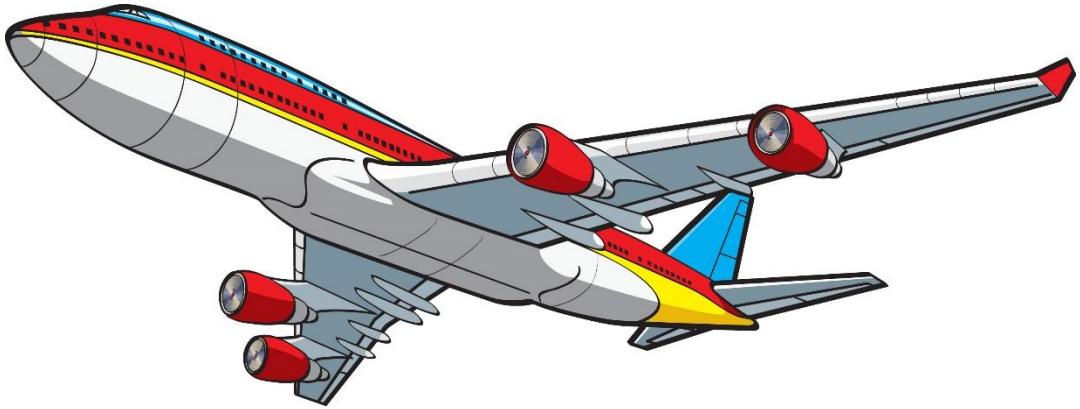
Projection, Inference, and Consistency

John Hooker
Carnegie Mellon University

IJCAI 2016, New York City

A high-level presentation.

Don’t worry about the details.



Projection as a Unifying Concept

- **Projection** underlies **inference**, **optimization**, and **consistency maintenance**

Projection as a Unifying Concept

- **Projection** underlies **inference**, **optimization**, and **consistency maintenance**
  - **Logical inference** is **projection** onto a subset of variables.

Projection as a Unifying Concept

- **Projection** underlies **inference**, **optimization**, and **consistency maintenance**
  - **Logical inference** is **projection** onto a subset of variables.
  - **Optimization** is **projection** onto one or more variables.

Projection as a Unifying Concept

- **Projection** underlies **inference**, **optimization**, and **consistency maintenance**
  - **Logical inference** is **projection** onto a subset of variables.
  - **Optimization** is **projection** onto one or more variables.
  - **Consistency maintenance** lies at the heart of **constraint programming** and is a form of **projection**.

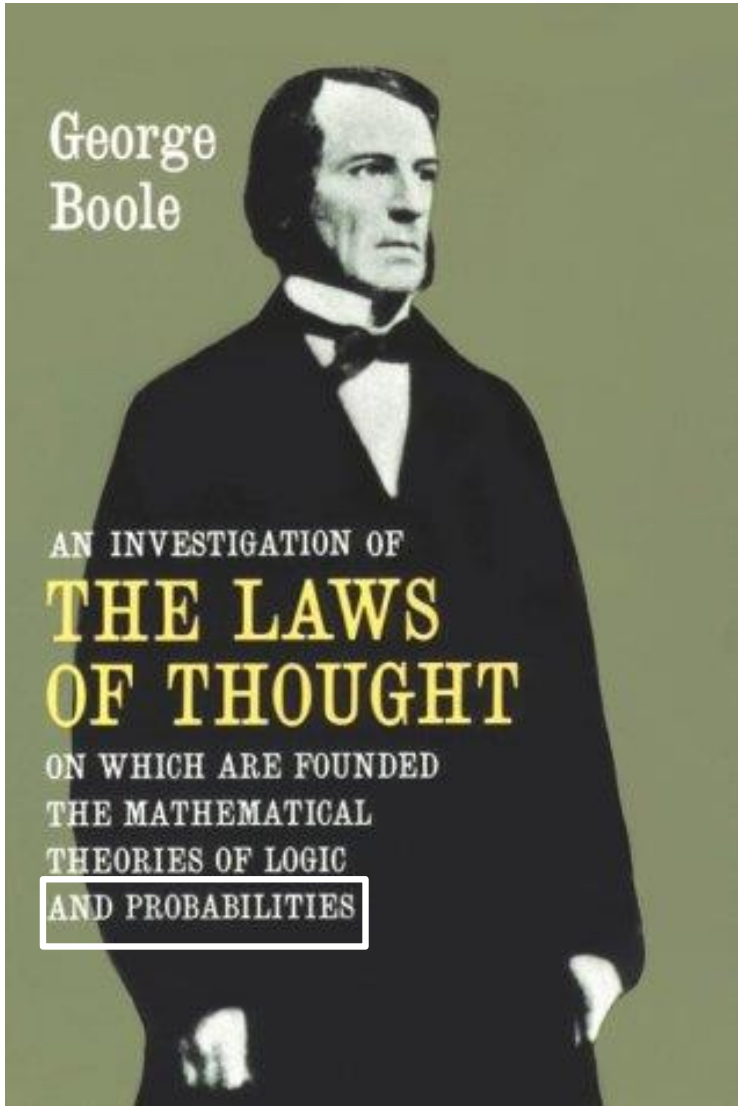
Projection as a Unifying Concept

- **Projection** underlies **inference**, **optimization**, and **consistency maintenance**
  - **Logical inference** is **projection** onto a subset of variables.
  - **Optimization** is **projection** onto one or more variables.
  - **Consistency maintenance** lies at the heart of **constraint programming** and is a form of **projection**.

- Let's start with **George Boole**…
  - …whose **probability logic** brings together projection, inference, and optimization.

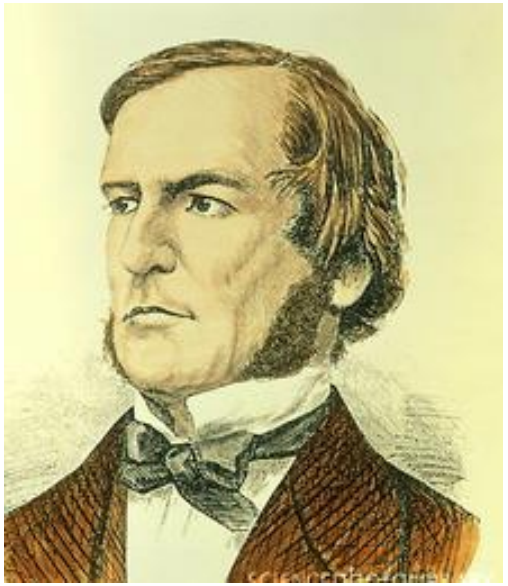
Probability Logic

- George Boole is best known for propositional logic and Boolean algebra.
  - But he proposed a highly original approach to **probability logic**.



Logical Inference

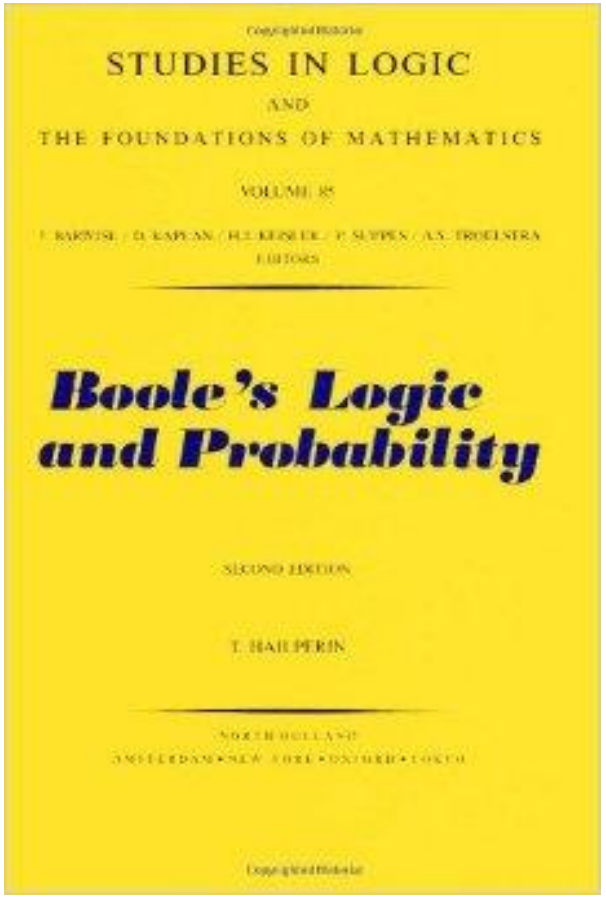
- The problem:
  - Given a set *S* of propositions
    - Each with a given probability.
  - And a proposition *P* that can be deduced from *S*…
  - What probability can be assigned to *P*?



