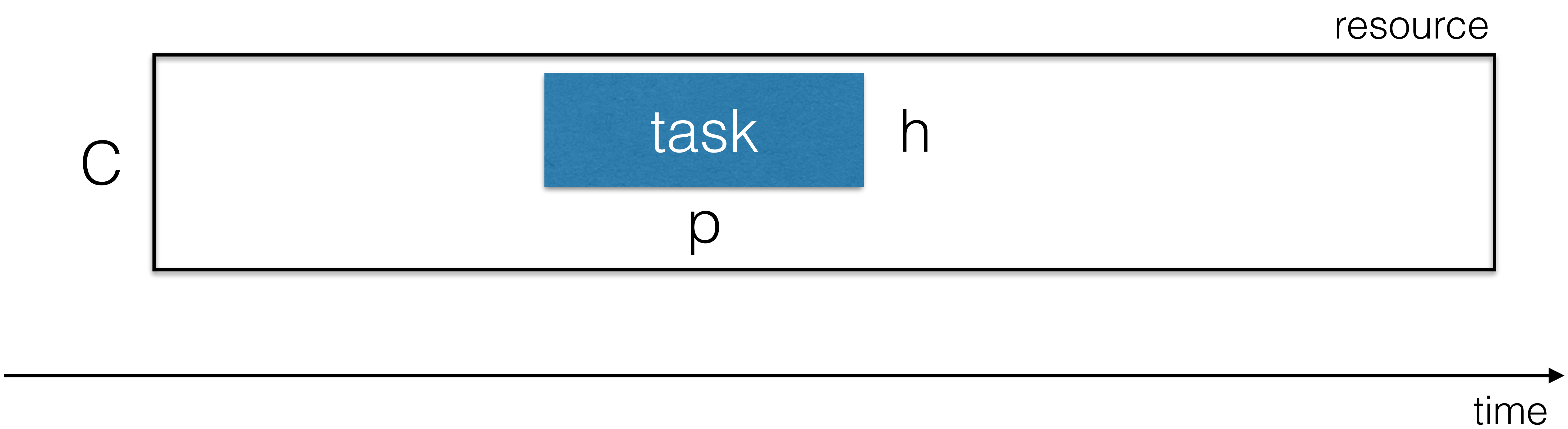
- C: resource capacity

Definitions



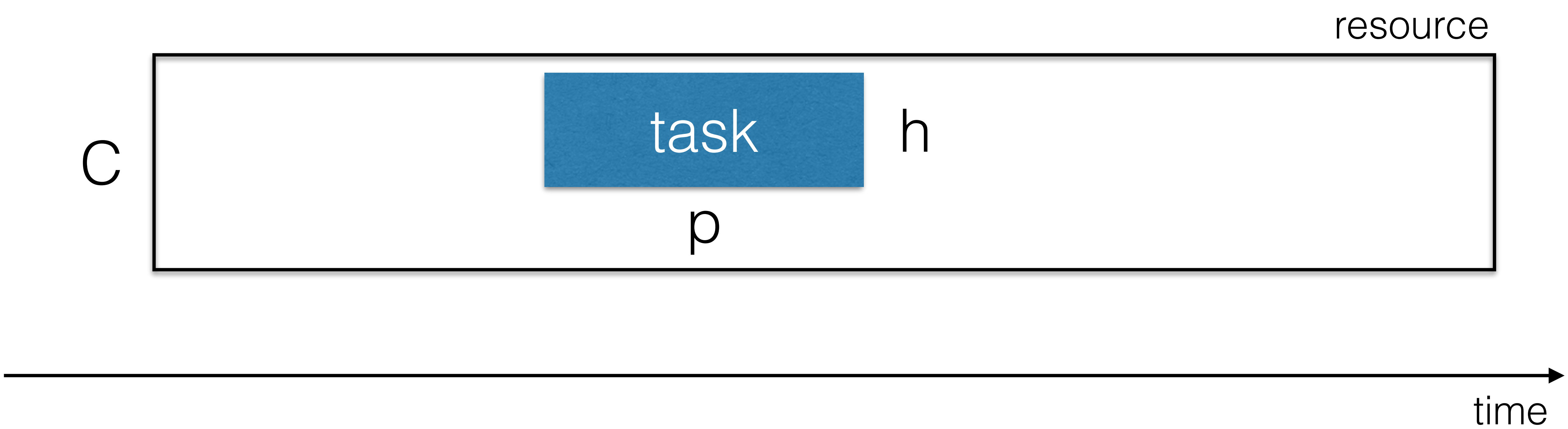
- C: resource capacity
- p: processing time

Definitions



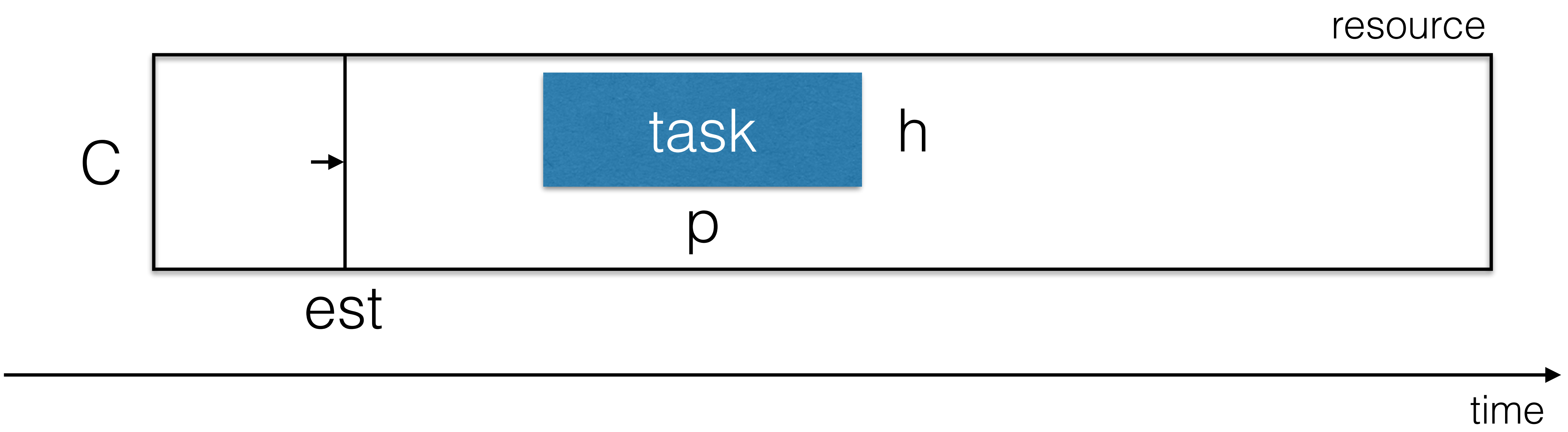
- C: resource capacity
- p: processing time
- h: height

Definitions



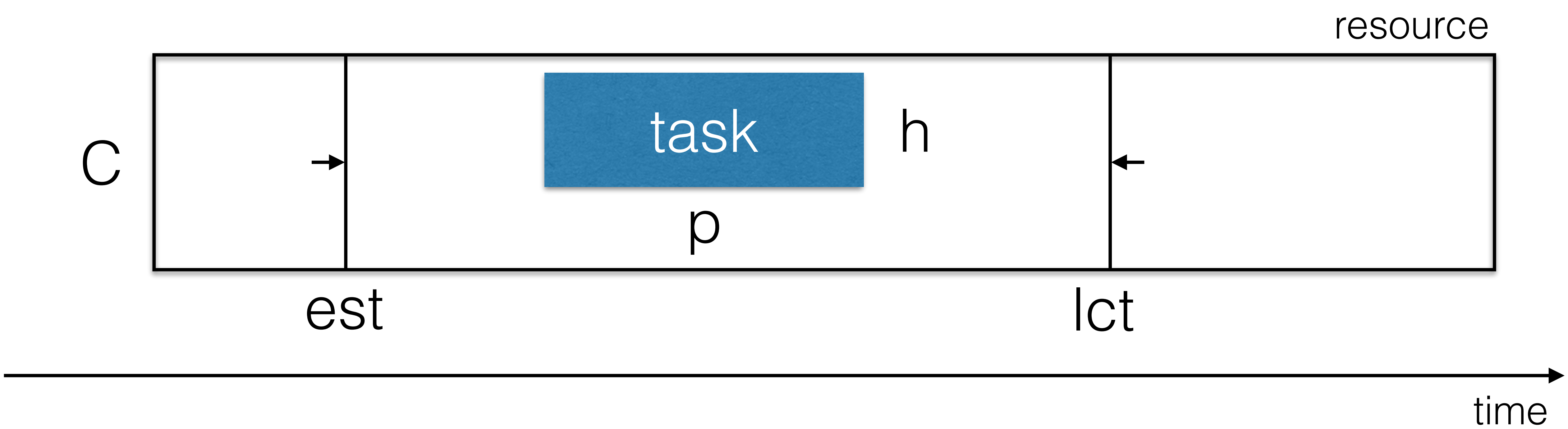
- C: resource capacity
- p: processing time
- h: height
- e: energy ($e = p \times h$)

Definitions



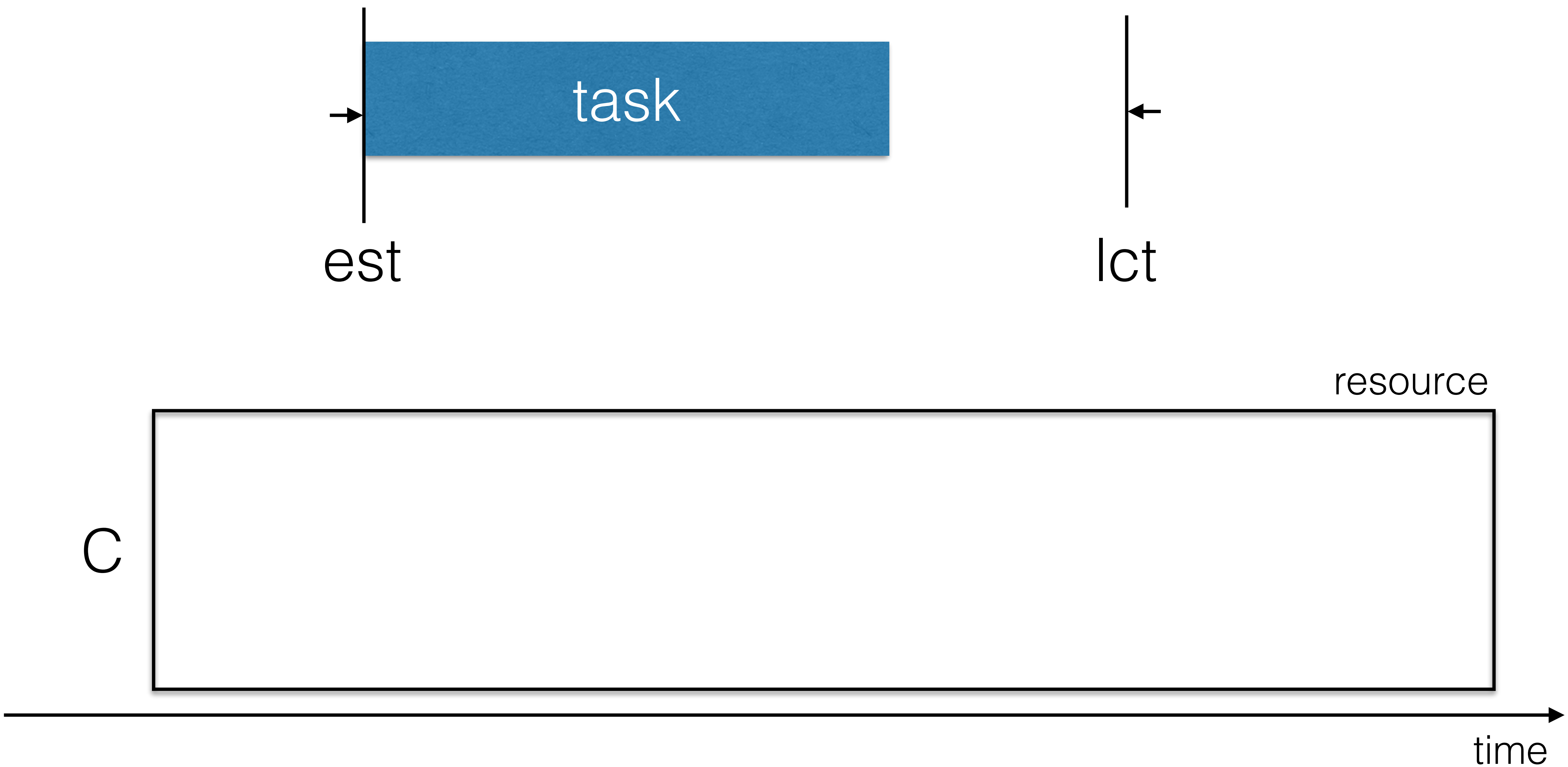
- C: resource capacity
- p: processing time
- h: height
- e: energy ($e = p \times h$)
- est: earliest starting time

Definitions

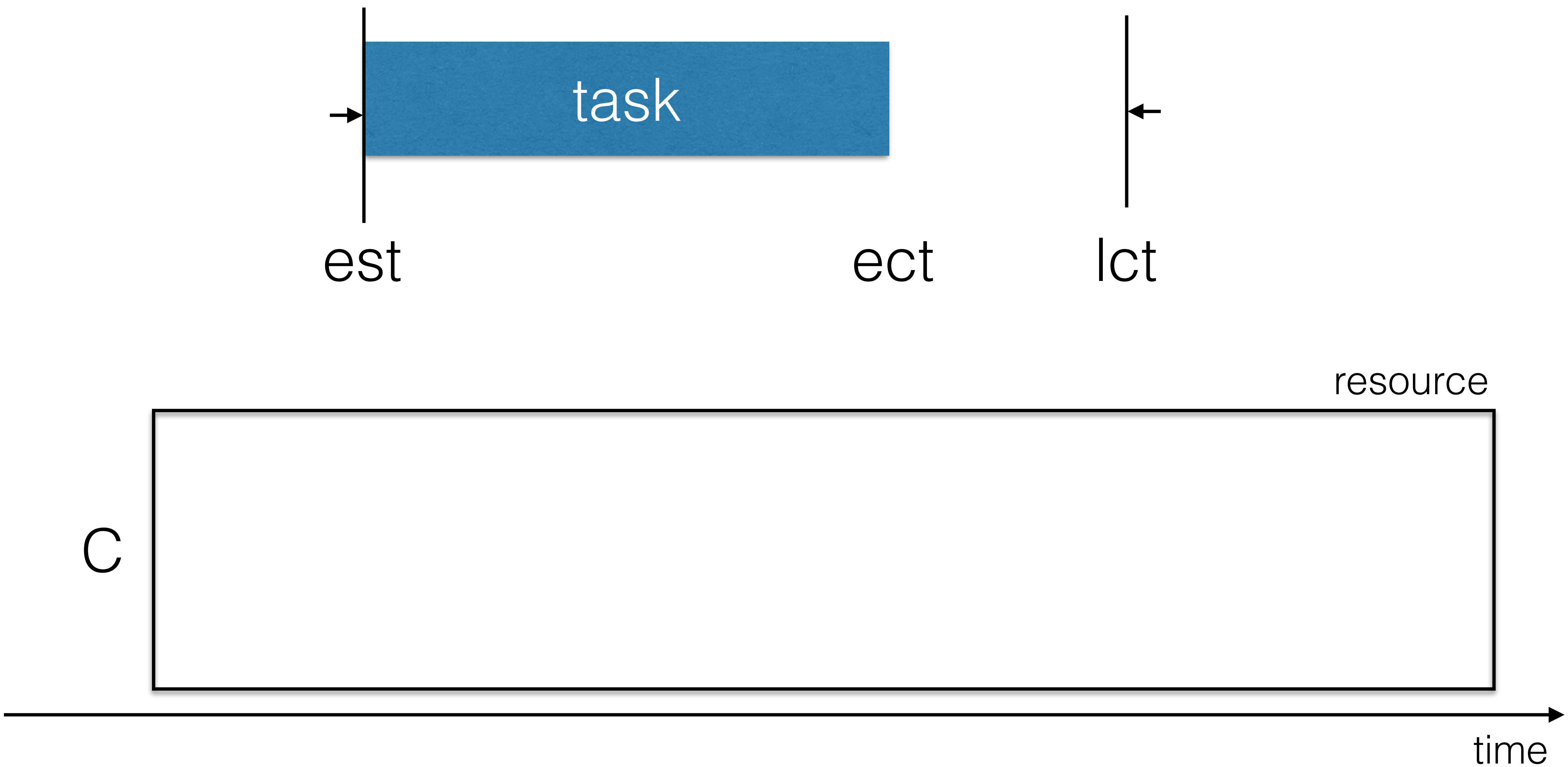


- C: resource capacity
- p: processing time
- h: height
- e: energy ($e = p \times h$)
- est: earliest starting time
- lct: latest completion time