Boole's Probability Logic

- In 1970s, Theodore Hailperin offered an interpretation of Boole's probability logic
  - …based on modern concept of **linear programming**.
  - LP first clearly formulated in 1930s by Kantorovich.

Hailperin (1976)



Boole's Probability Logic

Example

Clause	Probability
x_1	0.9
if x_1 then x_2	0.8
if x_2 then x_3	0.4

Deduce probability range for x_3

Boole's Probability Logic

Example

Clause	Probability
x_1	0.9
if x_1 then x_2	0.8
if x_2 then x_3	0.4

Interpret if-then statements as material conditionals

Deduce probability range for x_3

Boole's Probability Logic

Example

Clause	Probability
x_1	0.9
$\bar{x}_1 \vee x_2$	0.8
$\bar{x}_2 \vee x_3$	0.4

Interpret if-then statements as material conditionals

Deduce probability range for x_3

Boole's Probability Logic

Example

Clause	Probability
x_1	0.9
$\bar{x}_1 \vee x_2$	0.8
$\bar{x}_2 \vee x_3$	0.4

Deduce probability range for x_3

Linear programming model

$$\min/\max \ \pi_0$$

$$\begin{bmatrix} 01010101 \\ 00001111 \\ 11110011 \\ 11011101 \\ 11111111 \end{bmatrix} \begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ \vdots \\ p_{111} \end{bmatrix} = \begin{bmatrix} \pi_0 \\ 0.9 \\ 0.8 \\ 0.4 \\ 1 \end{bmatrix}$$

p_{000} = probability that $(x_1,x_2,x_3) = (0,0,0)$

Boole's Probability Logic

Example

Clause	Probability
x_1	0.9
$\bar{x}_1 \vee x_2$	0.8
$\bar{x}_2 \vee x_3$	0.4

Deduce probability range for x_3

Linear programming model

$$\min/\max \ \pi_0$$

$$\begin{bmatrix} 01010101 \\ 00001111 \\ 11110011 \\ 11011101 \\ 11111111 \end{bmatrix} \begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ \vdots \\ p_{111} \end{bmatrix} = \begin{bmatrix} \pi_0 \\ 0.9 \\ 0.8 \\ 0.4 \\ 1 \end{bmatrix}$$

p_{000} = probability that $(x_1,x_2,x_3) = (0,0,0)$

Solution: $\pi_0 \in [0.1, 0.4]$

Boole's Probability Logic

- This LP model was re-invented in AI community.

  Nilsson (1986)

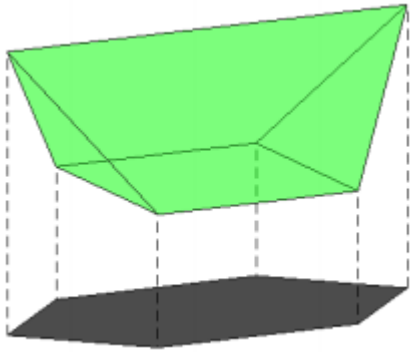
- Column generation methods are now available.
  - To deal with exponential number of variables.

Boole's Probability Logic

- **Inference:** infer probability of a proposition.
- **Projection:** project feasible set onto a variable
  - To derive range of probabilities
- **Optimization:** formulate as a linear programming problem.

Optimization as Projection

- An LP can be solved by **Fourier elimination**
  - The only known method in Boole's day
  - This is a **projection** method.



Fourier (1827)

Optimization as Projection

- Eliminate variables we want to project out.
  - To solve $\min/\max\{y_0 \mid y_0 = ay, Ay \geq b\}$

    project out all variables y_j except y_0.

Optimization as Projection

- Eliminate variables we want to project out.
  - To solve $\min/\max\{y_0 \mid y_0 = ay, Ay \geq b\}$

    project out all variables y_j except y_0.
  - To project out y_j, eliminate it from pairs of inequalities:

$$c\bar{y} + c_0 y_j \geq \gamma \qquad d\bar{y} - d_0 y_j \geq \delta$$

where $c_0, d_0 \geq 0$

Optimization as Projection

- Eliminate variables we want to project out.
  - To solve $\min/\max\{y_0 \mid y_0 = ay, Ay \geq b\}$

    project out all variables y_j except y_0.
  - To project out y_j, eliminate it from pairs of inequalities:

$$c\bar{y} + c_0 y_j \geq \gamma \qquad d\bar{y} - d_0 y_j \geq \delta$$

$$-\frac{c}{c_0}\bar{y} + \frac{\gamma}{c_0} \leq y_j \leq \frac{d}{d_0}\bar{y} - \frac{\delta}{d_0}$$

Optimization as Projection

- Eliminate variables we want to project out.
  - To solve $\min/\max\{y_0 \mid y_0 = ay, Ay \geq b\}$

    project out all variables y_j except y_0.
  - To project out y_j, eliminate it from pairs of inequalities:

$$c\bar{y} + c_0 y_j \geq \gamma \qquad d\bar{y} - d_0 y_j \geq \delta$$

$$-\frac{c}{c_0}\bar{y} + \frac{\gamma}{c_0} \leq \cancel{y_j} \leq \frac{d}{d_0}\bar{y} - \frac{\delta}{d_0}$$

Optimization as Projection

- Eliminate variables we want to project out.
  - To solve $\min/\max\{y_0 \mid y_0 = ay, Ay \geq b\}$

    project out all variables y_j except y_0.
  - To project out y_j, eliminate it from pairs of inequalities:

$$c\bar{y} + c_0 y_j \geq \gamma \qquad d\bar{y} - d_0 y_j \geq \delta$$

$$-\frac{c}{c_0}\bar{y} + \frac{\gamma}{c_0} \leq \cancel{y_j} \leq \frac{d}{d_0}\bar{y} - \frac{\delta}{d_0}$$

$$\left(\frac{c}{c_0} + \frac{d}{d_0}\right)\bar{y} \geq \frac{\gamma}{c_0} + \frac{\delta}{d_0}$$

Inference as Projection

- Project onto propositional variables of interest
  - Suppose we wish to infer from these clauses everything we can about propositions x_1, x_2, x_3

$$
\begin{array}{llll}
x_1 & & \vee x_4 & \vee x_5 \\
x_1 & & \vee x_4 & \vee \bar{x}_5 \\
x_1 & & \vee x_5 & \vee x_6 \\
x_1 & & \vee x_5 & \vee \bar{x}_6 \\
 & x_2 & \vee \bar{x}_5 & \vee x_6 \\
 & x_2 & \vee \bar{x}_5 & \vee \bar{x}_6 \\
 & x_3 & \vee \bar{x}_4 & \vee x_5 \\
 & x_3 & \vee \bar{x}_4 & \vee \bar{x}_5
\end{array}
$$

Inference as Projection

- Project onto propositional variables of interest
  - Suppose we wish to infer from these clauses everything we can about propositions x_1, x_2, x_3

We can deduce

$$x_1 \vee x_2$$

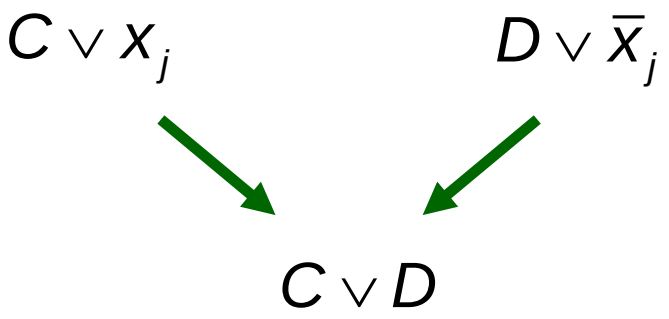
$$x_1 \vee x_3$$

This is a projection onto x_1, x_2, x_3

$$\begin{array}{lll}
x_1 & & \vee\, x_4 \vee x_5 \\
x_1 & & \vee\, x_4 \vee \bar{x}_5 \\
x_1 & & \vee\, x_5 \vee x_6 \\
x_1 & & \vee\, x_5 \vee \bar{x}_6 \\
 & x_2 & \vee\, \bar{x}_5 \vee x_6 \\
 & x_2 & \vee\, \bar{x}_5 \vee \bar{x}_6 \\
 & & x_3 \vee \bar{x}_4 \vee x_5 \\
 & & x_3 \vee \bar{x}_4 \vee \bar{x}_5
\end{array}$$

Inference as Projection

- Resolution is a projection method!
  - Similar to Fourier elimination
    - Actually, **identical** to Fourier elimination + **rounding**!
  - To project out x_j, eliminate it from pairs of clauses:



Quine (1952,1955)
JH (1992,2012)

Inference as Projection

- Projection methods similar to Fourier elimination
  - Resolution for logical clauses — Quine (1952,1955)
  - …for cardinality clauses — JH (1988)
  - …for 0-1 linear inequalities — JH (1992)
  - …for general integer linear inequalities — Williams & JH (2015)

Inference as Projection

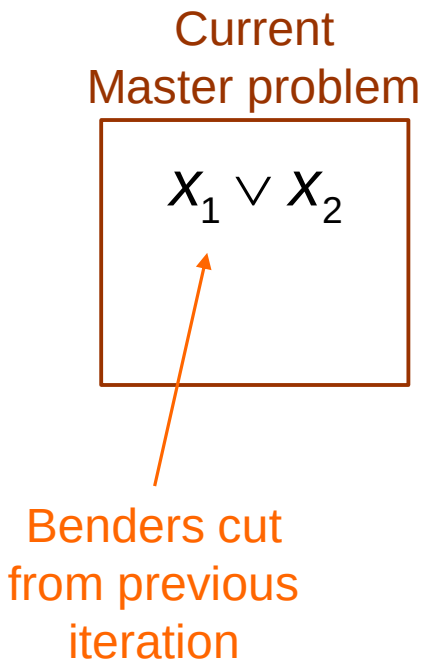
- Elimination-based methods tend to be **slow.**
- Another approach is an **optimization method**: logic-based **Benders decomposition**

Inference as Projection

- Elimination-based methods tend to be **slow.**
- Another approach is an **optimization method**: logic-based **Benders decomposition**
  - For problems that radically simplify when certain variables are fixed.
  - Resulting **subproblem** may be easy to solve.
  - Solve **master problem** to **search** over ways to fix variables.
  - Solution of subproblem yields a **Benders cut** that is added to master problem and helps direct search.
  - **Repeat** until problem is solved.

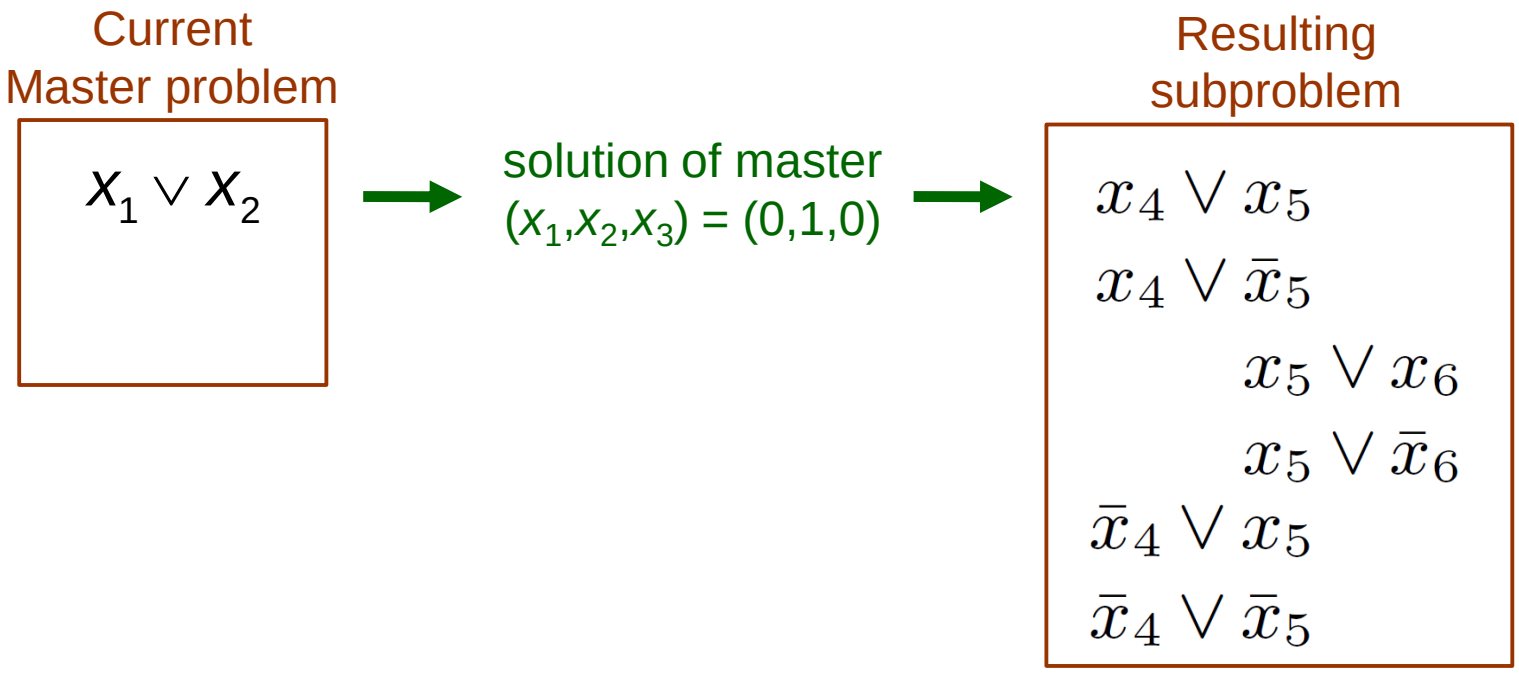
Benders Decomposition as Projection

- Benders decomposition computes a **projection!**
  - Benders cuts describe projection onto master problem variables.