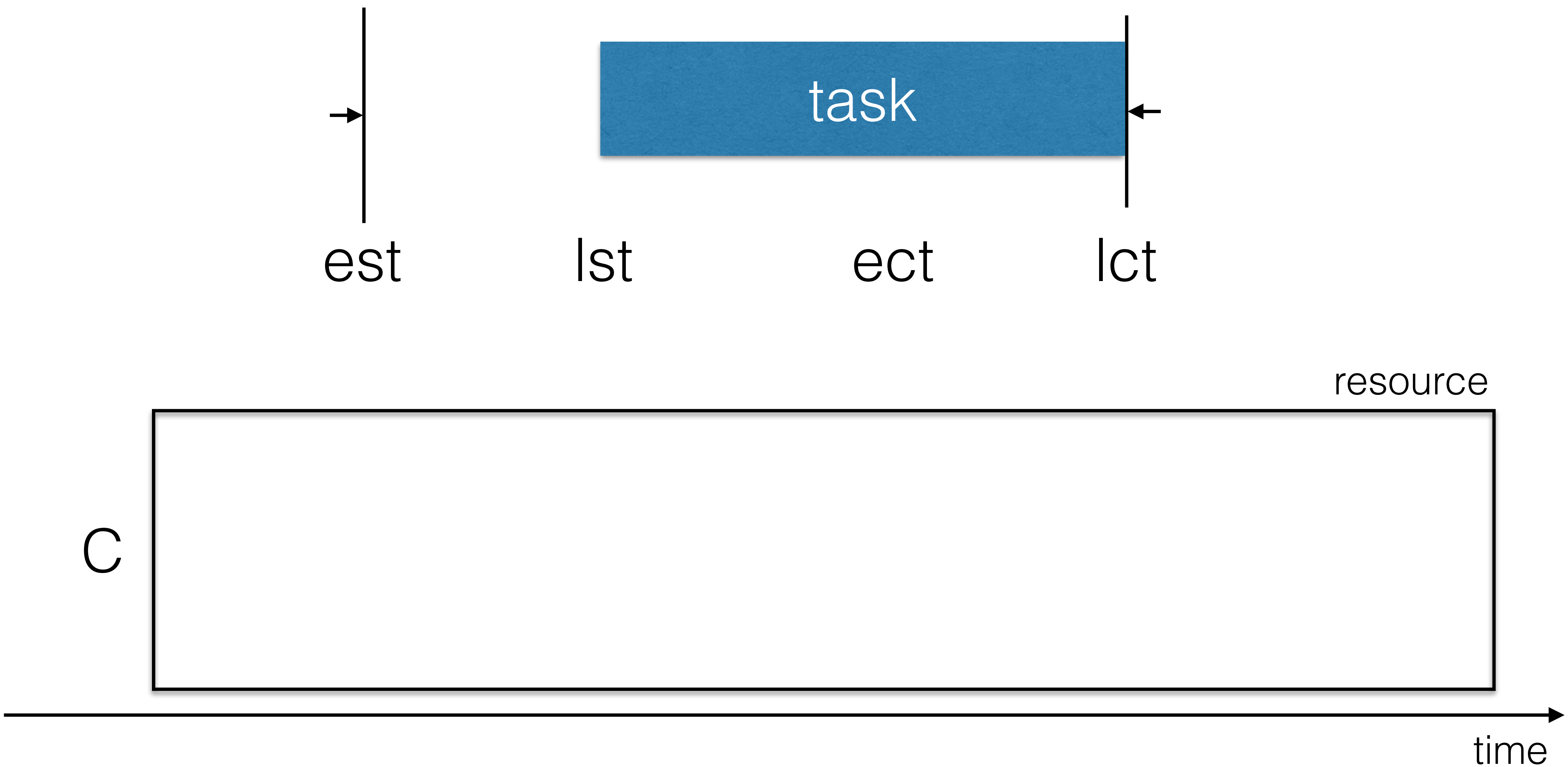
Time Tabling



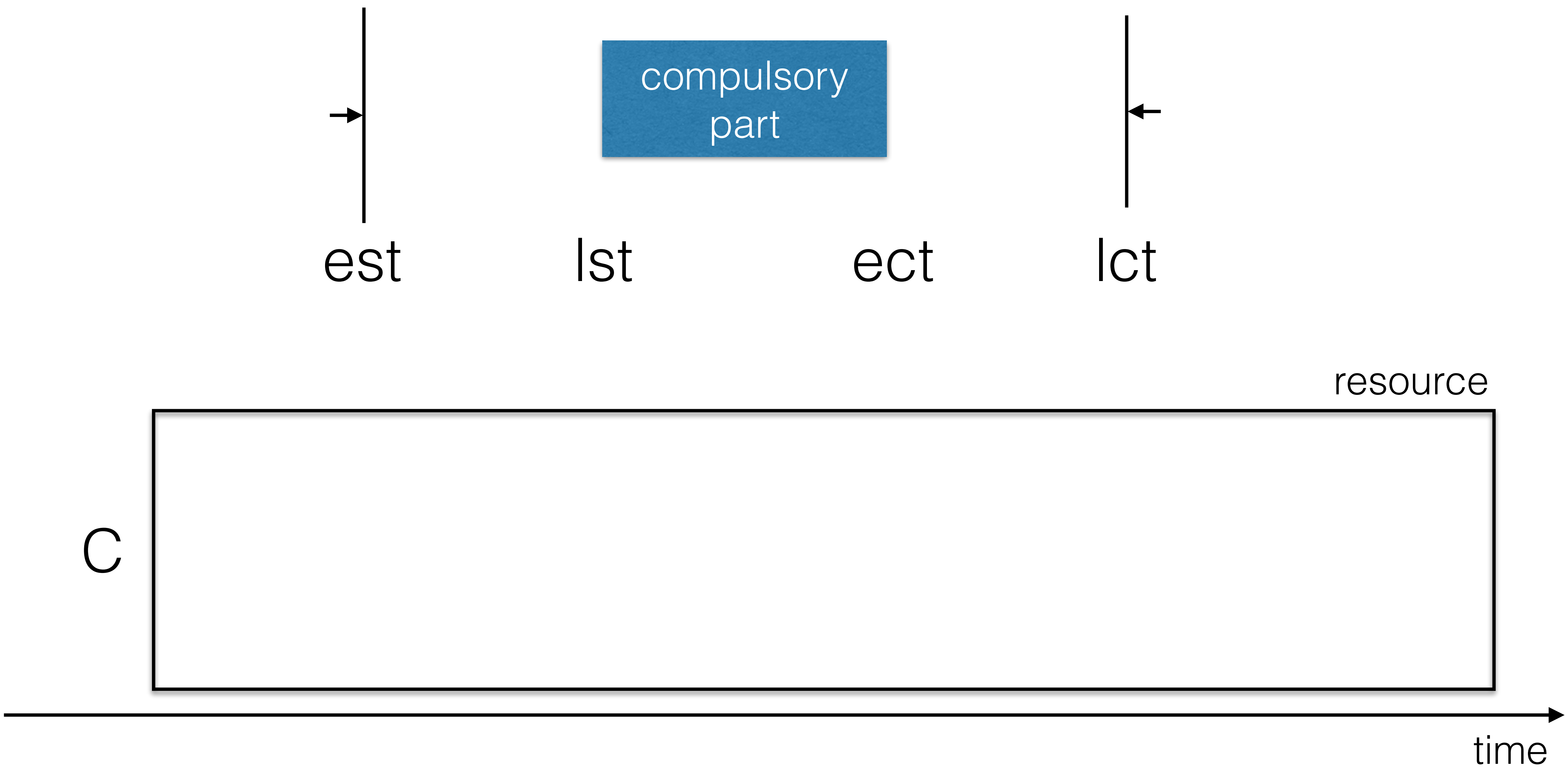
Time Tabling



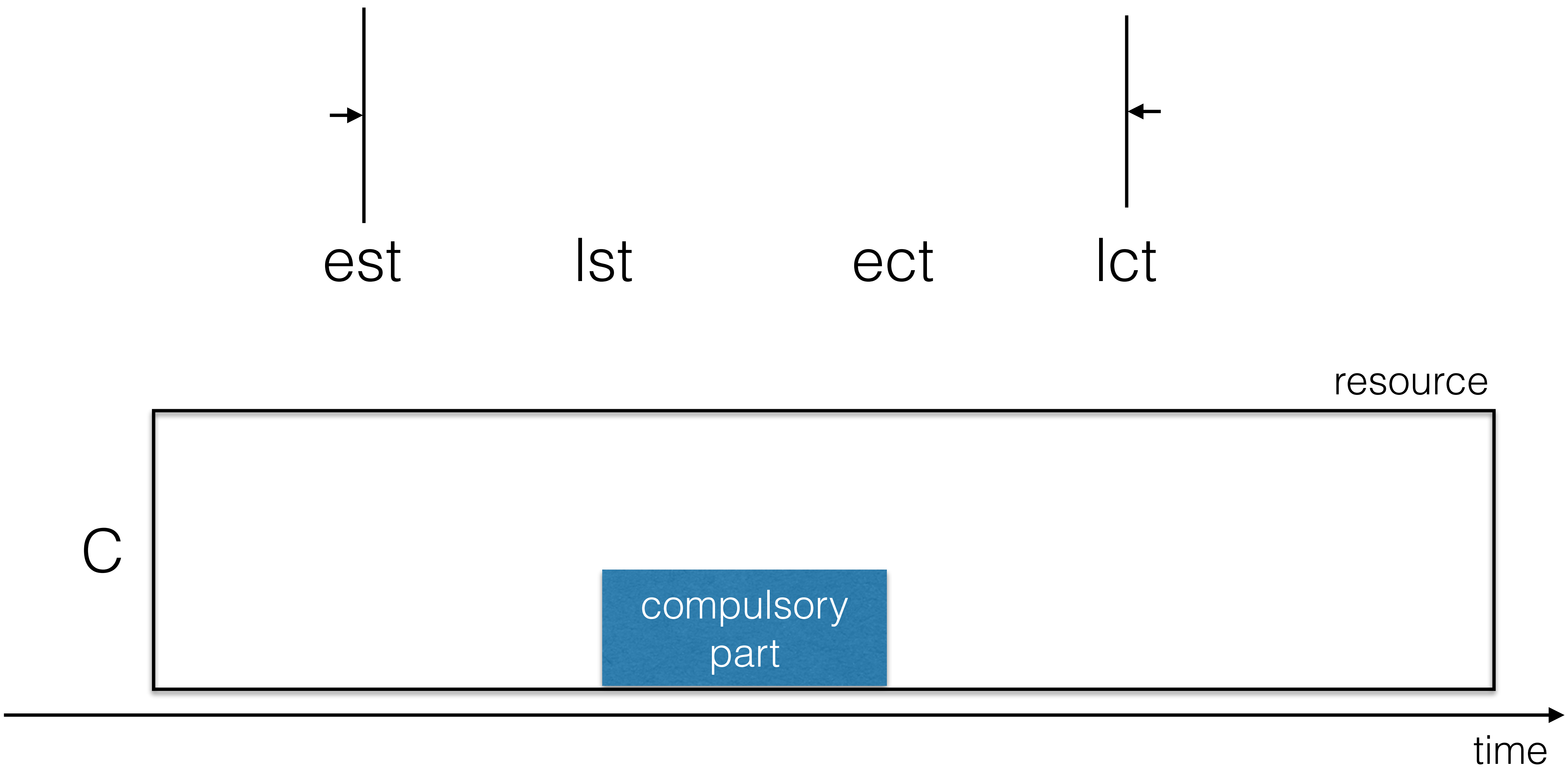
Time Tabling



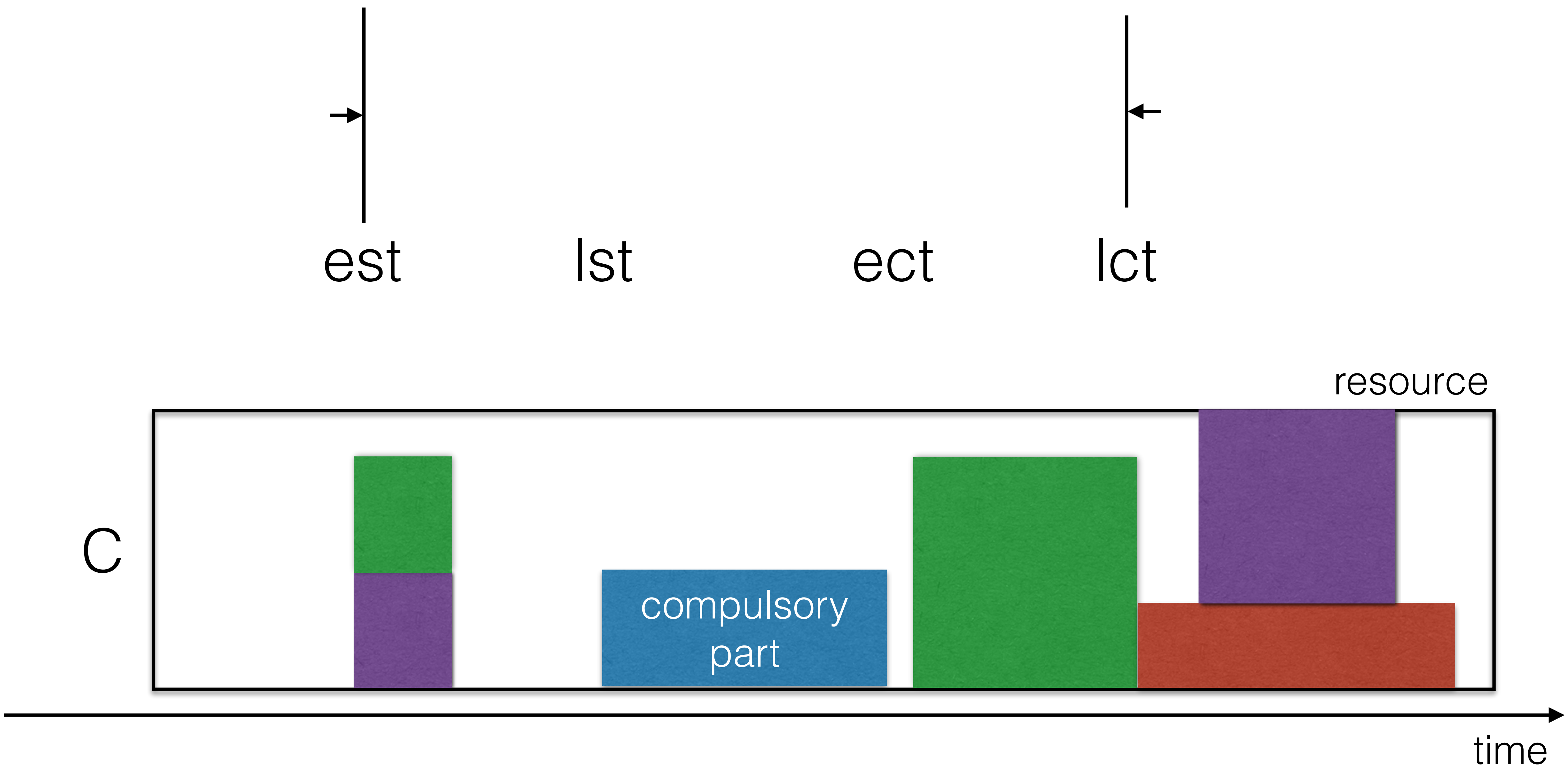
Time Tabling



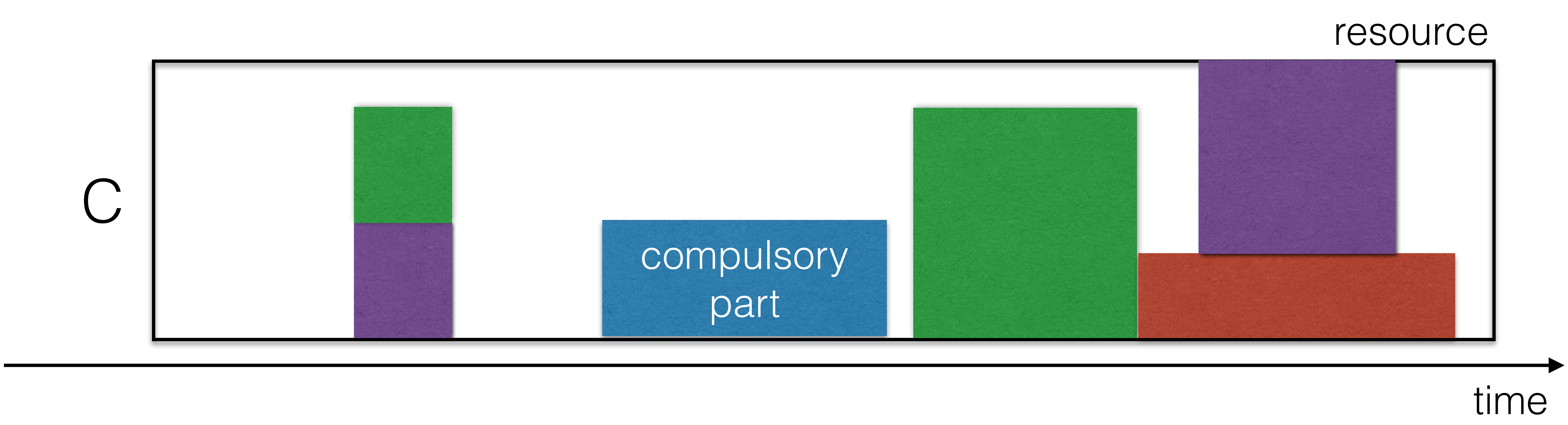
Time Tabling



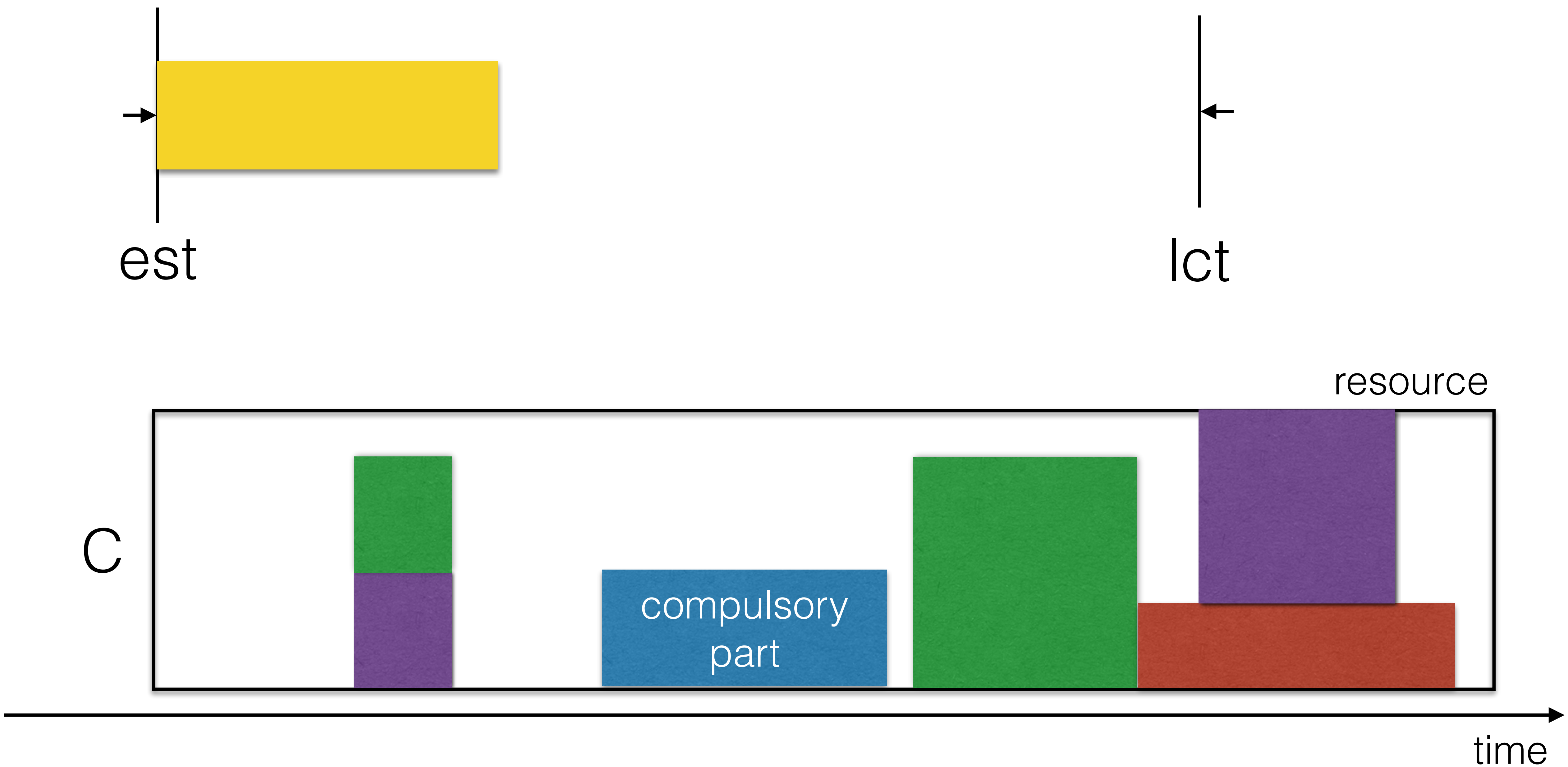
Time Tabling



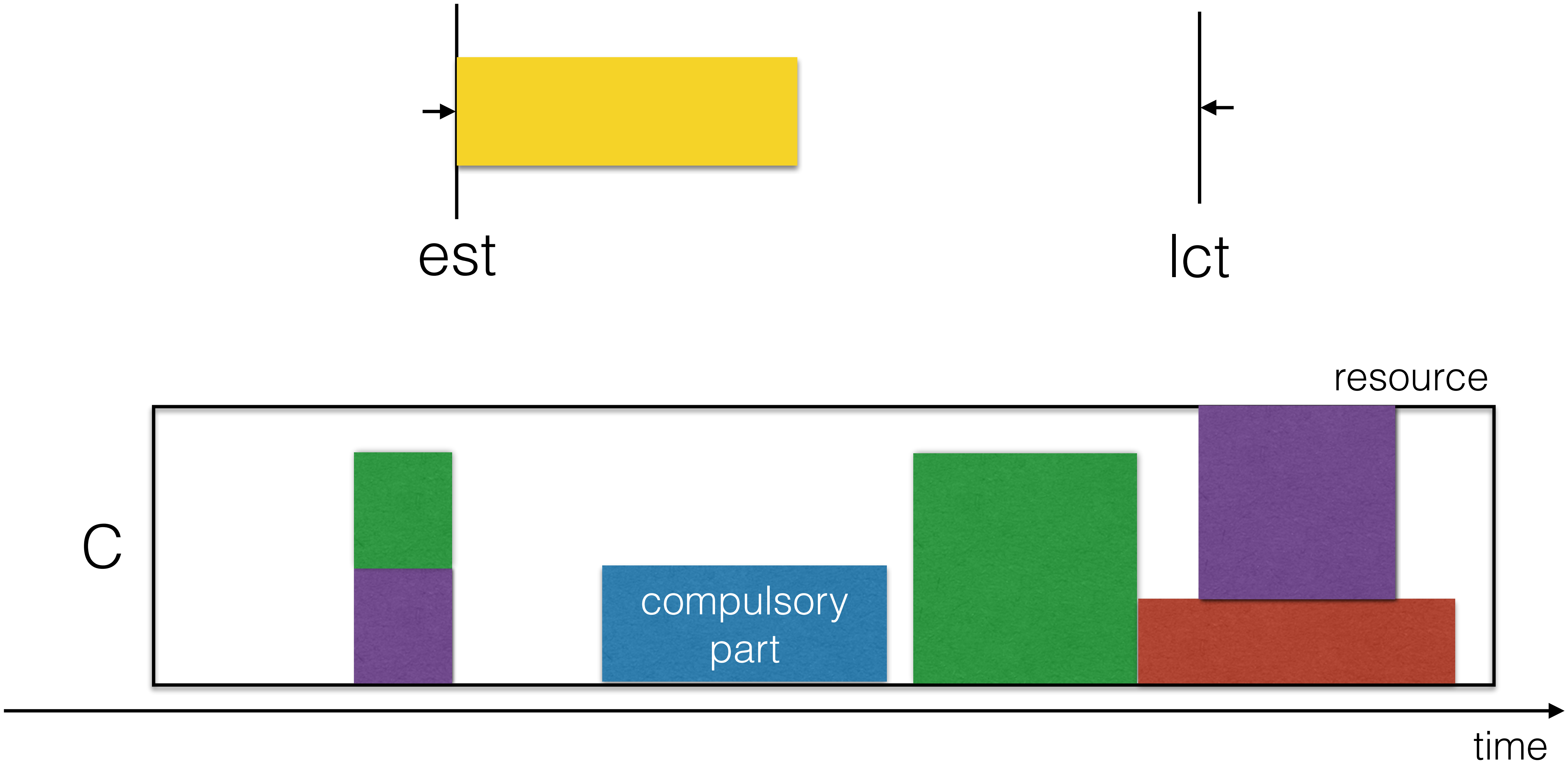
Time Tabling



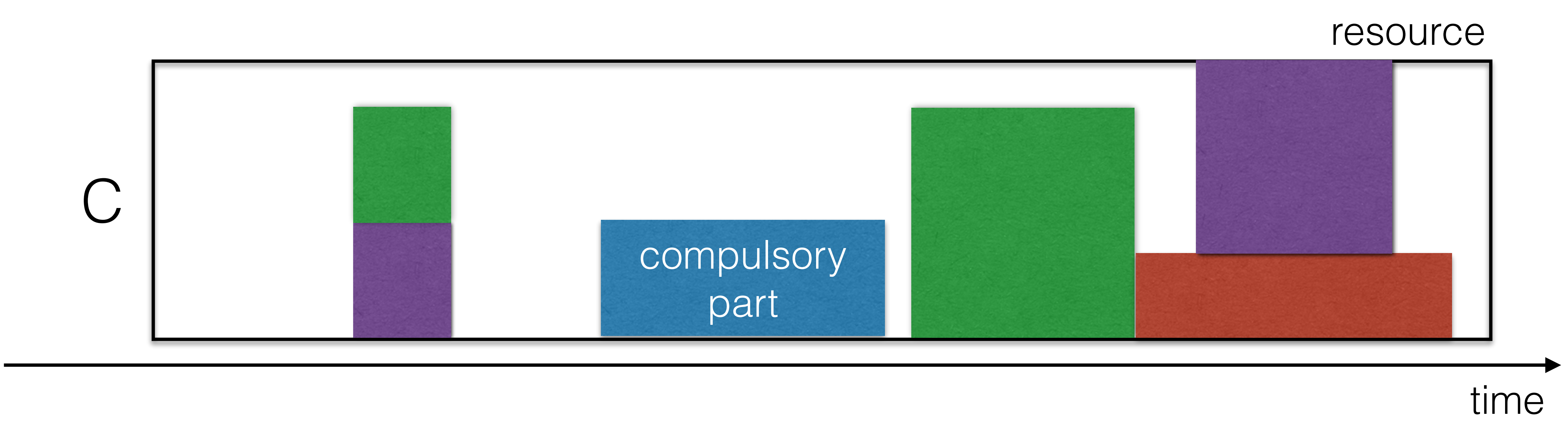
Time Tabling



Time Tabling



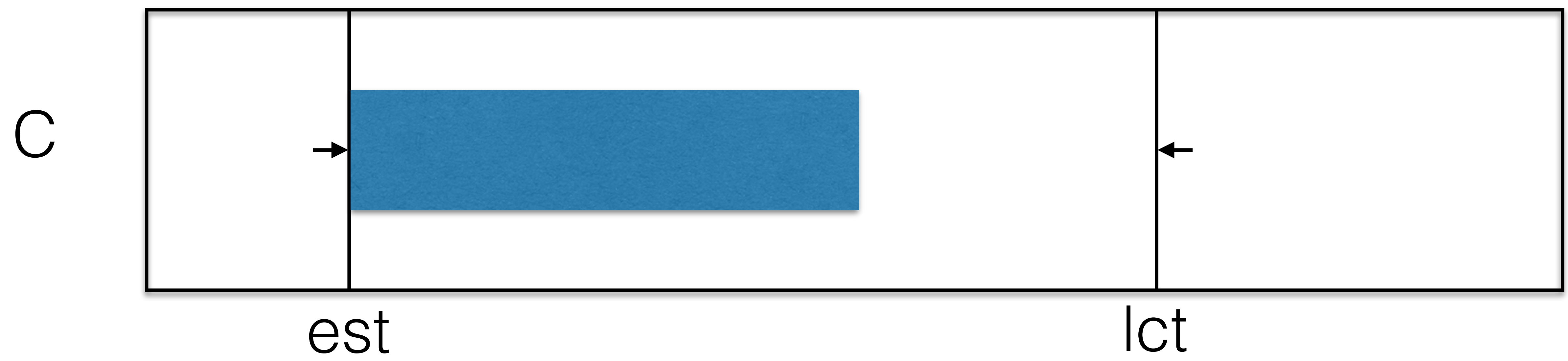
Time Tabling



Algorithms for the Time Tabling

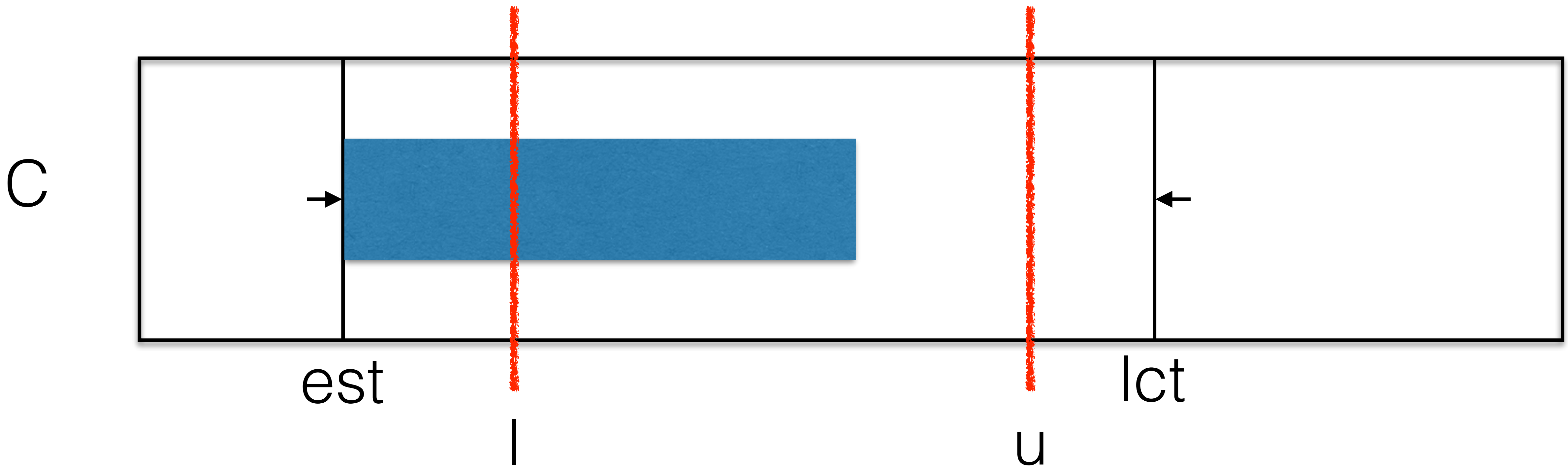
- Disjunctive: O(n) Fahimi, Ouellet & Quimper
- Cumulative: O(n) Gay et al.