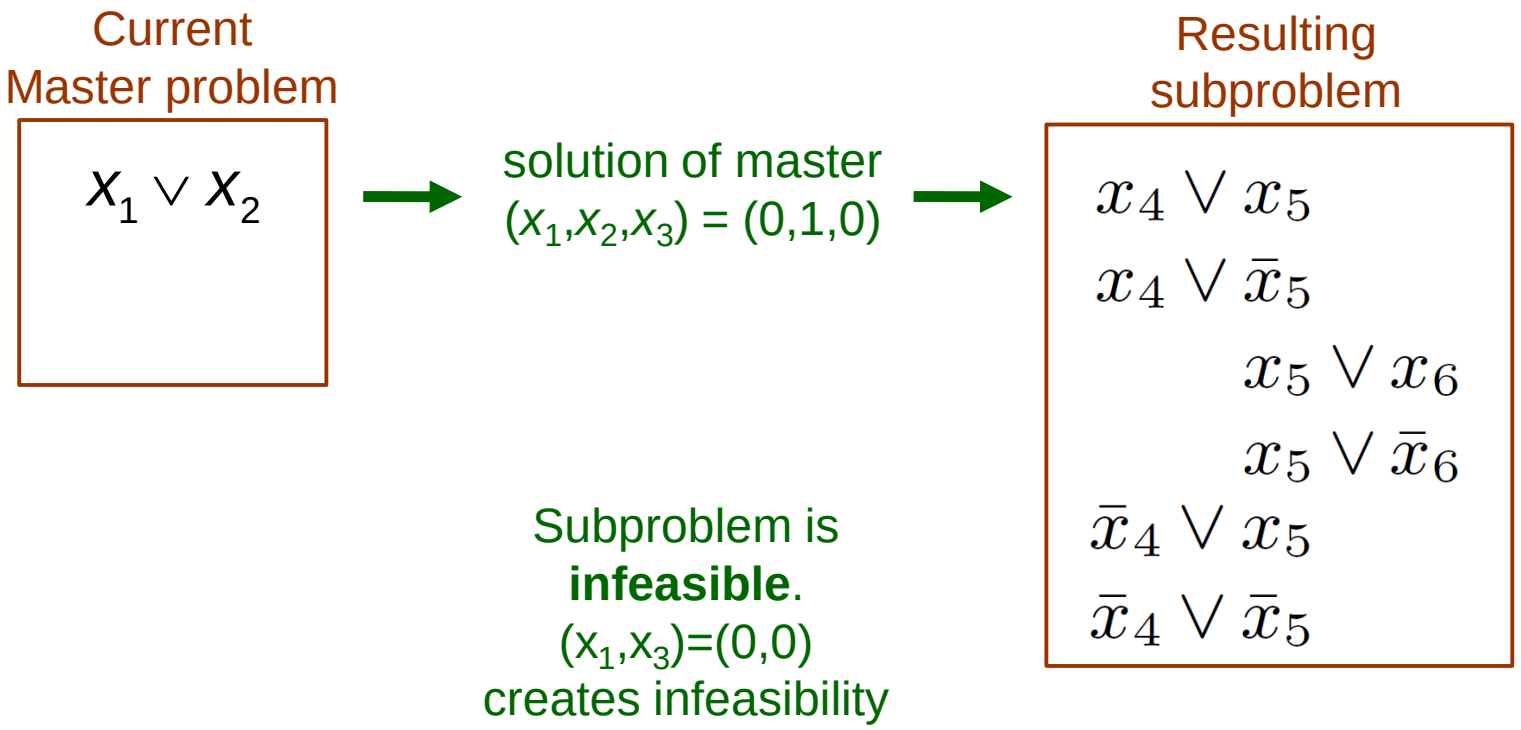
Benders Decomposition as Projection

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.



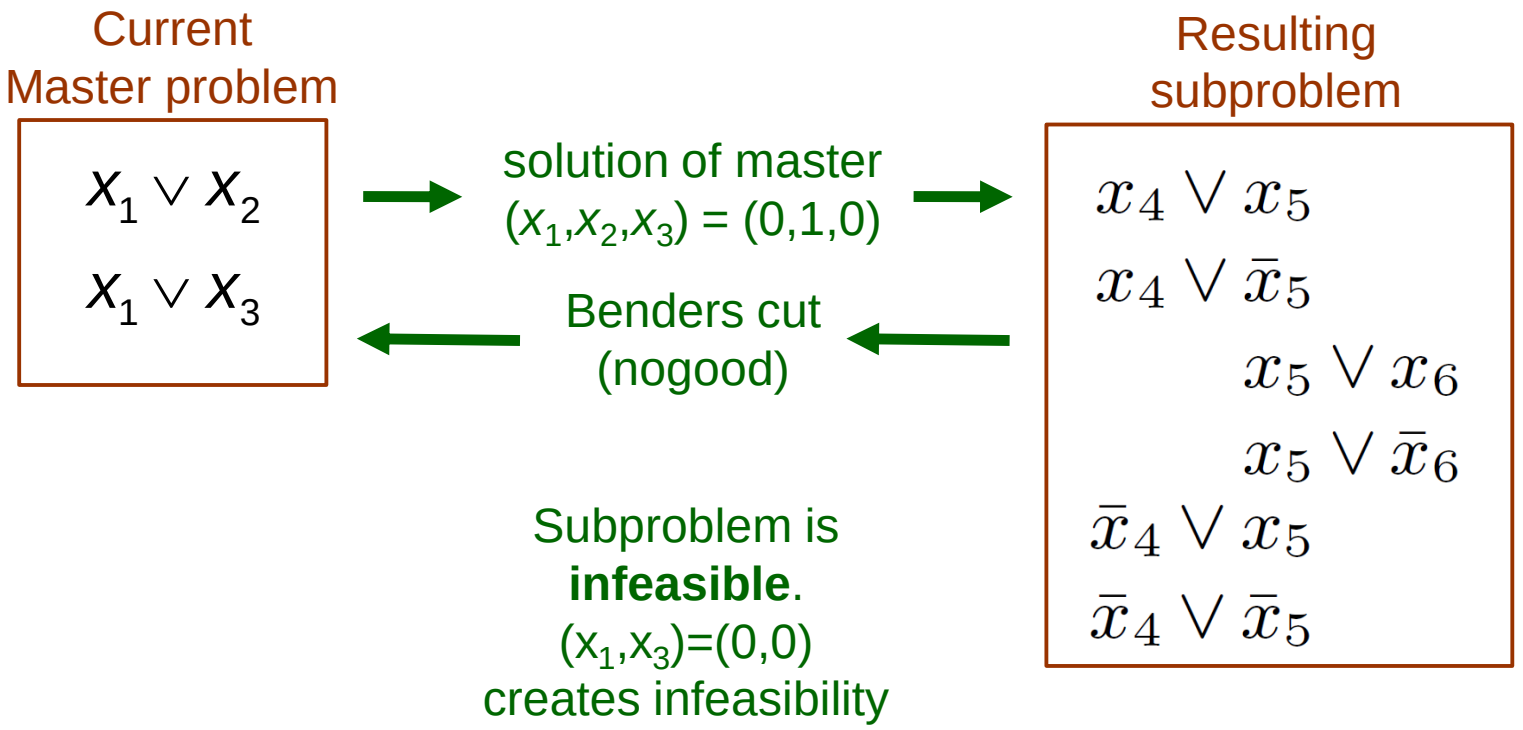
Benders Decomposition as Projection

- Benders decomposition computes a **projection!**
  - Benders cuts describe projection onto master problem variables.



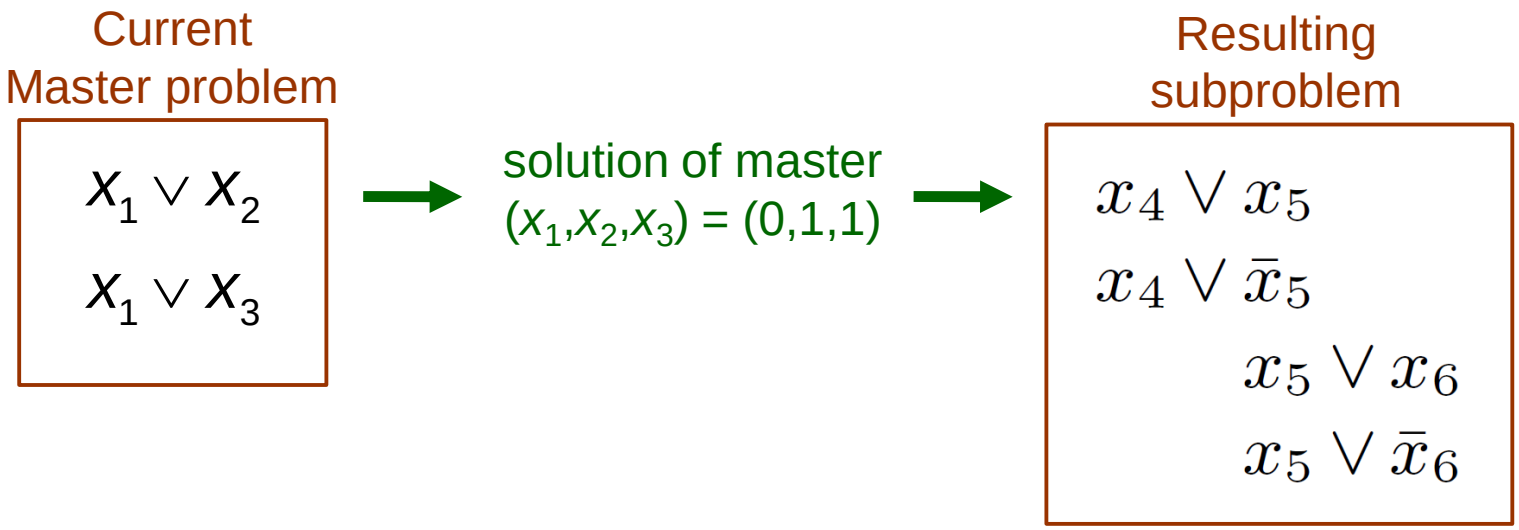
Benders Decomposition as Projection

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.