Energetic Reasonning



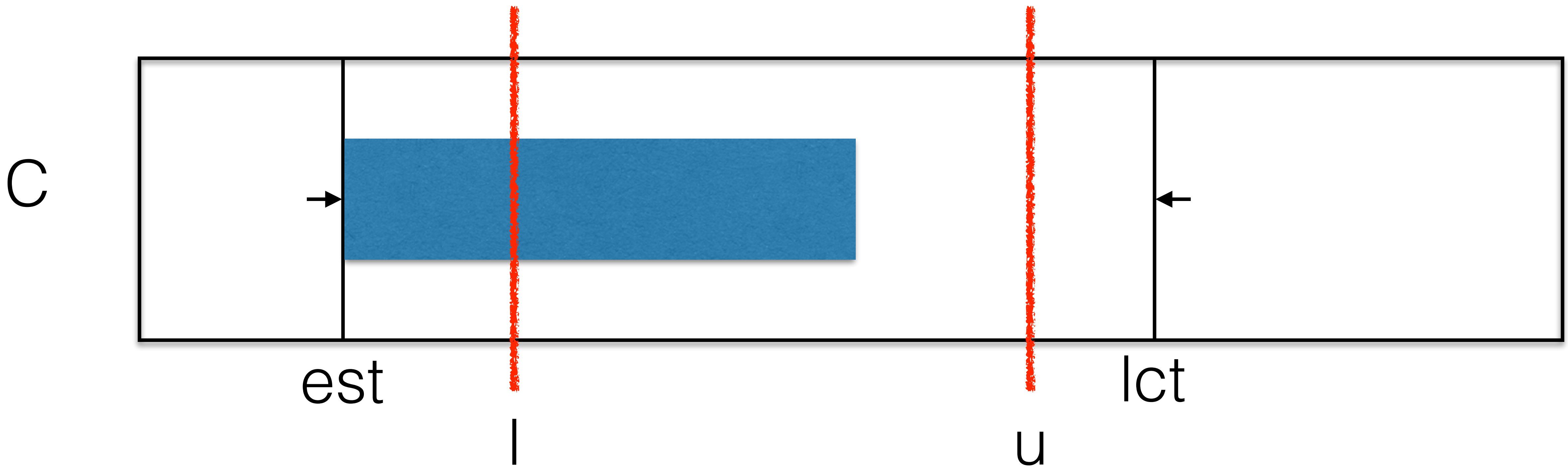
[Baptiste, Le Pape, Nuijten 1999]

Energetic Reasonning



[Baptiste, Le Pape, Nuijten 1999]

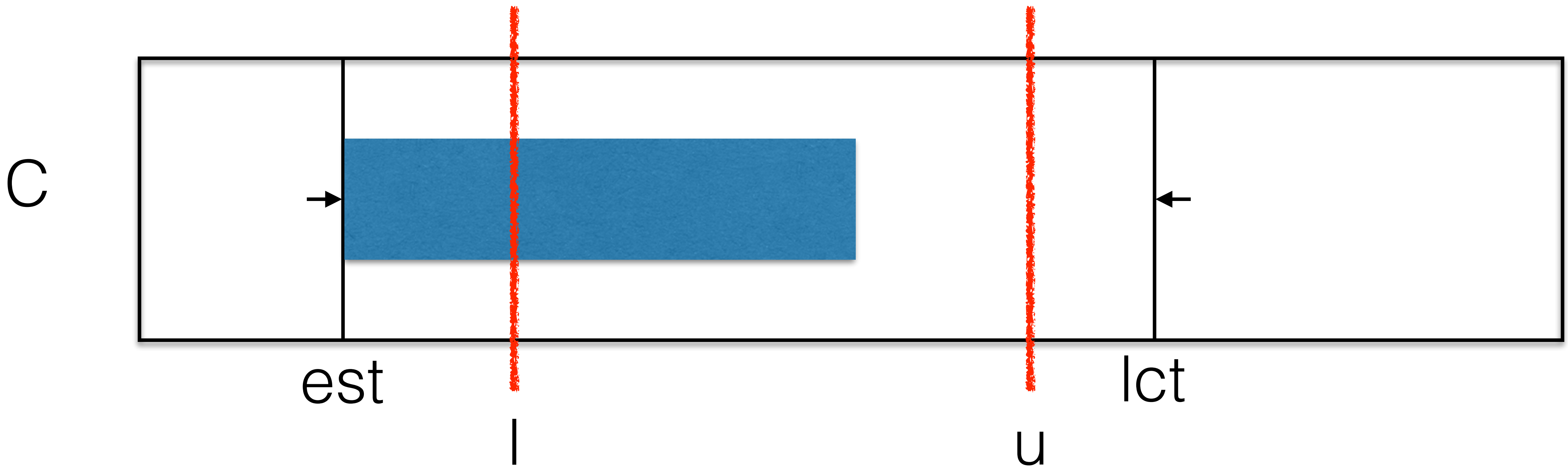
Energetic Reasonning



$$E(i, l, u) = h_i \cdot \max(0, \min(\qquad\qquad))$$

[Baptiste, Le Pape, Nuijten 1999]

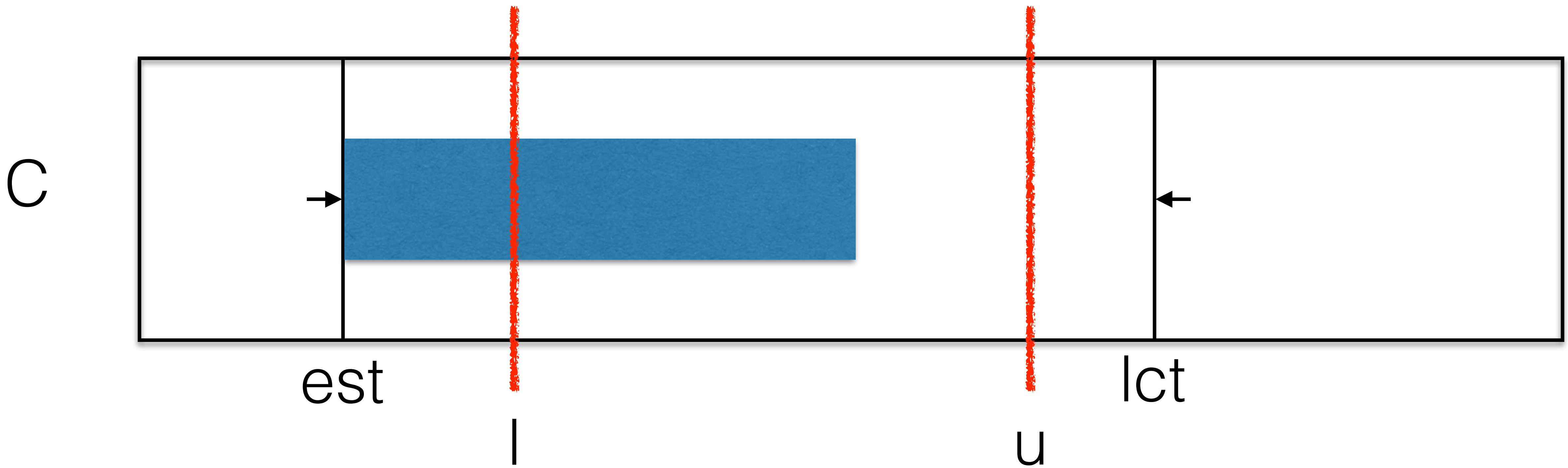
Energetic Reasonning



$$E(i, l, u) = h_i \cdot \max(0, \min(u - l, \qquad\qquad))$$

[Baptiste, Le Pape, Nuijten 1999]

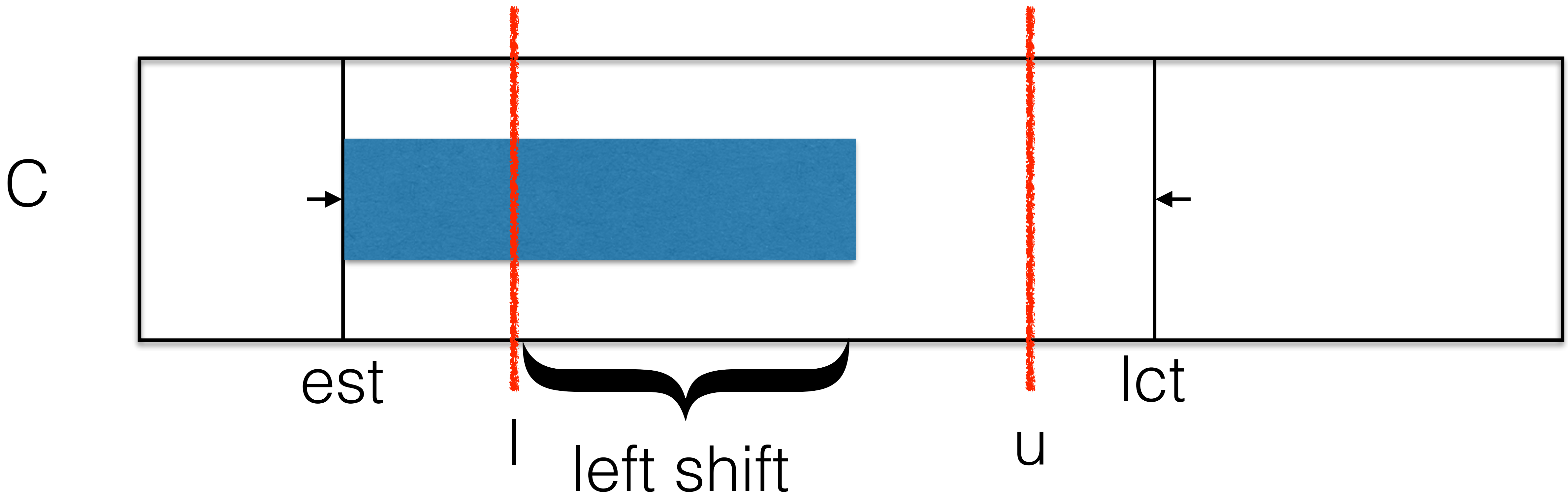
Energetic Reasonning



$$E(i, l, u) = h_i \cdot \max(0, \min(u - l, p_i, \qquad\qquad\qquad\qquad))$$

[Baptiste, Le Pape, Nuijten 1999]

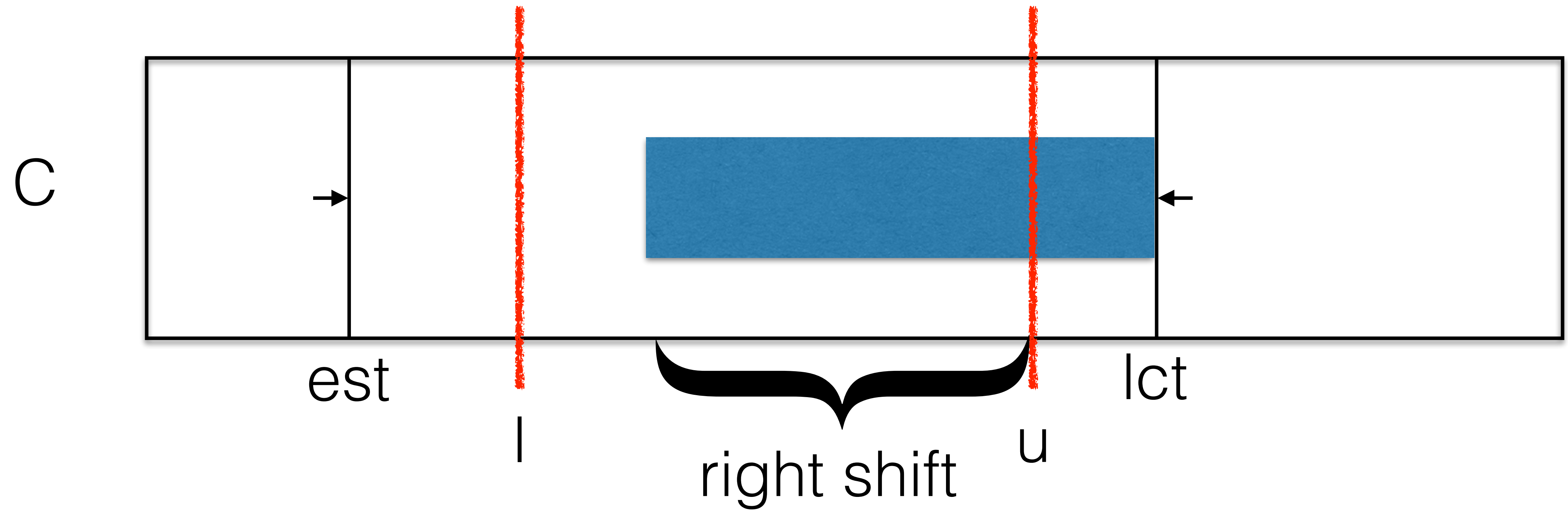
Energetic Reasonning



$$E(i,l,u) = h_i \cdot \max(0, \min(u-l, p_i, \mathrm{est}_i + p_i - l \qquad))$$

[Baptiste, Le Pape, Nuijten 1999]

Energetic Reasonning



$$E(i, l, u) = h_i \cdot \max(0, \min(u - l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

[Baptiste, Le Pape, Nuijten 1999]

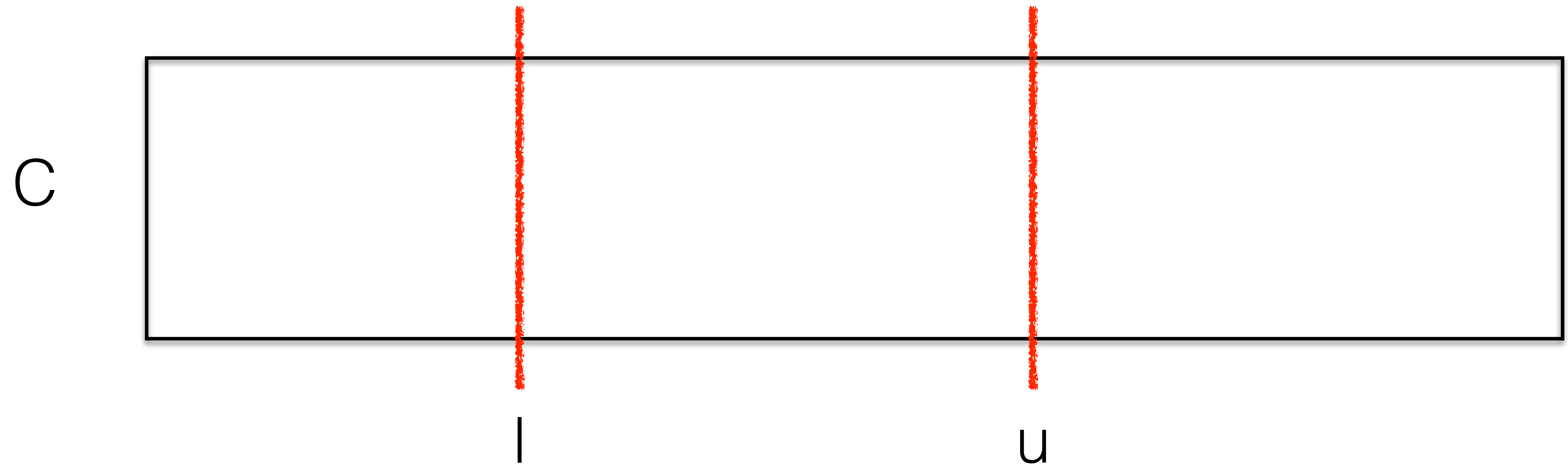
Energetic Reasoning

$$E(i,l,u) = h_i \cdot \max(0, \min(u-l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

$$S(l,u) = C \cdot (u-l) - \sum_i E(i,l,u)$$

[Baptiste, Le Pape, Nuijten 1999]

Energetic Reasonning



$$E(i, l, u) = h_i \cdot \max(0, \min(u - l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

$$S(l, u) = C \cdot (u - l) - \sum_i E(i, l, u) \geq 0$$

[Baptiste, Le Pape, Nuijten 1999]

Energetic Reasoning Check

Erschler et al.

1991

Baptiste et al.
$O(n^2)$

1999

Derrien et al.
$O(n^2)$

2014

Ouellet & Quimper
$O(n \log^2 n)$

2018

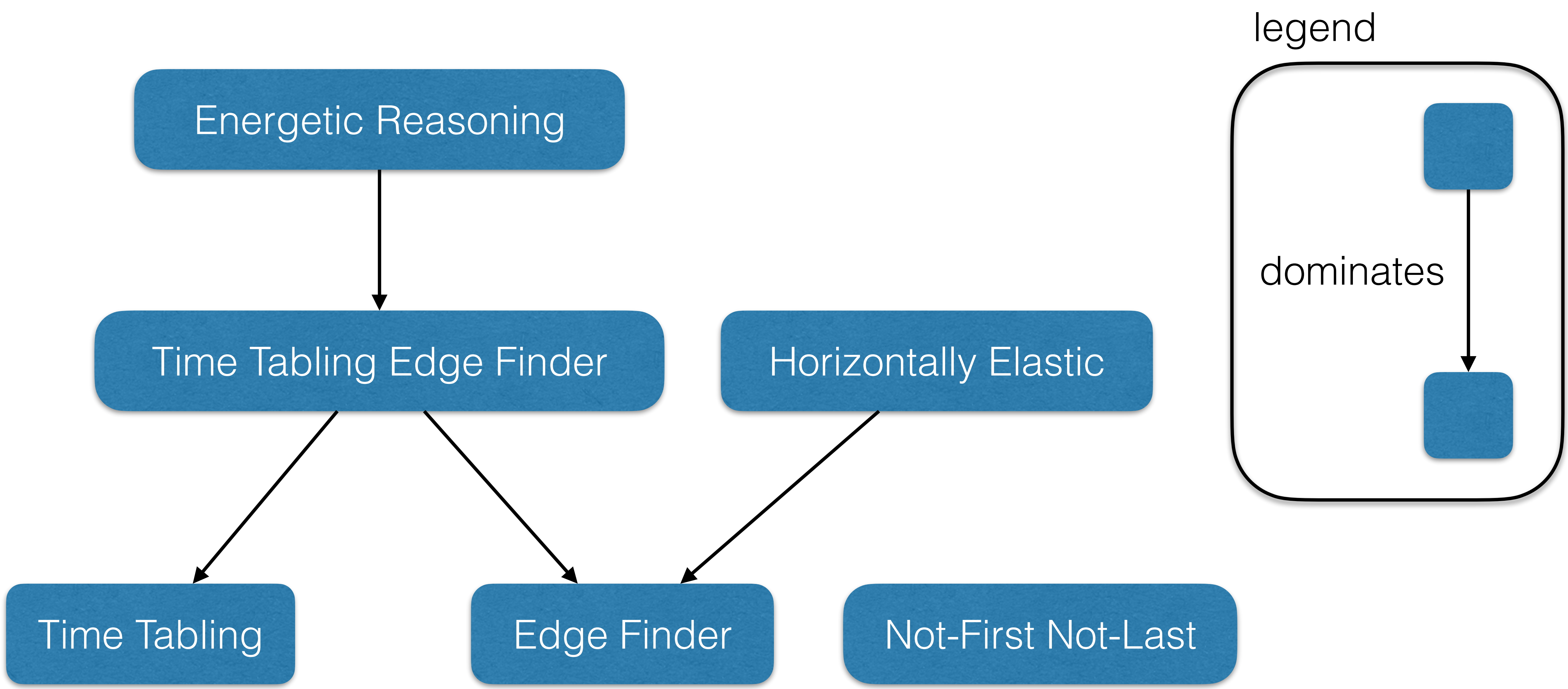
Carlier et al.
$O(n \log(n)\ \alpha(n))$

2020

Energetic Reasoning Filtering

- 1991 — Erschler et al.
- 1999 — Baptiste et al. $O(n^3)$
- 2014 — Derrien et al. $O(n^3)$
- 2018 — Ouellet & Quimper $O(n^2 \log n)$; Tesch $O(n^2 \log n)$
- 2020 — Carlier et al. $O(n^2)$

Filtering Rules



Nogood Learning

- Also called Lazy Clause Generation
- When a solver reaches a dead end in its search, it analyzes which choices are conflicting.

Nogood Learning

- Also called Lazy Clause Generation
- When a solver reaches a dead end in its search, it analyzes which choices are conflicting.
- The solver adds to the model a *clause*, i.e. a disjunction that forces at least once choice to be different.

Nogood Learning

- Also called Lazy Clause Generation
- When a solver reaches a dead end in its search, it analyzes which choices are conflicting.
- The solver adds to the model a *clause*, i.e. a disjunction that forces at least once choice to be different.
- By its filtering algorithm, this clause will prevent the solver from repeating its mistake.

Example

$$x_4 = \max([x_1, x_2, x_3])$$
$$x_1 \neq x_2$$
$$x_2 \neq x_3$$
$$x_1 \neq x_3$$
$$x_1 \in \{1, 2, 3\}$$
$$x_2 \in \{1, 2, 3\}$$
$$x_3 \in \{1, 2, 3\}$$
$$x_4 \in \{1, 2, 3\}$$

Example

$\boxed{x_4 = 2}$

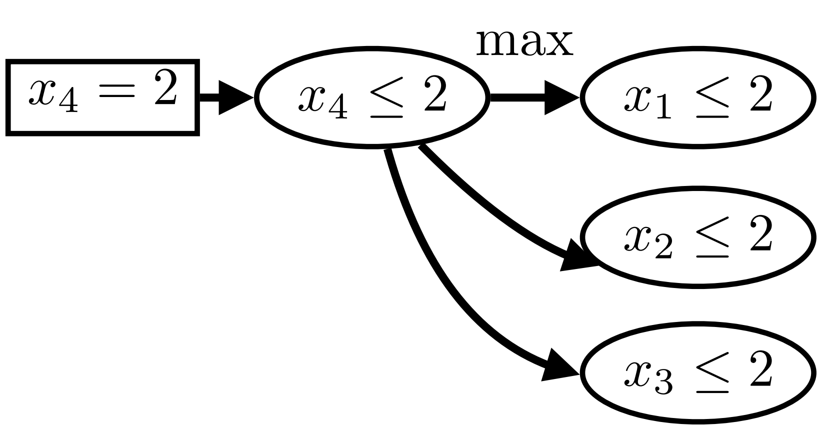
$$
\begin{aligned}
&x_4 = \max([x_1, x_2, x_3]) \\
&x_1 \neq x_2 \\
&x_2 \neq x_3 \\
&x_1 \neq x_3 \\
&x_1 \in \{1, 2, 3\} \\
&x_2 \in \{1, 2, 3\} \\
&x_3 \in \{1, 2, 3\} \\
&x_4 \in \{\cancel{1}, 2, \cancel{3}\}
\end{aligned}
$$

Example

$$\boxed{x_4 = 2} \rightarrow x_4 \leq 2$$

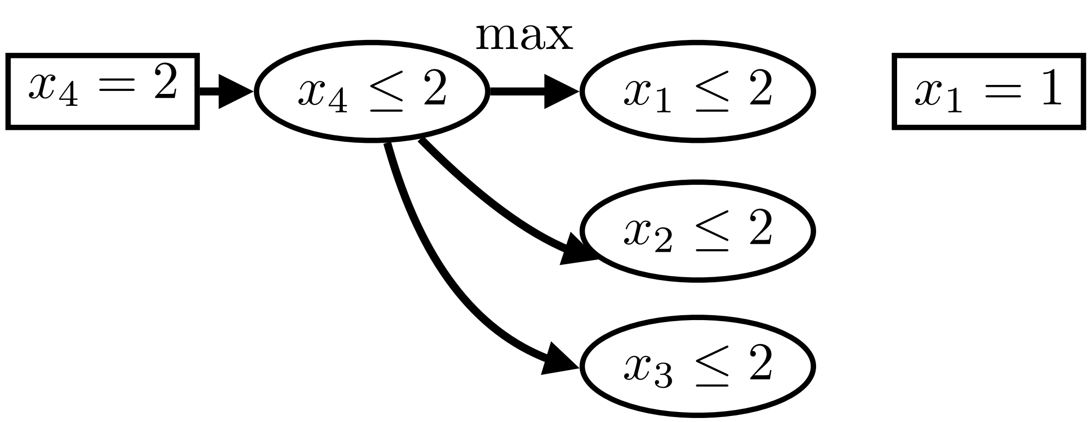
$$x_4 = \max([x_1, x_2, x_3])$$
$$x_1 \neq x_2$$
$$x_2 \neq x_3$$
$$x_1 \neq x_3$$
$$x_1 \in \{1, 2, 3\}$$
$$x_2 \in \{1, 2, 3\}$$
$$x_3 \in \{1, 2, 3\}$$
$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$
$$x_1 \neq x_2$$
$$x_2 \neq x_3$$
$$x_1 \neq x_3$$
$$x_1 \in \{1, 2, \cancel{3}\}$$
$$x_2 \in \{1, 2, \cancel{3}\}$$
$$x_3 \in \{1, 2, \cancel{3}\}$$
$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

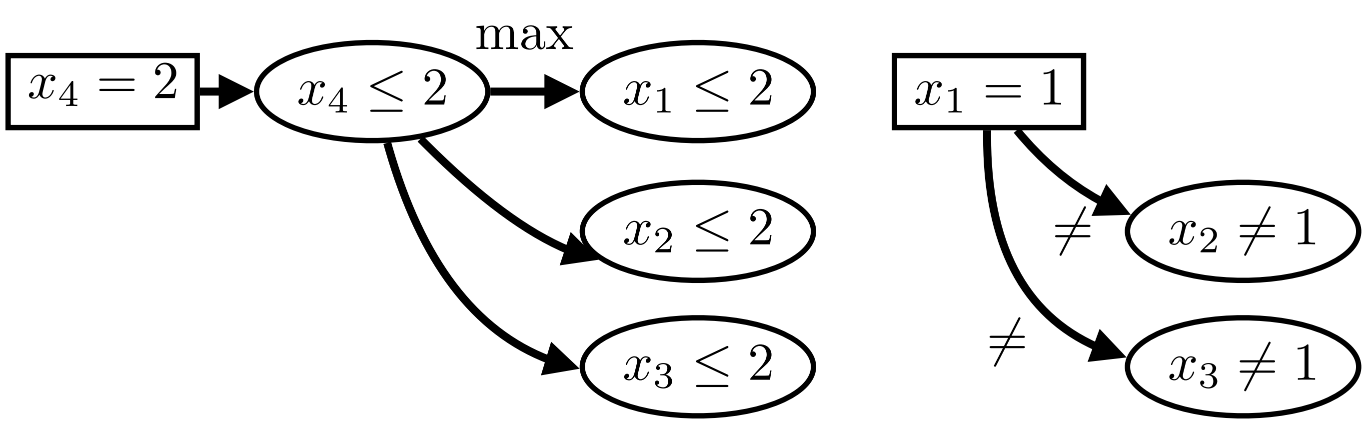
$$x_1 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_2 \in \{1, 2, \cancel{3}\}$$

$$x_3 \in \{1, 2, \cancel{3}\}$$

$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

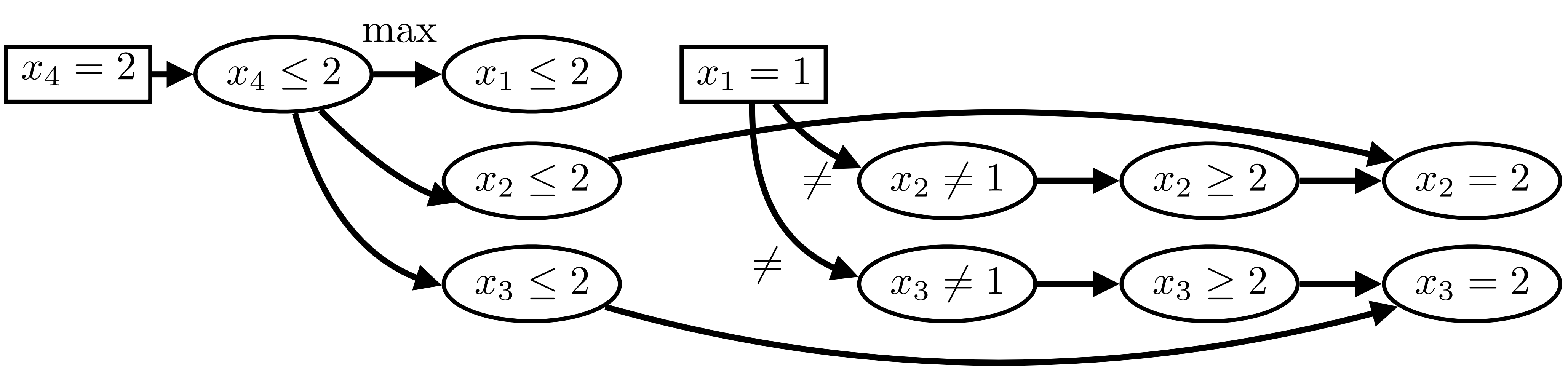
$$x_1 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_2 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$x_3 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