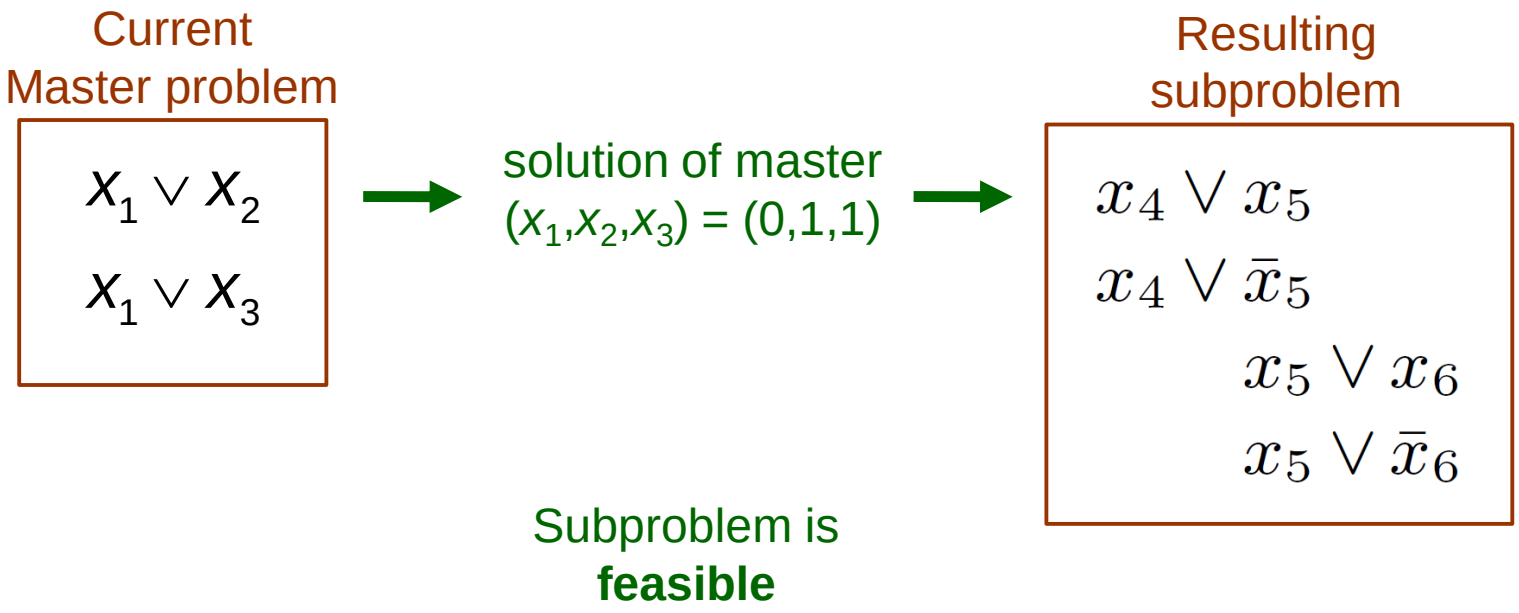
Benders Decomposition as Projection

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.



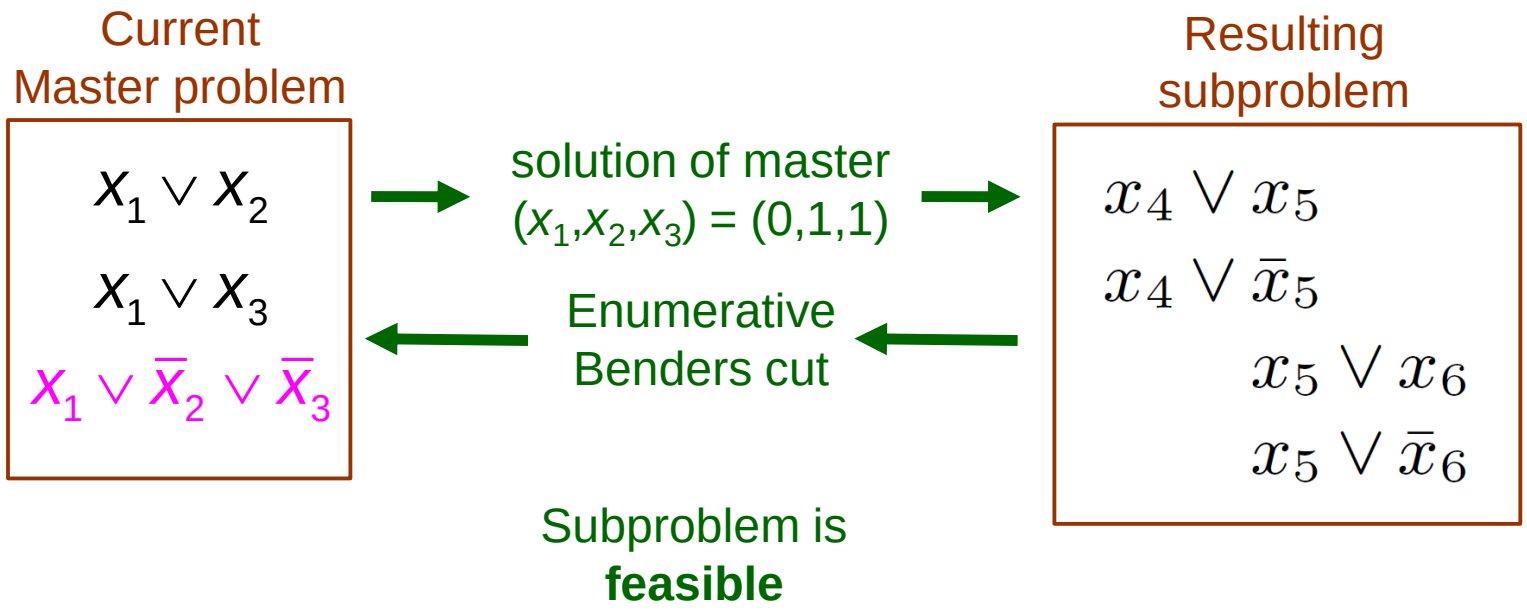
Benders Decomposition as Projection

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.



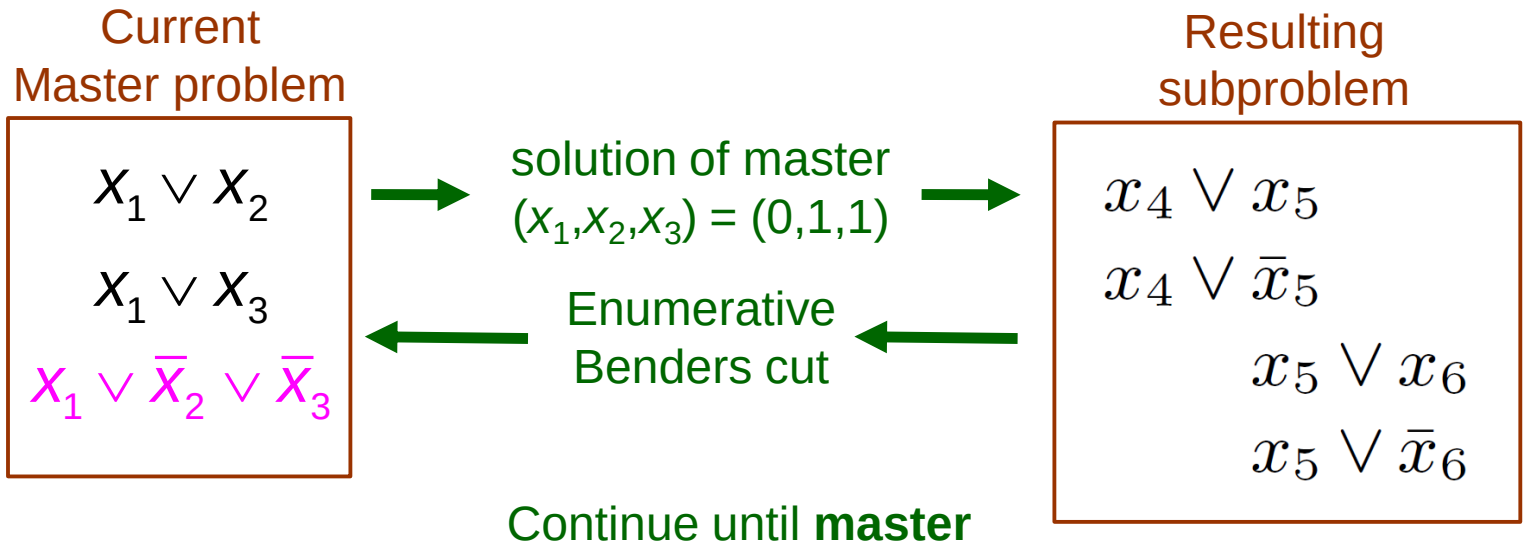
Benders Decomposition as Projection

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.