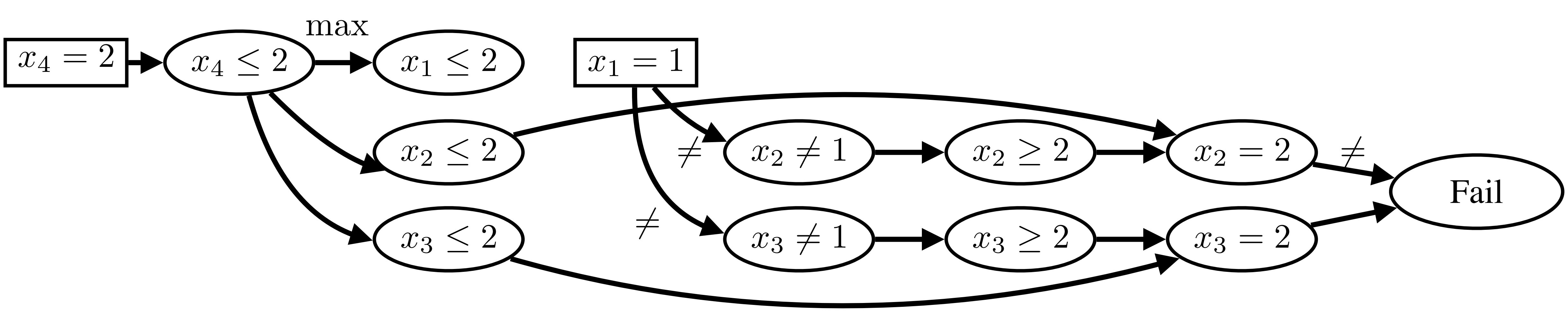
$$x_1 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_2 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$x_3 \in \{\cancel{1}, 2, \cancel{3}\}$$

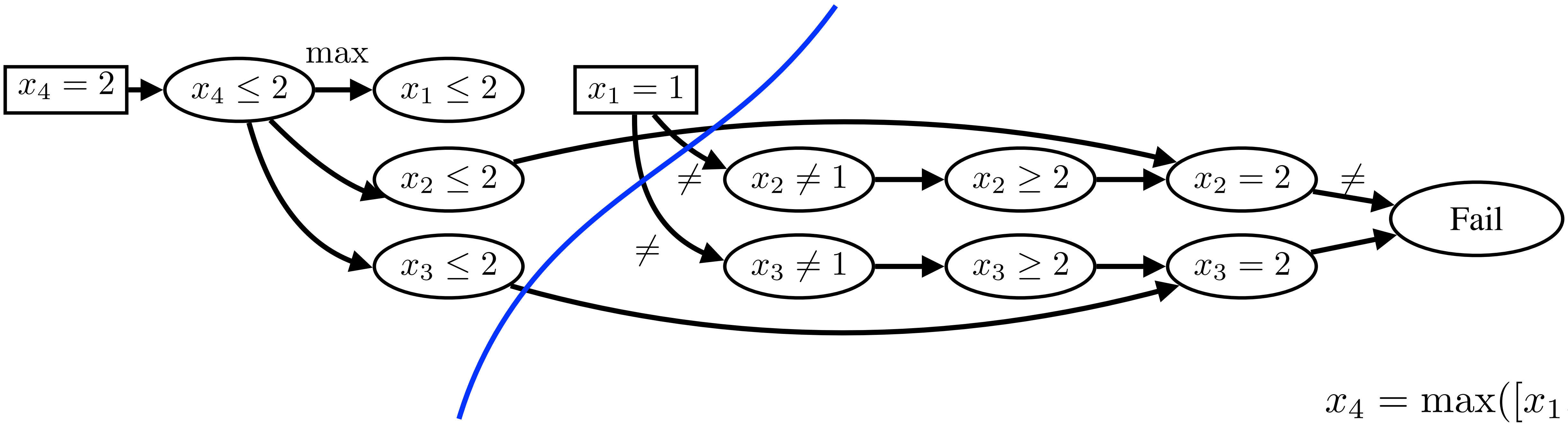
$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$
$$x_1 \neq x_2$$
$$x_2 \neq x_3$$
$$x_1 \neq x_3$$
$$x_1 \in \{1, \cancel{2}, \cancel{3}\}$$
$$x_2 \in \{\cancel{1}, 2, \cancel{3}\}$$
$$x_3 \in \{\cancel{1}, 2, \cancel{3}\}$$
$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

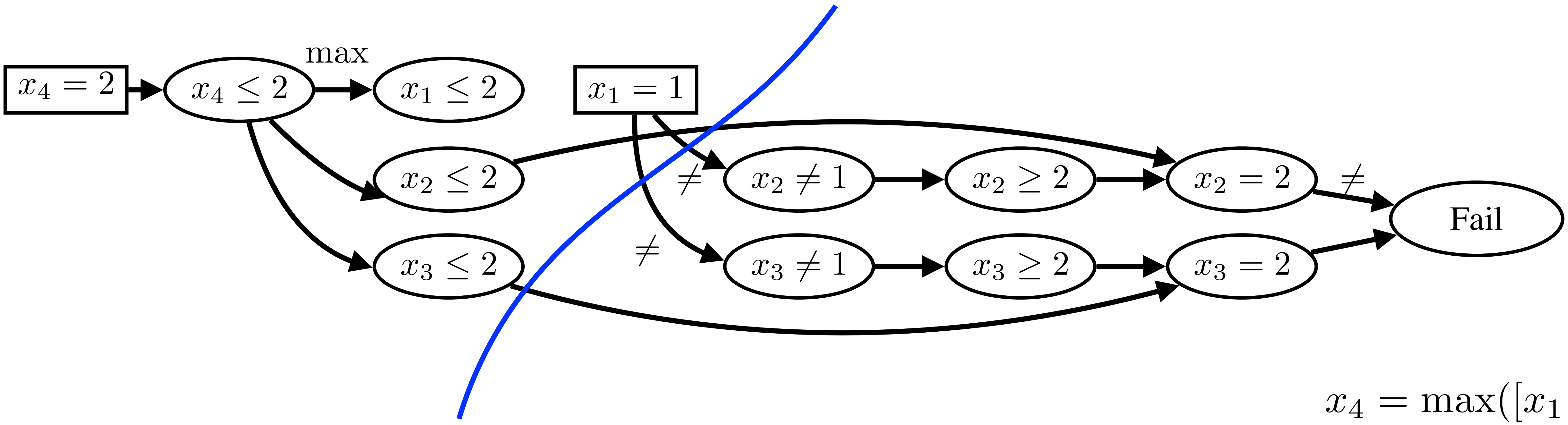
$$x_1 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_2 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$x_3 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

Example



Learnt clause $\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$

$x_4 = \max([x_1, x_2, x_3])$

$x_1 \neq x_2$

$x_2 \neq x_3$

$x_1 \neq x_3$

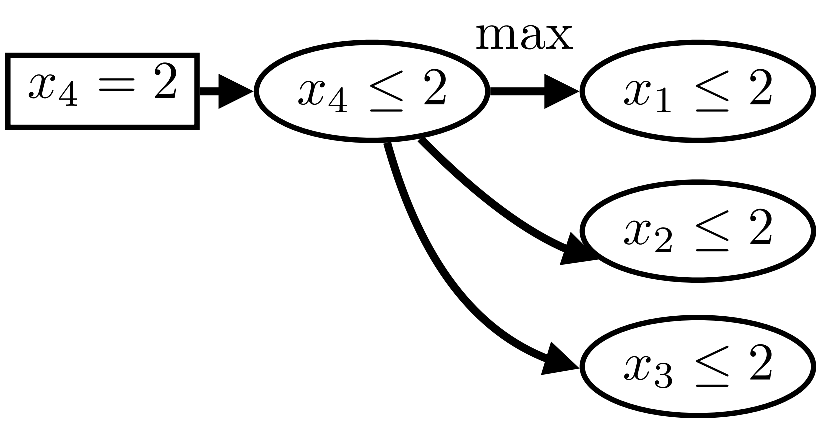
$x_1 \in \{1, \cancel{2}, \cancel{3}\}$

$x_2 \in \{\cancel{1}, 2, \cancel{3}\}$

$x_3 \in \{\cancel{1}, 2, \cancel{3}\}$

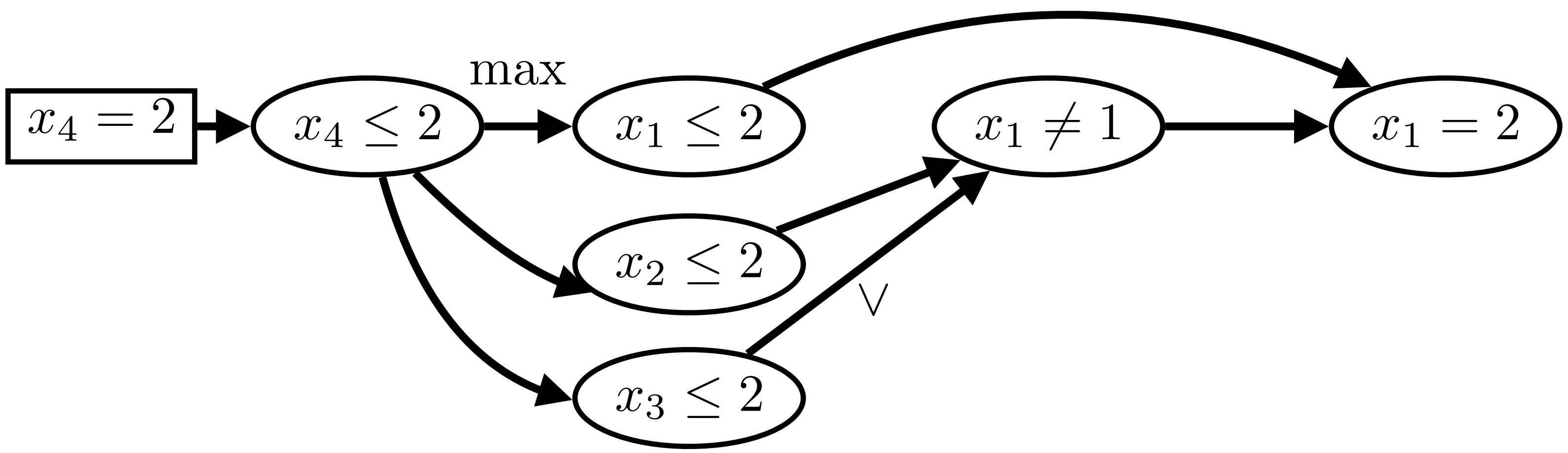
$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$

Example



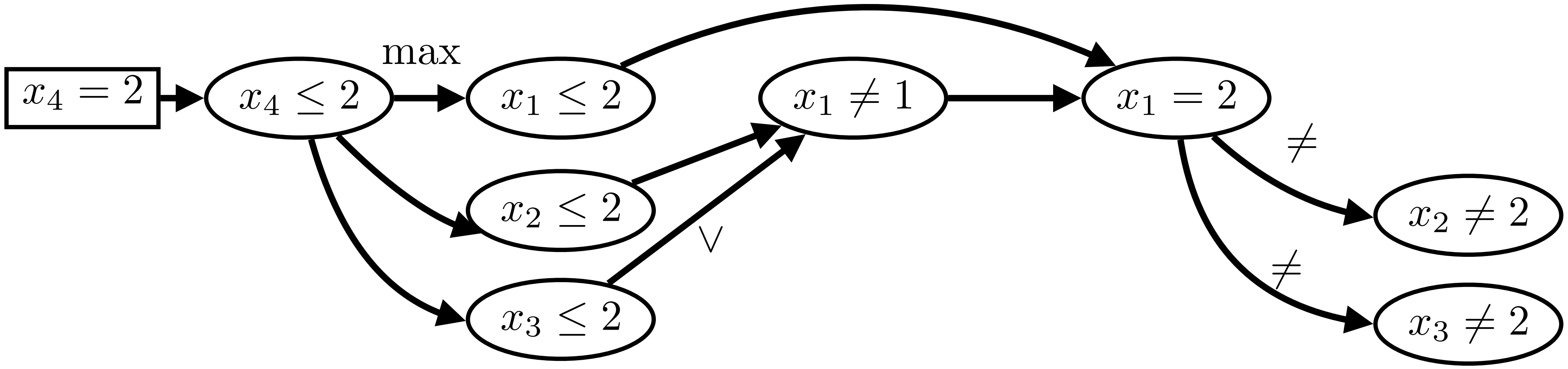
$$x_4 = \max([x_1, x_2, x_3])$$
$$x_1 \neq x_2$$
$$x_2 \neq x_3$$
$$x_1 \neq x_3$$
$$x_1 \in \{1, 2, \cancel{3}\}$$
$$x_2 \in \{1, 2, \cancel{3}\}$$
$$x_3 \in \{1, 2, \cancel{3}\}$$
$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$
$$\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$
$$x_1 \neq x_2$$
$$x_2 \neq x_3$$
$$x_1 \neq x_3$$
$$x_1 \in \{\cancel{1}, 2, \cancel{3}\}$$
$$x_2 \in \{1, 2, \cancel{3}\}$$
$$x_3 \in \{1, 2, \cancel{3}\}$$
$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$
$$\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

$$x_1 \in \{\cancel{1}, 2, \cancel{3}\}$$

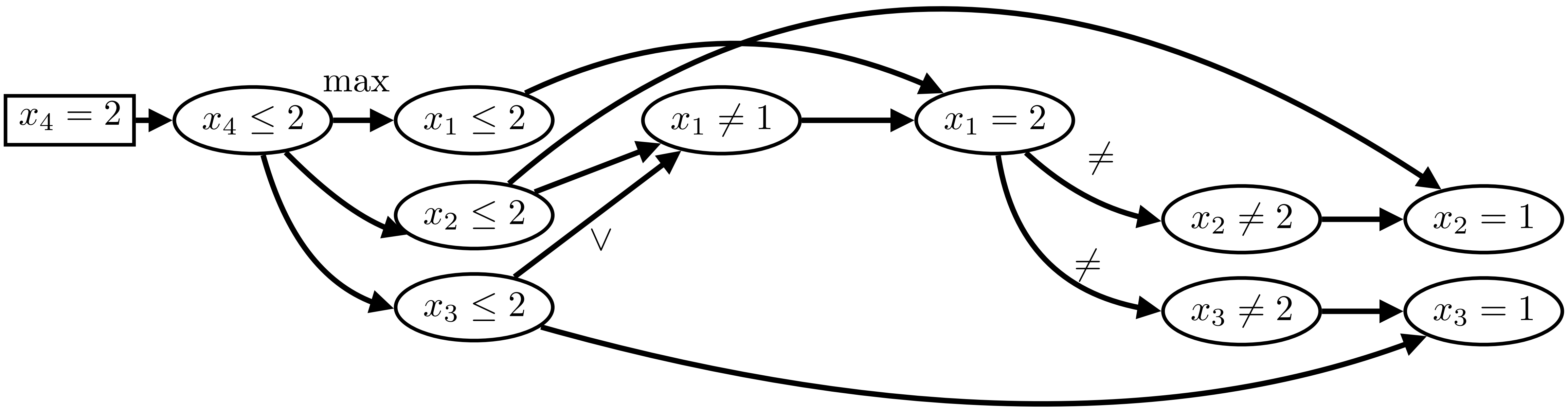
$$x_2 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_3 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

$$x_1 \in \{\cancel{1}, 2, \cancel{3}\}$$

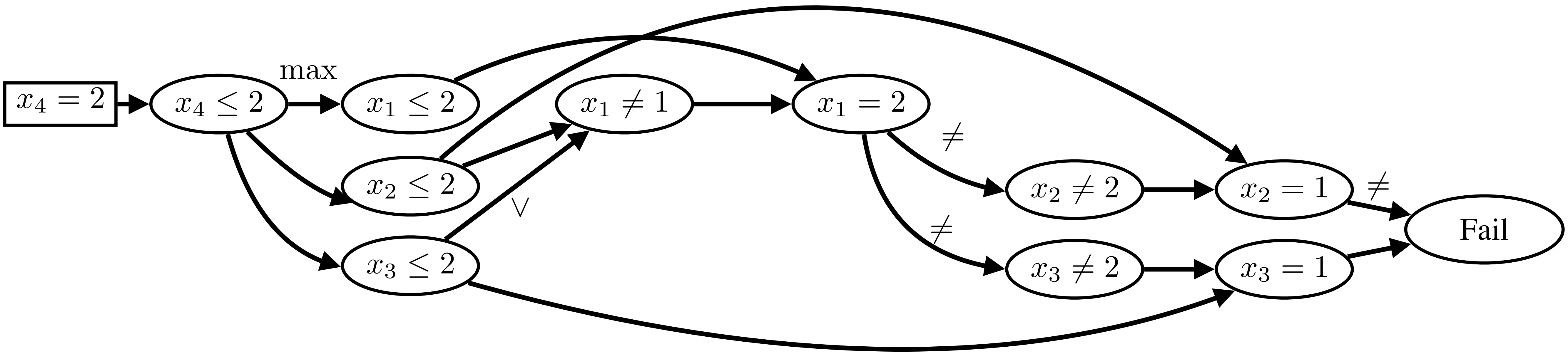
$$x_2 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_3 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$$

Example



$$x_4 = \max([x_1, x_2, x_3])$$

$$x_1 \neq x_2$$

$$x_2 \neq x_3$$

$$x_1 \neq x_3$$

$$x_1 \in \{\cancel{1}, 2, \cancel{3}\}$$

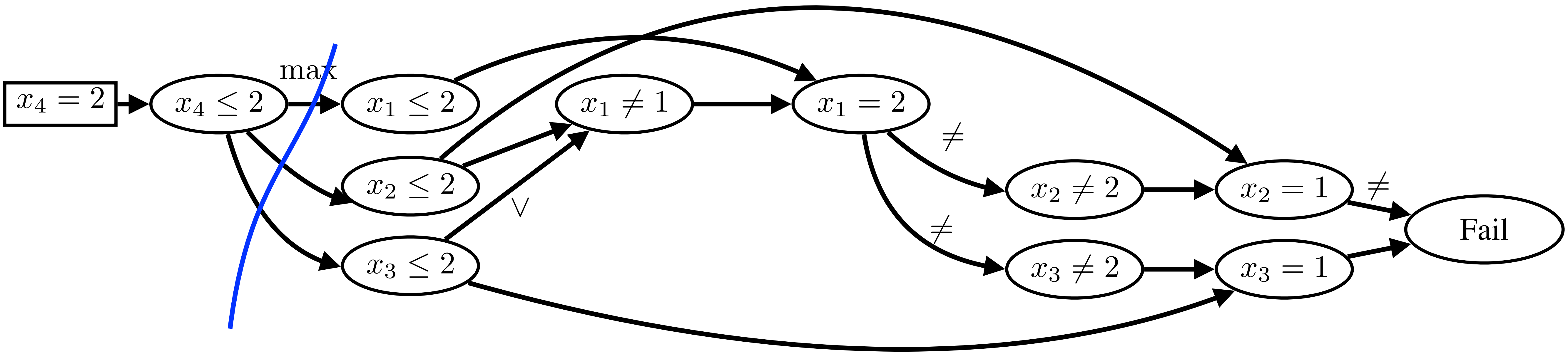
$$x_2 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_3 \in \{1, \cancel{2}, \cancel{3}\}$$