Benders Decomposition as Projection

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.



Continue until **master** is infeasible.

Black Benders cuts describe projection.

JH (2000)
JH and Ottosson (2003)

Projection in Satisfiability Solvers

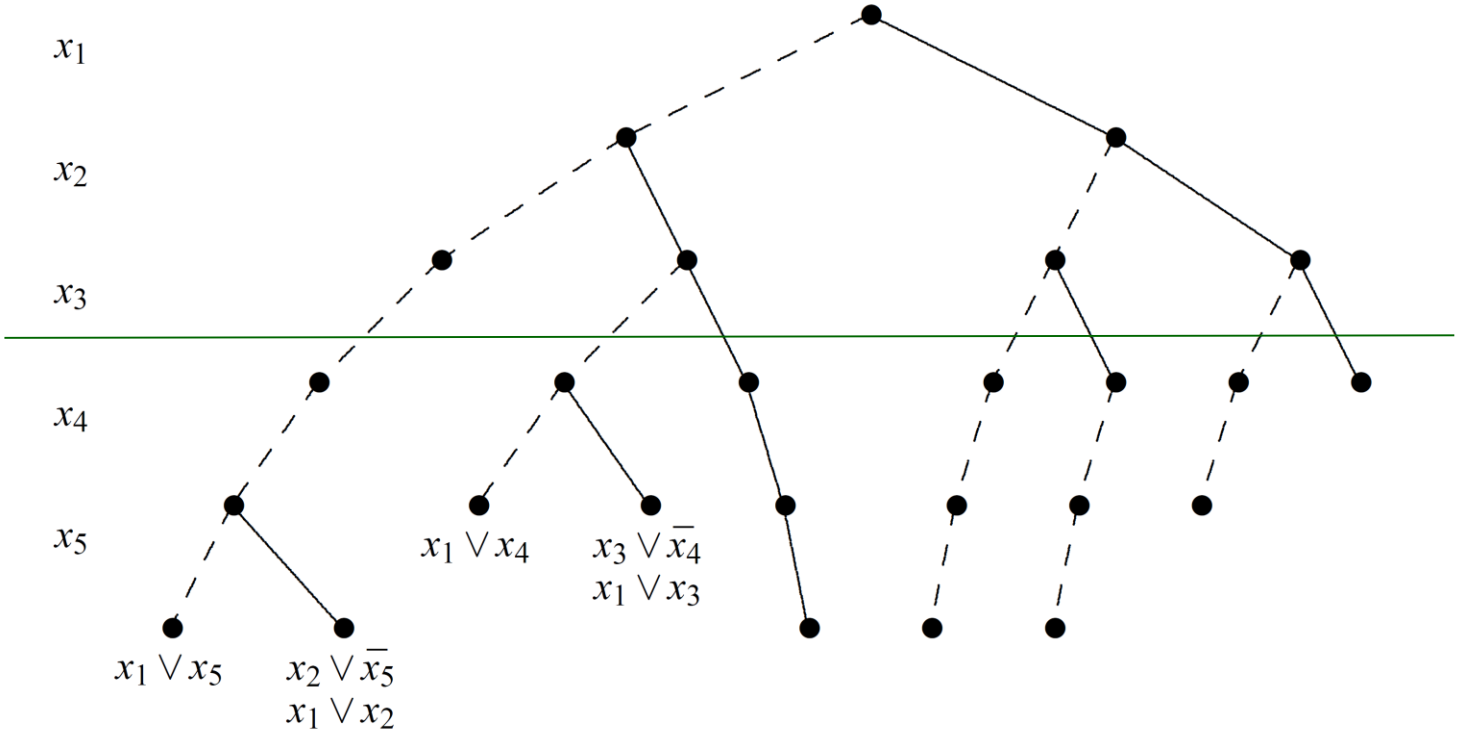
- **Satisfiability solvers** solve inference problems by a **projection** method.
  - Using **conflict clauses**.

Projection in Satisfiability Solvers

- **Satisfiability solvers** solve inference problems by a **projection** method.
  - Using **conflict clauses**.
- They do so by generating **Benders cuts!**
  - Conflict clauses = Benders cuts

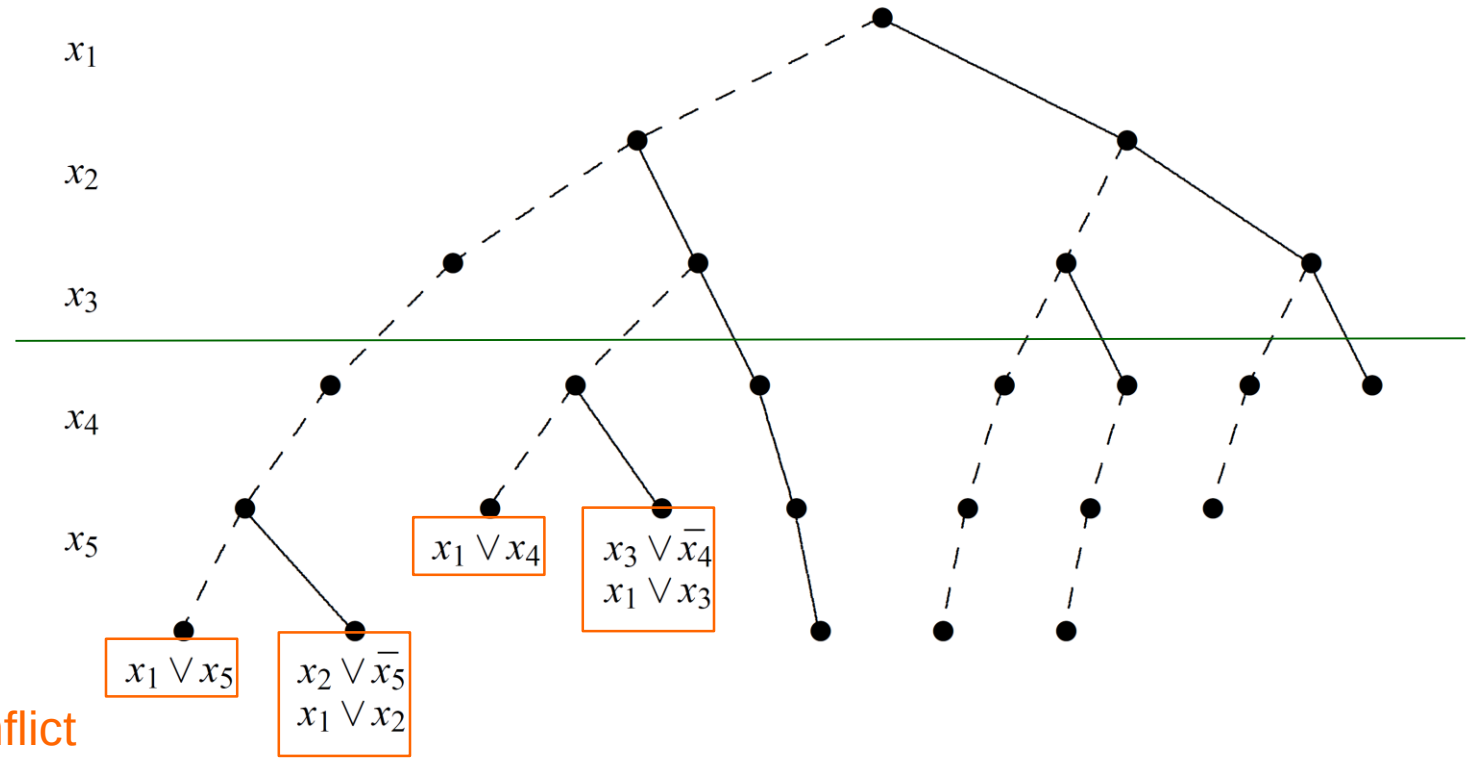
Projection in Satisfiability Solvers

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Projection in Satisfiability Solvers