$$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$$

$$\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$$

Example



$x_4 = \max([x_1, x_2, x_3])$

$x_1 \neq x_2$

$x_2 \neq x_3$

$x_1 \neq x_3$

$x_1 \in \{\cancel{1}, 2, \cancel{3}\}$

$x_2 \in \{1, \cancel{2}, \cancel{3}\}$

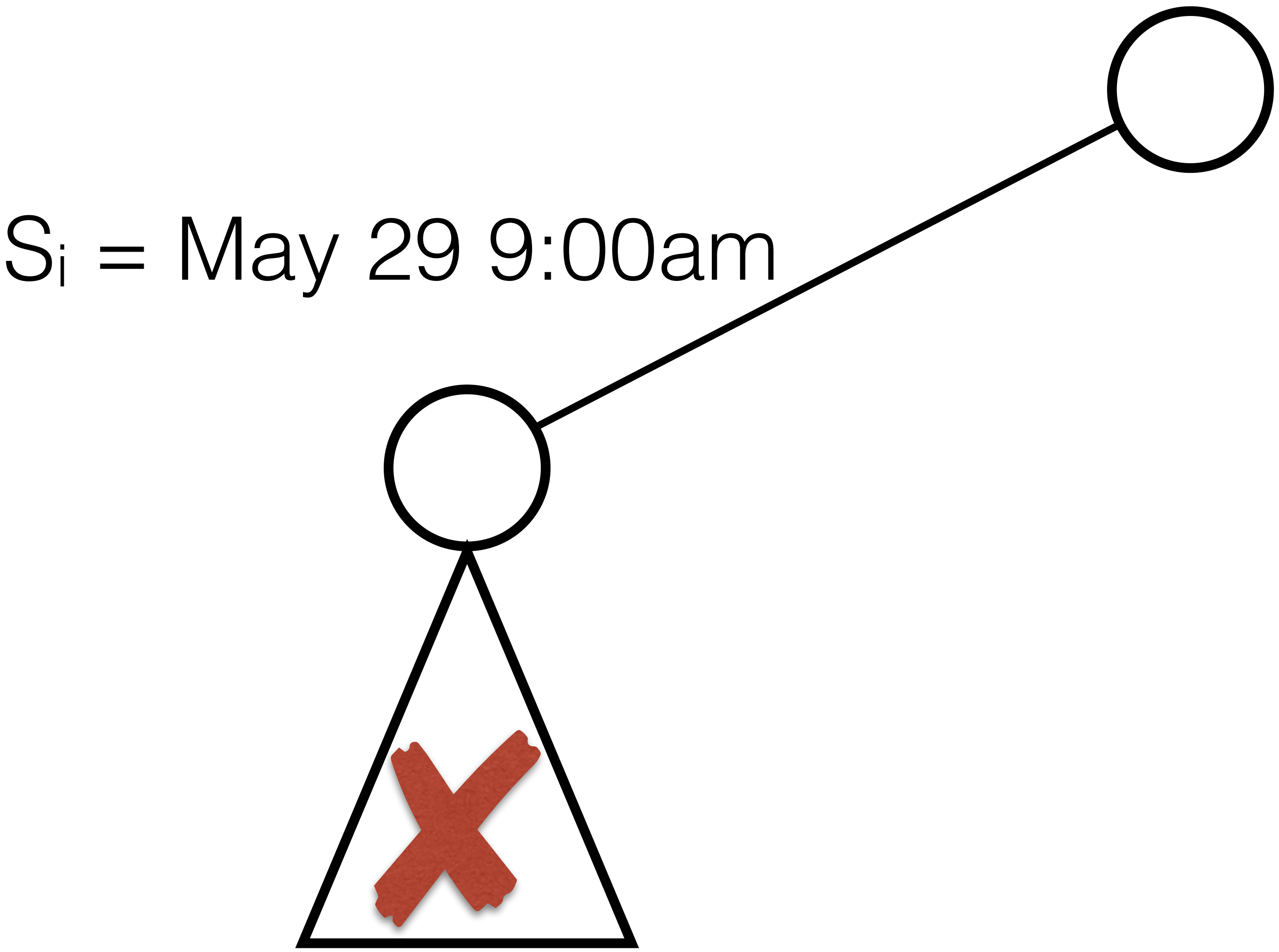
$x_3 \in \{1, \cancel{2}, \cancel{3}\}$

$x_4 \in \{\cancel{1}, 2, \cancel{3}\}$

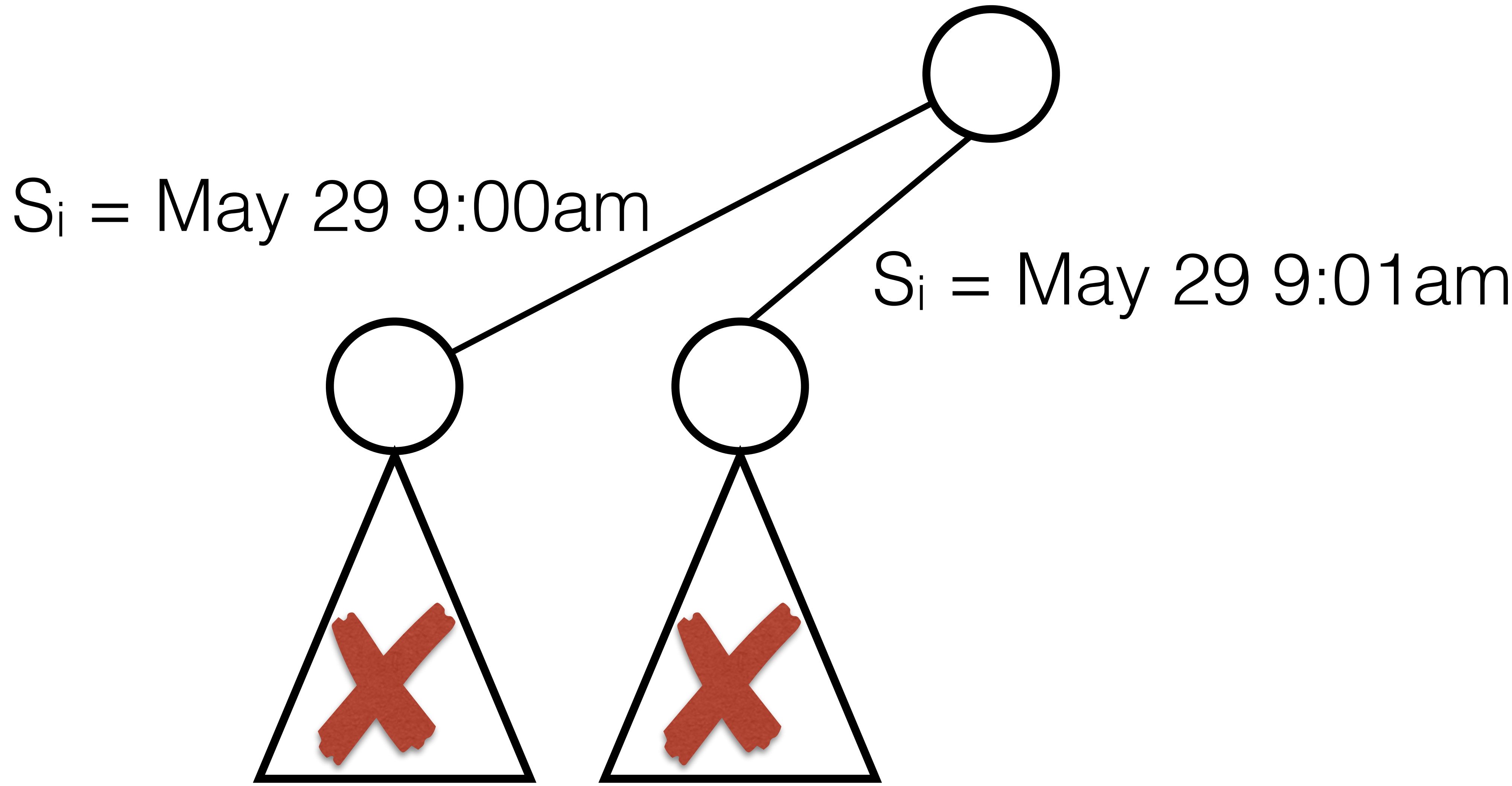
Learnt clause: $\neg(x_4 \leq 2)$

$\neg(x_2 \leq 2) \vee \neg(x_3 \leq 2) \vee \neg(x_1 = 1)$

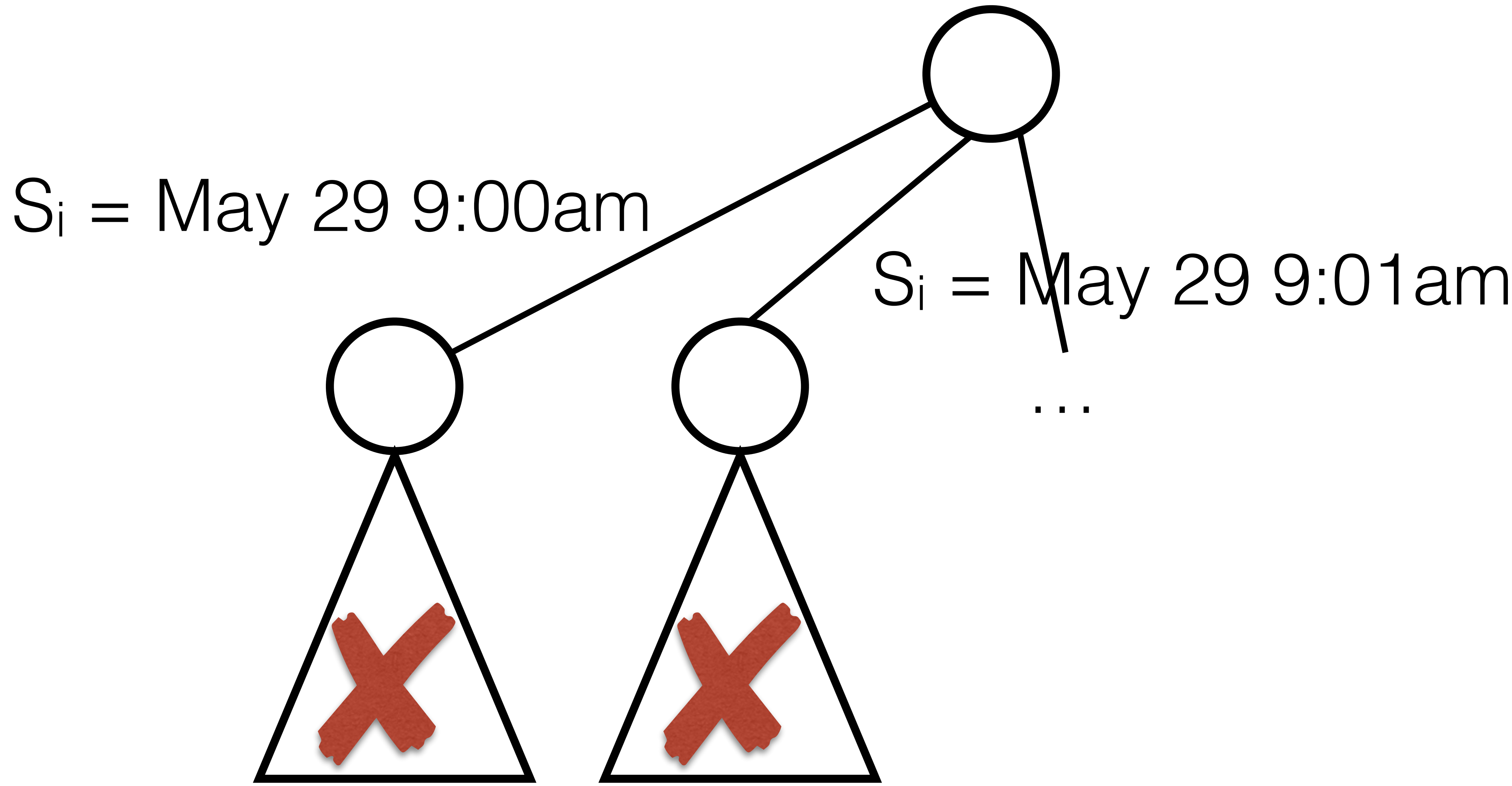
Impact on Scheduling



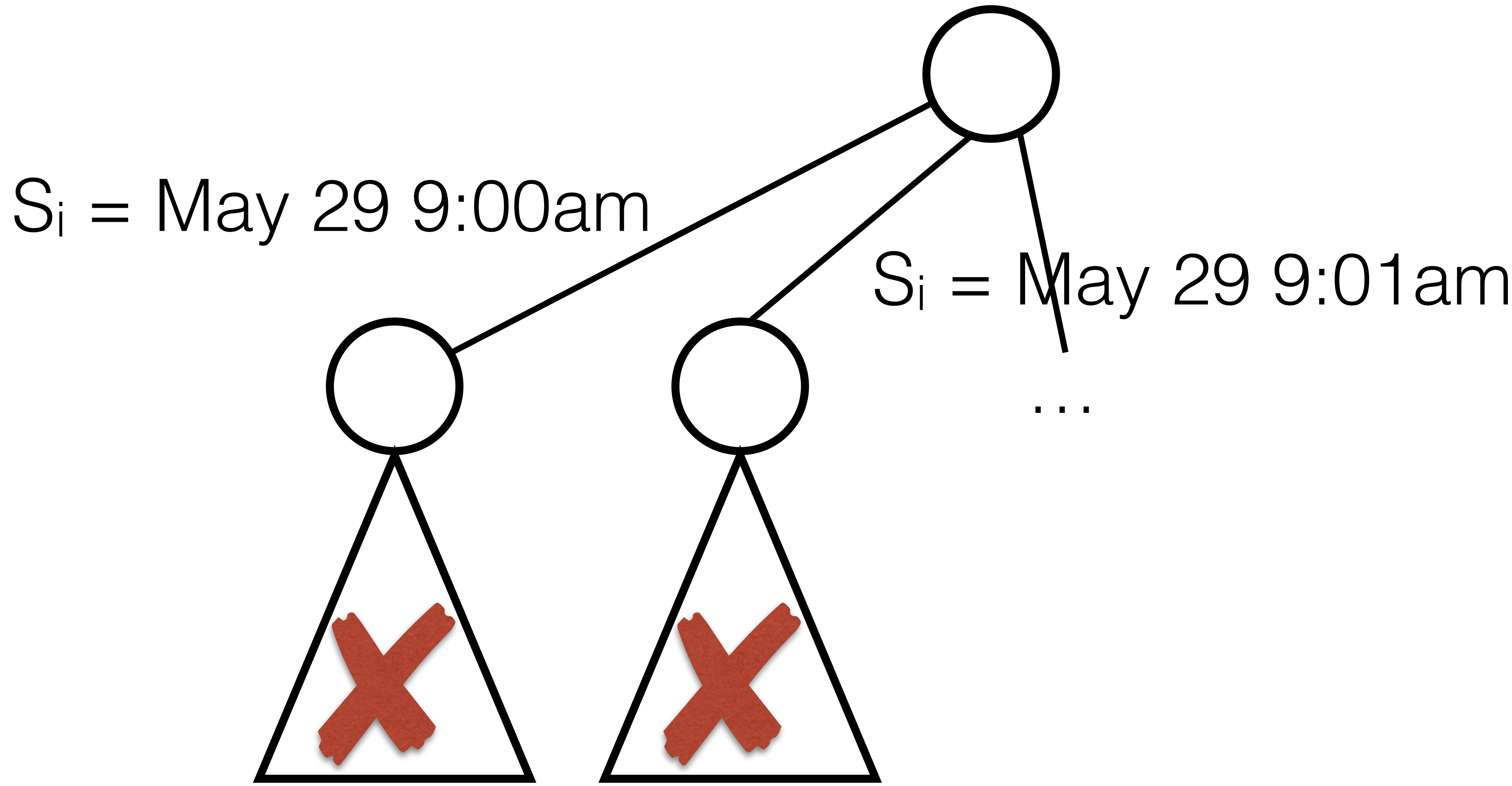
Impact on Scheduling



Impact on Scheduling



Impact on Scheduling



- Nogood: $S_i >$ May 29 13h00

Nogood vs Cutting Planes

- If the task A does not start before June 23rd, the task B is postponed to June 25th.

Nogood vs Cutting Planes

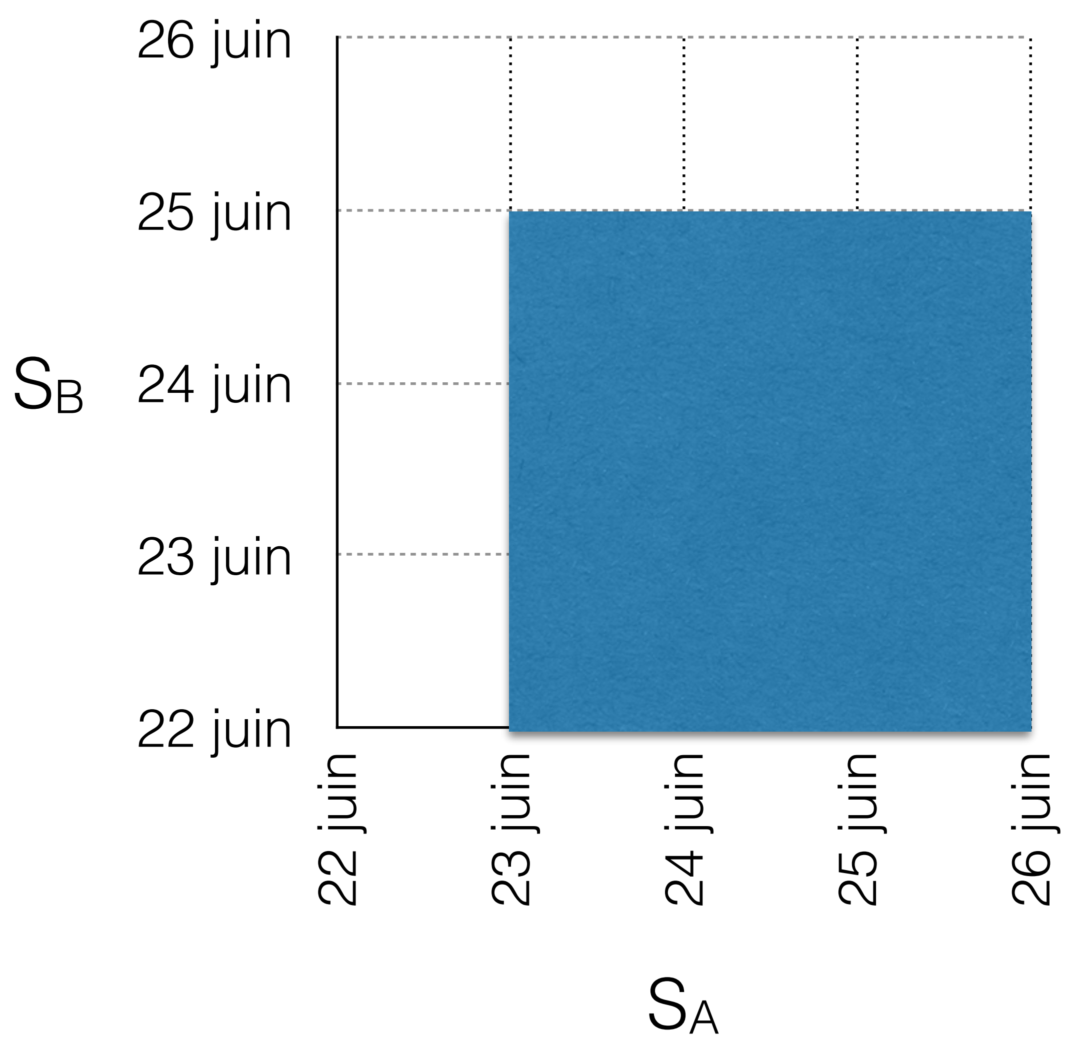
- If the task A does not start before June 23rd, the task B is postponed to June 25th.

$S_A \geq \text{June } 23^{rd} \Longrightarrow S_B \geq \text{June } 25^{th}$

Nogood vs Cutting Planes

- If the task A does not start before June 23rd, the task B is postponed to June 25th.