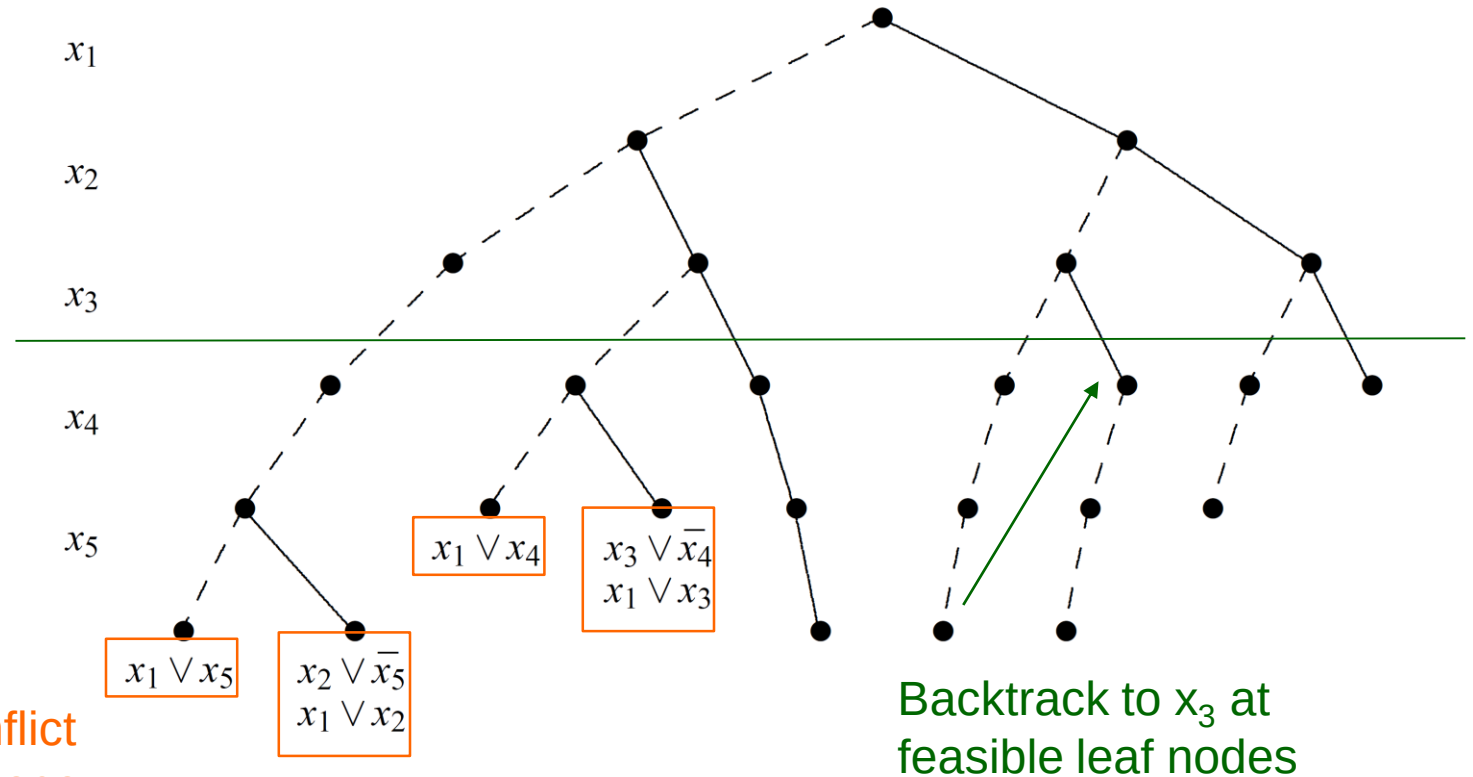
- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Conflict clauses

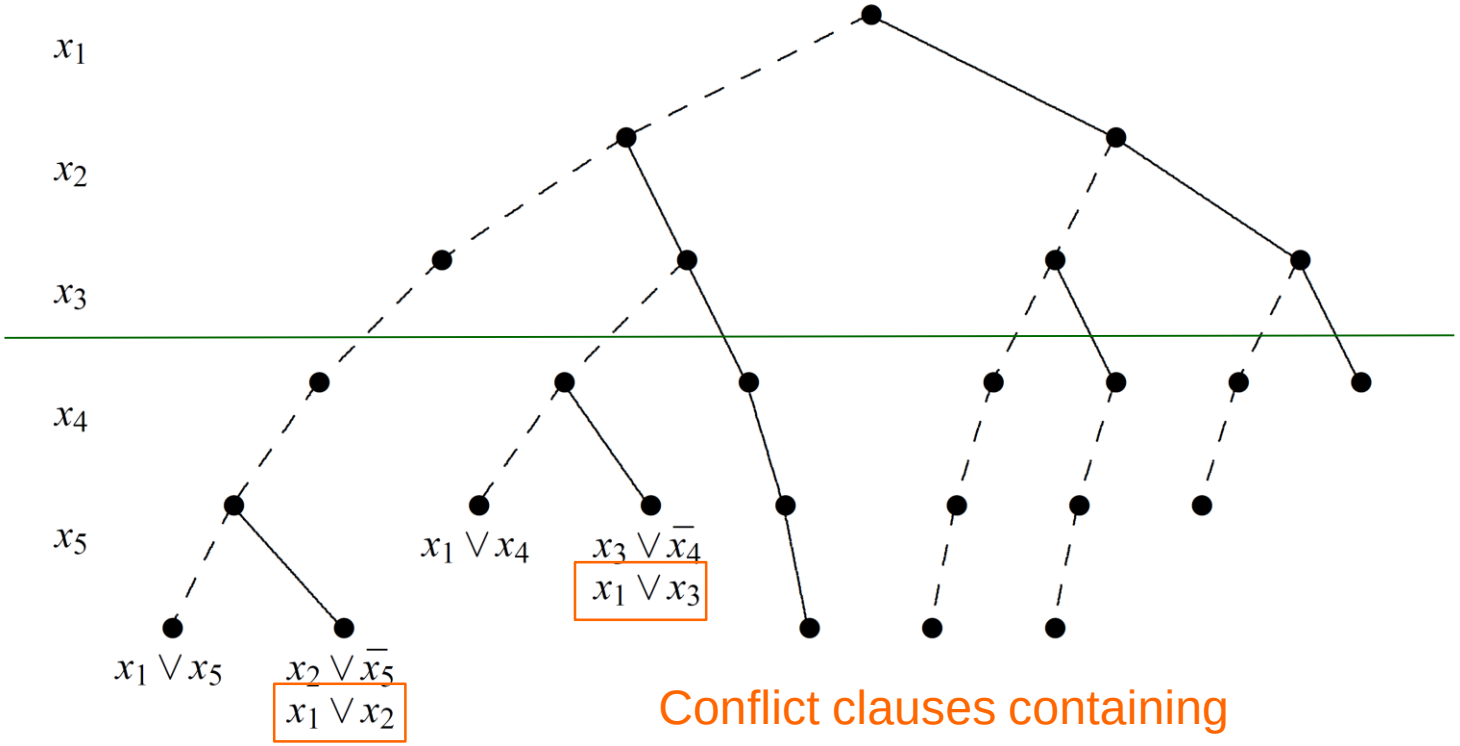
Projection in Satisfiability Solvers

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Projection in Satisfiability Solvers

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Conflict clauses containing x_1, x_2, x_3 describe projection

Consistency as Projection

- **Consistency** is a fundamental concept in **constraint programming** (CP).

Consistency as Projection

- **Consistency** is a fundamental concept in **constraint programming** (CP).

- CP models use high-level **global constraints**
  - Such as **all-different**(x_1, …, x_n) to require that x_1, …, x_n take different values.
  - …rather than writing $x_i \neq x_j$ for all i, j

Consistency as Projection

- **Filtering** algorithms remove values from variable domains that are **inconsistent** with individual global constraints.
  - This ideally achieves **domain consistency**.

Consistency as Projection

- **Filtering** algorithms remove values from variable domains that are **inconsistent** with individual global constraints.
  - This ideally achieves **domain consistency**.
- Reduced domains are **propagated** to the next constraint for further filtering.
  - Smaller domains **simplify search**.

Consistency as Projection

- Achieving domain consistency is a form of **projection**.
  - Project onto each individual variable x_j.

Example:

Constraint set

$$\text{alldiff}\left(x_1, x_2, x_3\right)$$

$$x_1 \in \{a, b\}$$

$$x_2 \in \{a, b\}$$

$$x_3 \in \{b, c\}$$

Consistency as Projection

- Achieving domain consistency is a form of **projection**.
  - Project onto each individual variable x_j.