$S_A \geq$ June $23^{rd} \Longrightarrow S_B \geq$ June 25^{th}

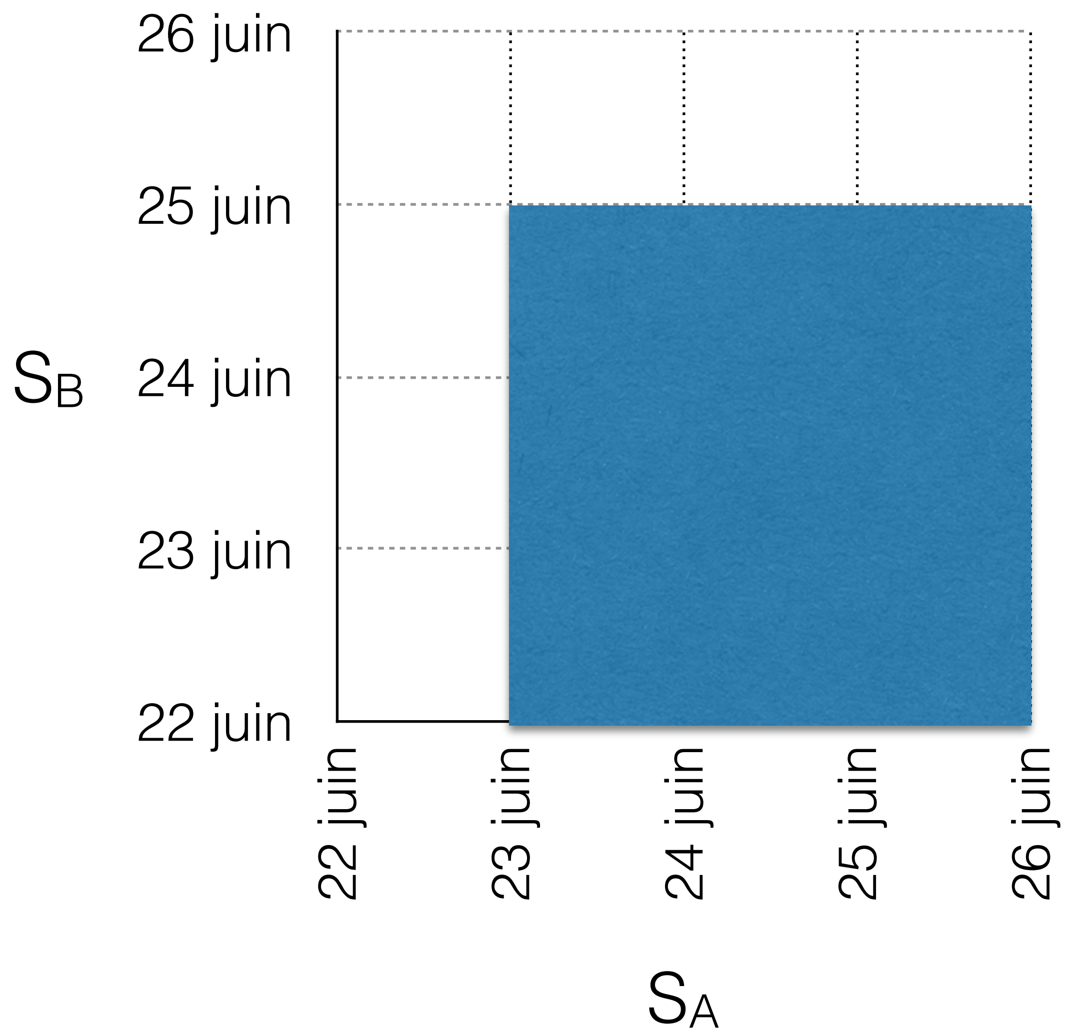


Nogood vs Cutting Planes

- If the task A does not start before June 23rd, the task B is postponed to June 25th.

$S_A \geq$ June 23$^{rd} \Longrightarrow S_B \geq$ June 25th

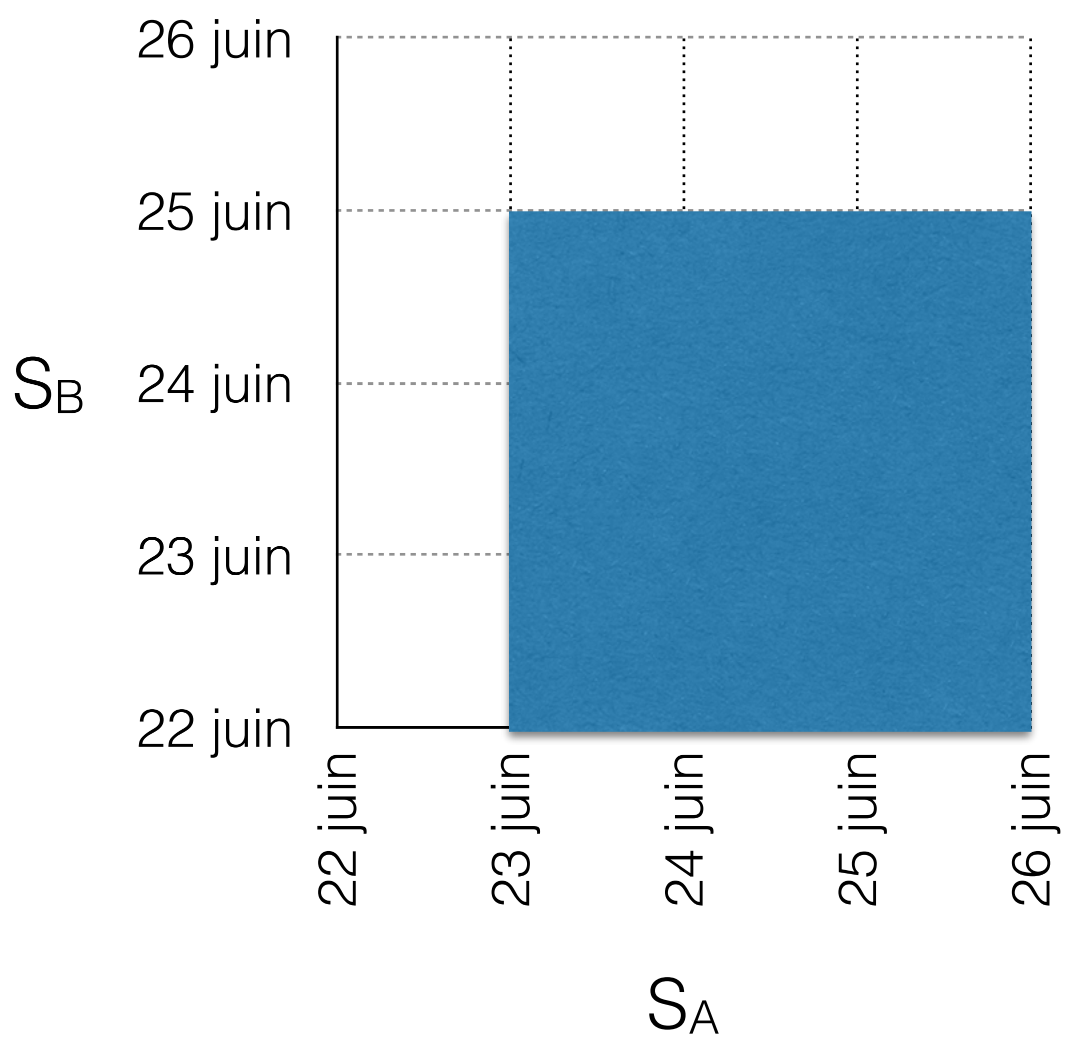
$S_A - S_B \geq 1$ day



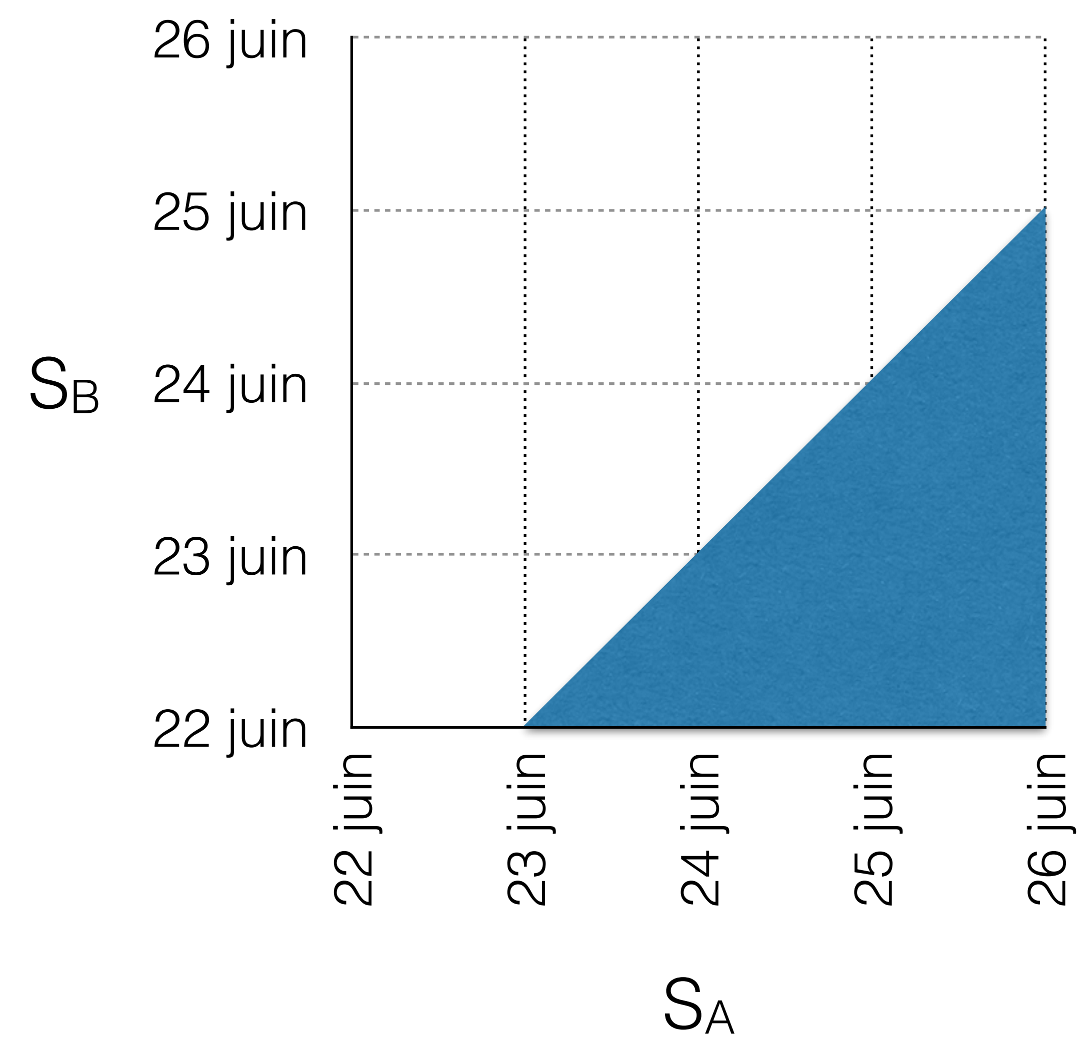
Nogood vs Cutting Planes

- If the task A does not start before June 23rd, the task B is postponed to June 25th.

$S_A \geq$ June 23rd $\Longrightarrow S_B \geq$ June 25th



$S_A - S_B \geq 1$ day



How to scale?

Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems

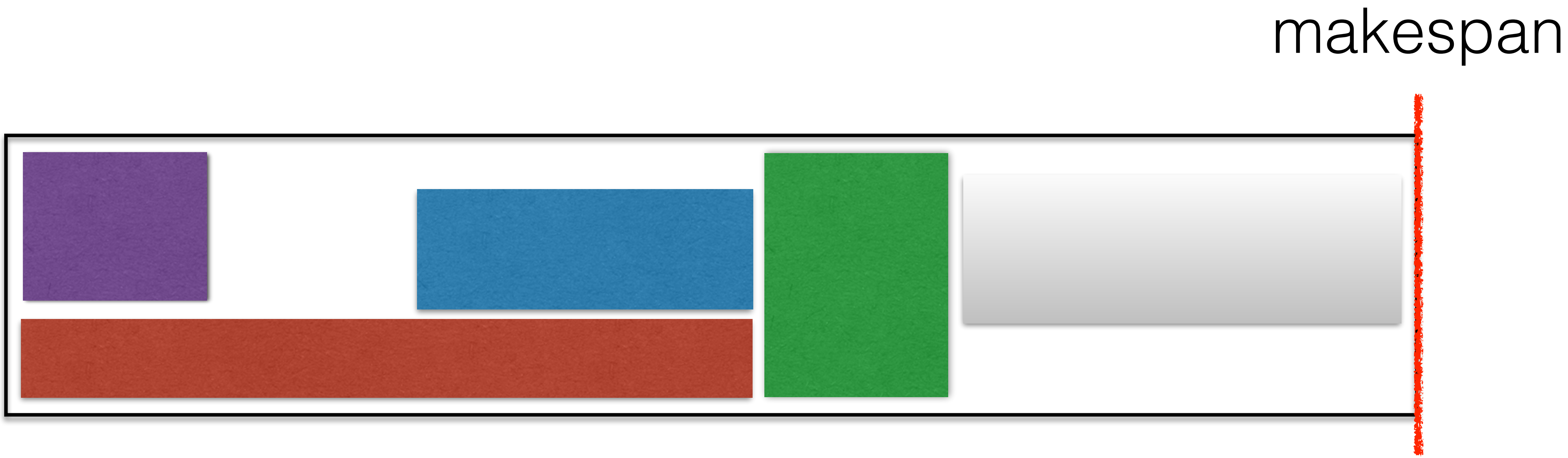
Paul Shaw*

ILOG S.A.
9, rue de Verdun, BP 85
94253 Gentilly Cedex, FRANCE.
shaw@ilog.fr

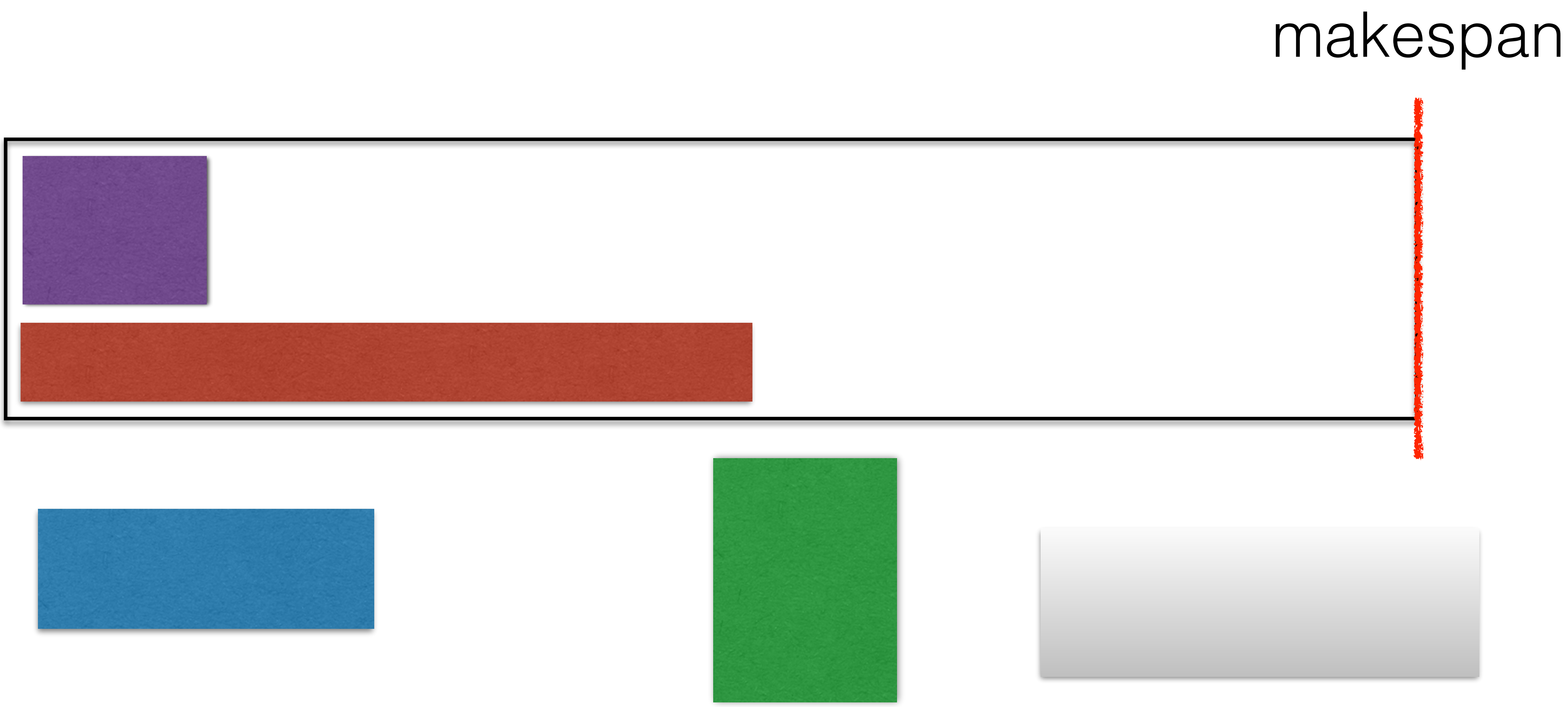
Abstract. We use a local search method we term Large Neighbourhood Search (LNS) to solve vehicle routing problems. LNS is analogous to the shuffling technique of job-shop scheduling, and so meshes well with constraint programming technology. LNS explores a large neighbourhood of

CP'98

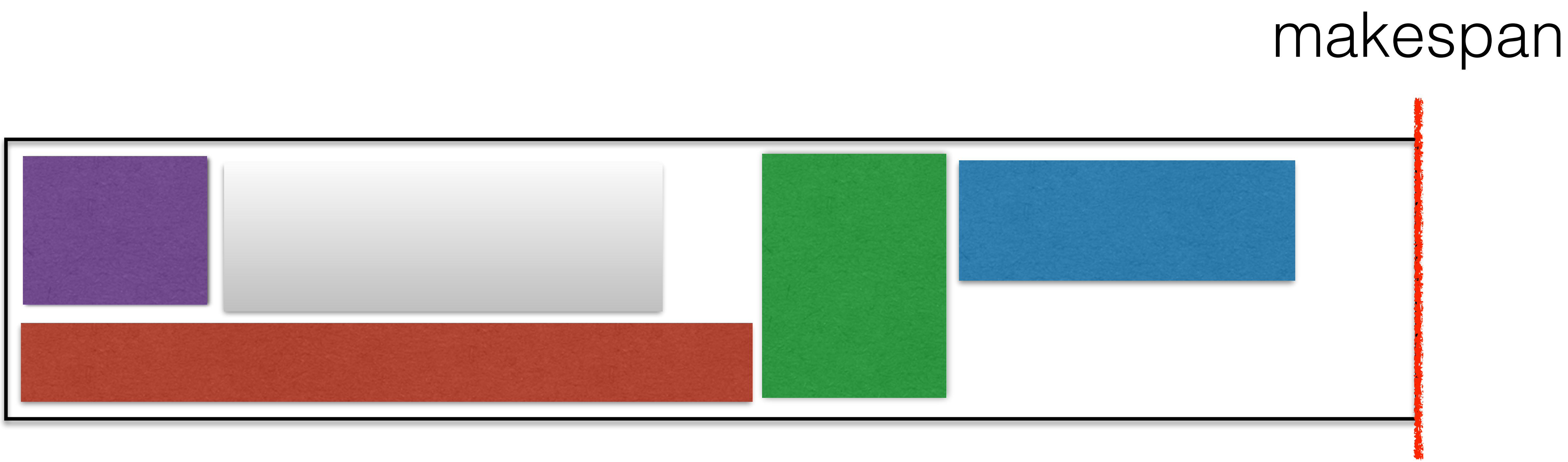
Large Neighbourhood Search



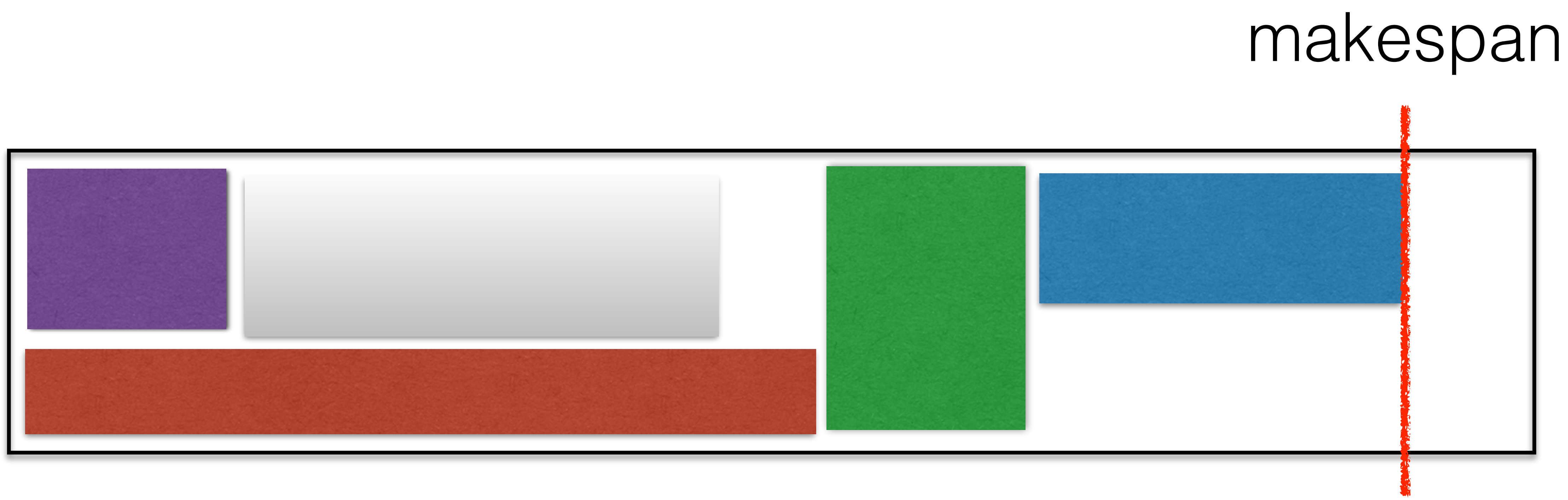
Large Neighbourhood Search



Large Neighbourhood Search



Large Neighbourhood Search



How to implement a LNS?

Actes JFPC 2017

Une simple heuristique pour rapprocher DFS et LNS pour les COP

Julien Vion   Sylvain Piechowiak

Université de Valenciennes et du Hainaut Cambrésis
LAMIH CNRS UMR 8201
{julien.vion, sylvain.piechowiak}@univ-valenciennes.fr

Résumé

Dans cet article, nous montrons comment une combinaison de stratégies de branchement et de redémarrages pour la recherche en profondeur d'abord (DFS) permet de reproduire le fonctionnement de la recherche par grand voisinage (LNS) pour la résolution de problèmes d'optimisation à contraintes, ce qui permet de rapprocher considérablement les deux techniques. En particulier, nous pouvons implémenter une stratégie DFS qui bénéficie des propriétés de passage à l'échelle de LNS tout en étant capable de prouver l'optimalité des solutions.

hybride [16] qui réalise des « déplacements » itératifs de manière similaire à une recherche locale, mais utilise une DFS et la propagation de contraintes pour améliorer la meilleure solution connue [17]. L'idée de LNS est de *relâcher* la meilleure solution connue en restaurant le domaine d'une partie de ses variables. Le « *fragment* » obtenu est alors *réoptimisé* par une DFS. Comme la plupart des stratégies incomplètes, LNS passe bien mieux à l'échelle qu'une DFS classique, mais elle ne peut pas prouver l'optimalité d'une solution (ou l'inconsistance d'un problème). De plus le choix des fragments à réoptimiser est une tâche difficile

JFPC 2017

Solution-Based Phase Saving for CP: A Value-Selection Heuristic to Simulate Local Search Behavior in Complete Solvers

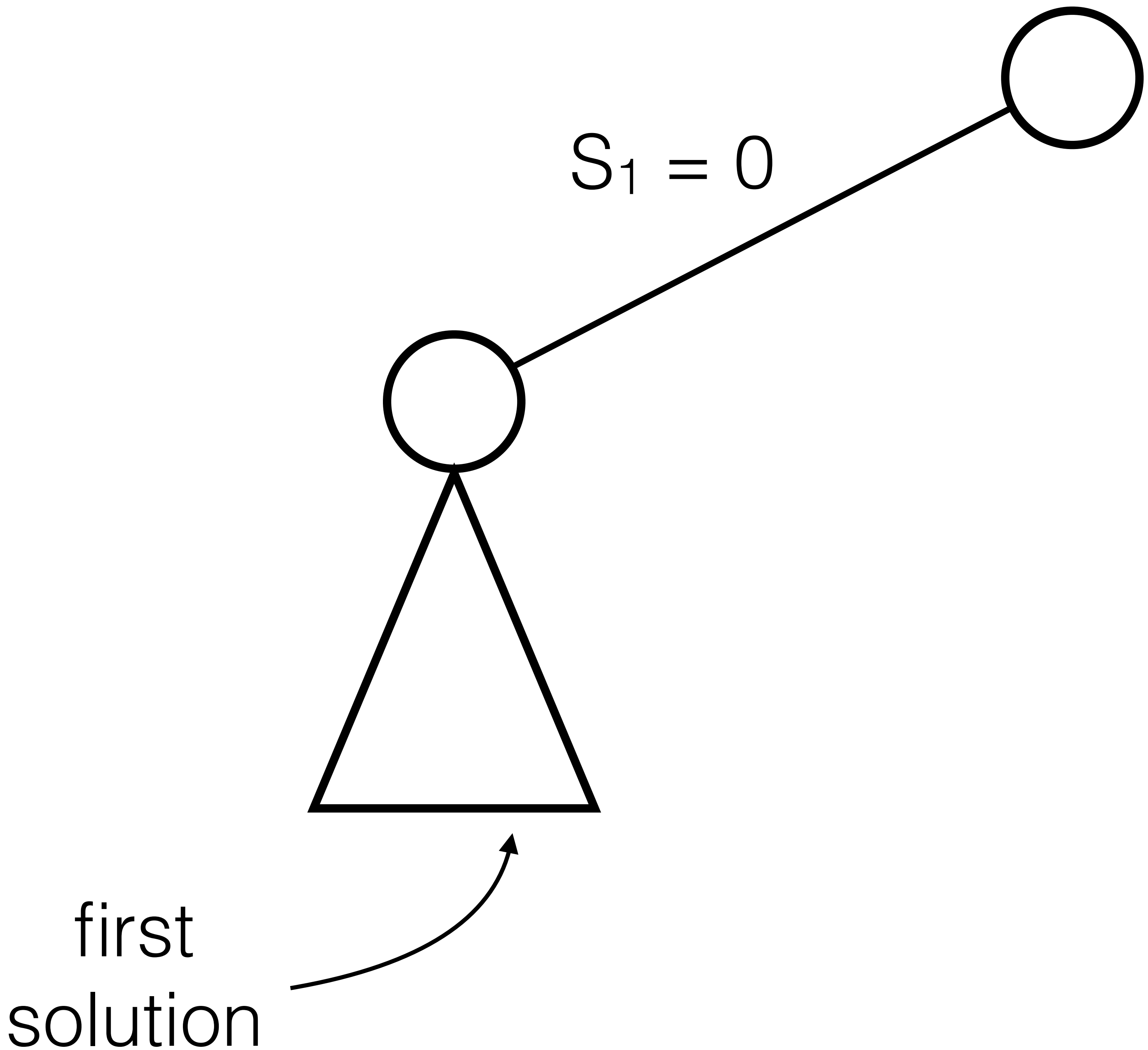
Emir Demirović[(✉)], Geoffrey Chu, and Peter J. Stuckey

School of Computing and Information Systems, University of Melbourne, Melbourne, Australia
{emir.demirovic,pstuckey}@unimelb.edu.au

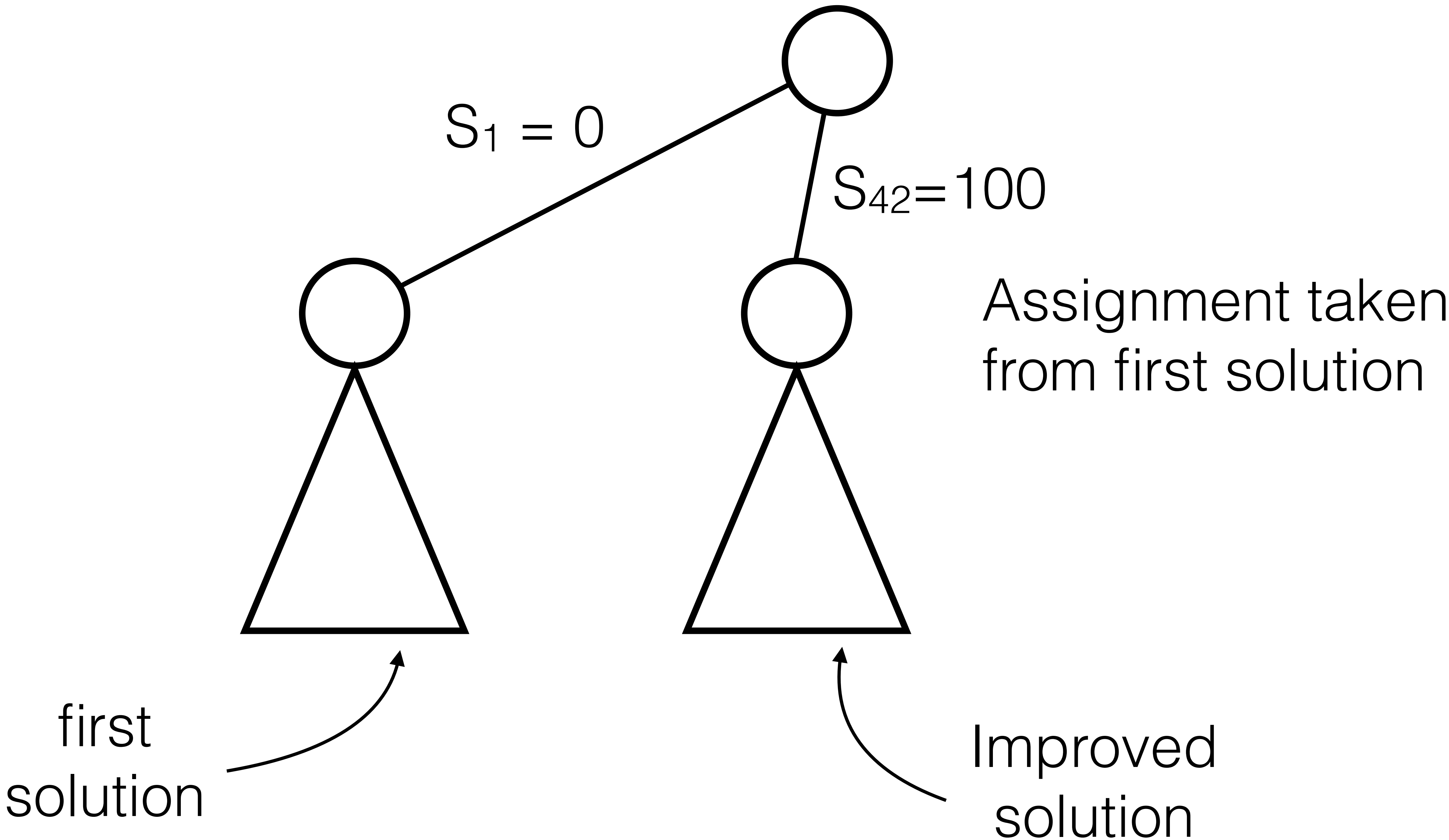
Abstract. Large neighbourhood search, a meta-heuristic, has proven to be successful on a wide range of optimisation problems. The algorithm repeatedly generates and searches through a neighbourhood around the current best solution. Thus, it finds increasingly better solutions by solv-

CP 2018

How to implement a LNS?



How to implement a LNS?



Which tool should I use?

- First choose the solver that performs the best for your type of problem.
- Solver competition results can help you identify the solver.
    - MiniZinc challenge
    - XCSP Competitions
- Do not only look at the general ranking. Look at the scores on specific problems.

Constraint Solvers

- ACE
- BTD
- Choco
- Chuffed
- CoSoCo
- CpoFzn
- Eschequer
- Fun-sCOP
- Geas
- Gecode
- Glasgow
- CP Optimizer
- JaCoP
- MiniCPBP
- Mistral
- MZN/CPLEX
- MZN/Cbc
- MZN/Gurobi
- MZN/HiGHS
- MZN/SCIP
- Nacre
- OR-Tools
- OscaR
- Picat
- PicatSAT
- RBO
- SICStus Prolog
- Sat4j-CSP-PB
- SeaPearl
- Yuck
- flatzingo
- iZplus
- toulbar2

Modelling language

Modelling language

- Do use a modelling language.

Modelling language

- Do use a modelling language.
  - The development is much faster even if you have no prior knowledge on the language.

Modelling language

- Do use a modelling language.
    - The development is much faster even if you have no prior knowledge on the language.
    - It allows to quickly readjust your choice of solver.

Modelling language

- Do use a modelling language.
  - The development is much faster even if you have no prior knowledge on the language.
  - It allows to quickly readjust your choice of solver.
  - Some IDE come with a search tree visualisation tool.

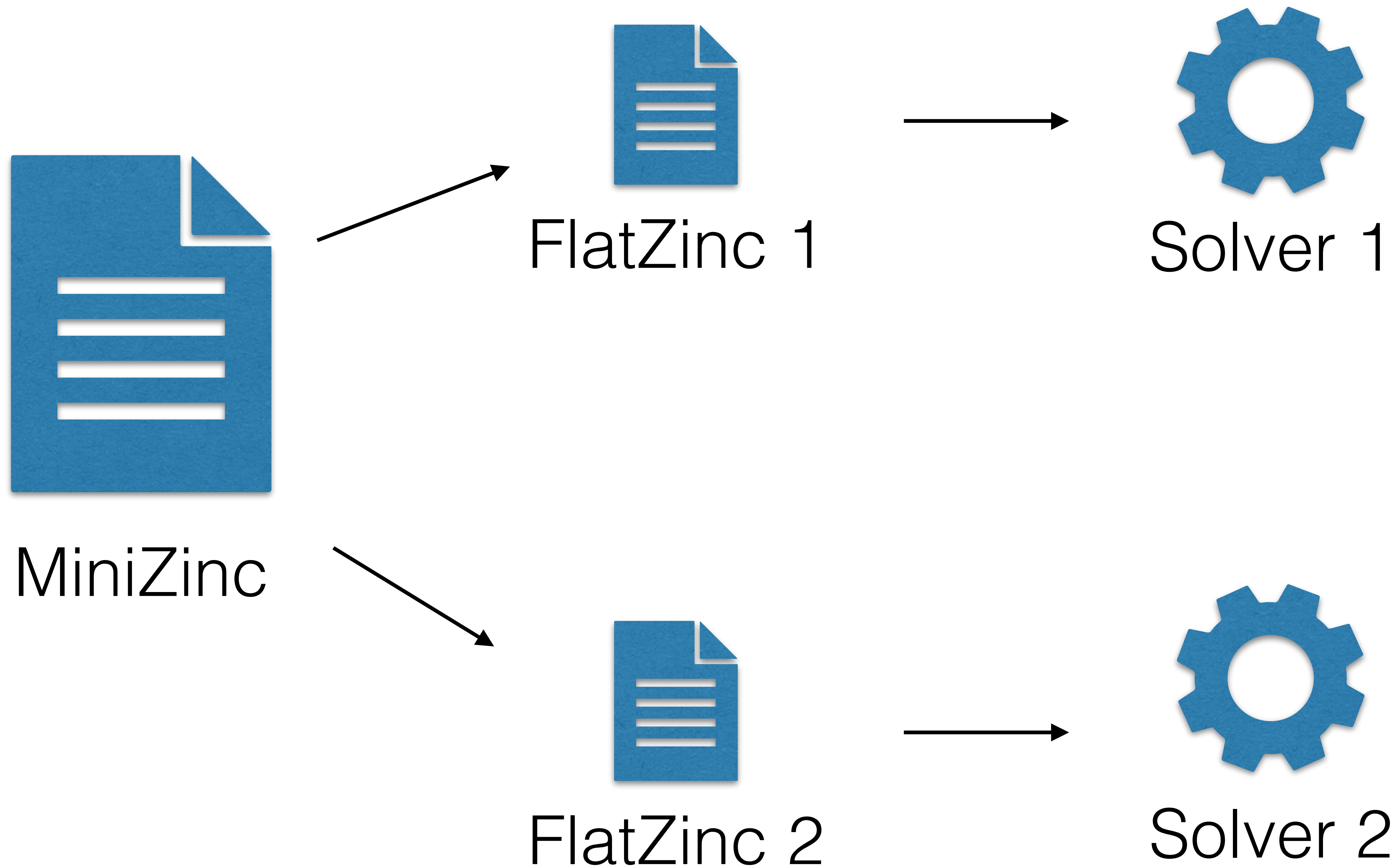
Modelling language

- Do use a modelling language.
  - The development is much faster even if you have no prior knowledge on the language.
  - It allows to quickly readjust your choice of solver.
  - Some IDE come with a search tree visualisation tool.
  - Some IDE come with a tool for detecting sets of contradictory constraints

Modelling language

- Do use a modelling language.
  - The development is much faster even if you have no prior knowledge on the language.
  - It allows to quickly readjust your choice of solver.
  - Some IDE come with a search tree visualisation tool.
  - Some IDE come with a tool for detecting sets of contradictory constraints
- Down side: branching heuristics are limited

MiniZinc



Demo

Result

- Instance: J60_14_3 from PSPLib
- 60 tasks and 4 ressources

Solver	Option	Temps
Gecode	Default	> 1h 30m

Result

- Instance: J60_14_3 from PSPLib
- 60 tasks and 4 ressources

Solver	Option	Temps
Gecode	Default	> 1h 30m
Chuffed	Time Tabling Check (Default)	2m 08s

Result

- Instance: J60_14_3 from PSPLib
- 60 tasks and 4 ressources

Solver	Option	Temps
Gecode	Default	> 1h 30m
Chuffed	Time Tabling Check (Default)	2m 08s
Chuffed	Time Tabling Check & Filtering Time-Table Edge-Finder Check & Filtering	4m05s

Result

- Instance: J60_14_3 from PSPLib
- 60 tasks and 4 ressources

Solver	**Option**	**Temps**
Gecode	Default	> 1h 30m
Chuffed	Time Tabling Check (Default)	2m 08s
Chuffed	Time Tabling Check & Filtering Time-Table Edge-Finder Check & Filtering	4m05s
Chuffed	Time Tabling Check SBPS	46s

Result

- Instance: J60_14_3 from PSPLib
- 60 tasks and 4 ressources

Solver	Option	Temps
Gecode	Default	> 1h 30m
Chuffed	Time Tabling Check (Default)	2m 08s
Chuffed	Time Tabling Check & Filtering Time-Table Edge-Finder Check & Filtering	4m05s
Chuffed	Time Tabling Check SBPS	46s
Chuffed	Time Tabling Check & Filtering Time-Table Edge-Finder Check & Filtering SBPS	5s

Conclusion

- Constraint programming is based on a very simple exploration of a search tree.
- Filtering algorithms play a major role in reducing the size of the search tree, hence the importance of global constraints.
- Nogood learning prevents the solver from repeating the same mistakes
- Large Neighbourhood Search allows to scale over larger instances.
  - SBPS allows to retrieve the optimal solution.
- There exist multiple constraint solvers (commercial and open source)