Example:

Constraint set	Solutions
$\text{alldiff}(x_1, x_2, x_3)$	(x_1, x_2, x_3)
$x_1 \in \{a, b\}$	(a, b, c)
$x_2 \in \{a, b\}$	(b, a, c)
$x_3 \in \{b, c\}$	

Consistency as Projection

- Achieving domain consistency is a form of **projection**.
  - Project onto each individual variable x_j.

Example:

Constraint set

$$\text{alldiff}(x_1, x_2, x_3)$$
$$x_1 \in \{a, b\}$$
$$x_2 \in \{a, b\}$$
$$x_3 \in \{b, c\}$$

Solutions

$$(x_1, x_2, x_3)$$
$$(a, b, c)$$
$$(b, a, c)$$

Projection onto x_1

$$x_1 \in \{a, b\}$$

Projection onto x_2

$$x_2 \in \{a, b\}$$

Projection onto x_3

$$x_3 \in \{c\}$$

Consistency as Projection

- Achieving domain consistency is a form of **projection**.
  - Project onto each individual variable x_j.

Example:

Constraint set

$$\text{alldiff}(x_1, x_2, x_3)$$
$$x_1 \in \{a, b\}$$
$$x_2 \in \{a, b\}$$
$$x_3 \in \{b, c\}$$

Solutions

$$(x_1, x_2, x_3)$$
$$(a, b, c)$$
$$(b, a, c)$$

This achieves domain consistency.

Projection onto x_1

$$x_1 \in \{a, b\}$$

Projection onto x_2

$$x_2 \in \{a, b\}$$

Projection onto x_3

$$x_3 \in \{c\}$$

Consistency as Projection

- Achieving domain consistency is a form of **projection**.
  - Project onto each individual variable x_j.

Example:

Constraint set

$$\text{alldiff}(x_1, x_2, x_3)$$
$$x_1 \in \{a, b\}$$
$$x_2 \in \{a, b\}$$
$$x_3 \in \{b, c\}$$

Solutions

$$(x_1, x_2, x_3)$$
$$(a, b, c)$$
$$(b, a, c)$$

Projection onto x_1

$$x_1 \in \{a, b\}$$

Projection onto x_2

$$x_2 \in \{a, b\}$$

Projection onto x_3

$$x_3 \in \{c\}$$

This achieves domain consistency.

We will regard a projection as a **constraint set**.

Consistency as Projection

- Once we view domain consistency as projection, we see how to extend it to a **stronger** property.
  - Project onto **sets** of variables rather than single variables.
  - Constraint set is ***J*-consistent** if it contains its **projection** onto a set *J* of variables.
    - Domain consistent = { *j* }-consistent for each *j*.
    - Resolution and logic-based Benders achieve *J*-consistency for SAT.

Consistency as Projection

- *J*-consistency and propagation.
  - *J*-consistency can be useful if we propagate through a **richer structure** than domains.
    - …such as **decision diagrams**

Andersen, Hadžić JH, Tiedemann (2007)
Bergman, Ciré, van Hoeve, JH (2014)

Consistency as Projection

- Problem: projection generally results in **greater complexity**.
  - Such as projecting a polyhedron.
  - **Lifting** into a higher space tends to **reduce complexity**.
    - For example, resolution vs. extended resolution.
    - Disaggregation of variables.
  - Let's see what happens for **some common global constraints**.

Consistency as Projection

Constraint	How hard to project?
among	Easy and fast.
sequence	More complicated but fast.
regular	Easy and basically same labor as domain consistency.
alldiff	Surprisingly complicated but practical for small domains.

Projecting Among Constraint

- Requires that a certain number of variables in a set take specified values.
  - Many applications.
  - $\text{among}((x_1,\ldots,x_n),V,t,u)$ requires that at least t and at most u variables among x_1, …, x_n take values in V.

Projecting Among Constraint

Projection of $\text{among}\big((x_1,\ldots,x_n),V,t,u\big)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}\big((x_1,\ldots,x_{n-1}),V,t',u'\big)$$

where

$$(t',u') = \begin{Bmatrix} \big((t-1)^+,u-1\big) & \text{if } D_n \subseteq V \\ \big(t,\min\{u,n-1\}\big) & \text{if } D_n \cap V = \varnothing \\ \big((t-1)^+,\min\{u,n-1\}\big) & \text{otherwise} \end{Bmatrix}$$

Projecting Among Constraint

Projection of $\text{among}\big((x_1,\ldots,x_n),V,t,u\big)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}\big((x_1,\ldots,x_{n-1}),V,t',u'\big)$$

where

$$(t',u') = \begin{Bmatrix} \big((t-1)^+,u-1\big) & \text{if } D_n \subseteq V \\ \big(t,\min\{u,n-1\}\big) & \text{if } D_n \cap V = \varnothing \\ \big((t-1)^+,\min\{u,n-1\}\big) & \text{otherwise} \end{Bmatrix}$$

Example

$$\text{among}\big((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u\big)$$

$D_1 = \{a,b\}$

$D_2 = \{a,b,c\}$

$D_3 = \{a,\mathrm{d}\}$

$D_4 = \{c,d\}$

$D_5 = \{d\}$

Projecting Among Constraint

Projection of $\text{among}\big((x_1,\ldots,x_n),V,t,u\big)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}\big((x_1,\ldots,x_{n-1}),V,t',u'\big)$$

where

$$(t',u') = \begin{Bmatrix} \big((t-1)^+,u-1\big) & \text{if } D_n \subseteq V \\ \big(t,\min\{u,n-1\}\big) & \text{if } D_n \cap V = \varnothing \\ \big((t-1)^+,\min\{u,n-1\}\big) & \text{otherwise} \end{Bmatrix}$$

Example

$$\text{among}\big((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u\big)$$

$$\text{among}\big((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1\big)$$

$D_1 = \{a,b\}$
$D_2 = \{a,b,c\}$
$D_3 = \{a,\mathrm{d}\}$
$D_4 = \{c,d\}$
$D_5 = \{d\}$

Projecting Among Constraint

Projection of $\text{among}\big((x_1,\ldots,x_n),V,t,u\big)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}\big((x_1,\ldots,x_{n-1}),V,t',u'\big)$$

where

$$(t',u') = \begin{Bmatrix} \big((t-1)^+,u-1\big) & \text{if } D_n \subseteq V \\ \big(t,\min\{u,n-1\}\big) & \text{if } D_n \cap V = \varnothing \\ \big((t-1)^+,\min\{u,n-1\}\big) & \text{otherwise} \end{Bmatrix}$$

Example

$$D_1 = \{a,b\}$$
$$D_2 = \{a,b,c\}$$
$$D_3 = \{a,\mathrm{d}\}$$
$$D_4 = \{c,d\}$$
$$D_5 = \{d\}$$

$$\text{among}\big((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u\big)$$
$$\text{among}\big((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1\big)$$
$$\text{among}\big((x_1,x_2,x_3),\{c,d\},(t-2)^+,u-2\big)$$

Projecting Among Constraint

Projection of $\text{among}\big((x_1,\ldots,x_n),V,t,u\big)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}\big((x_1,\ldots,x_{n-1}),V,t',u'\big)$$

where

$$(t',u') = \begin{Bmatrix} \big((t-1)^+,u-1\big) & \text{if } D_n \subseteq V \\ \big(t,\min\{u,n-1\}\big) & \text{if } D_n \cap V = \varnothing \\ \big((t-1)^+,\min\{u,n-1\}\big) & \text{otherwise} \end{Bmatrix}$$

Example

$$D_1 = \{a,b\}$$
$$D_2 = \{a,b,c\}$$
$$D_3 = \{a,\mathrm{d}\}$$
$$D_4 = \{c,d\}$$
$$D_5 = \{d\}$$

$$\text{among}\big((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u\big)$$
$$\text{among}\big((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1\big)$$
$$\text{among}\big((x_1,x_2,x_3),\{c,d\},(t-2)^+,u-2\big)$$
$$\text{among}\big((x_1,x_2),\{c,d\},(t-3)^+,\min\{u-2,2\}\big)$$

Projecting Among Constraint

Projection of $\text{among}((x_1,\ldots,x_n),V,t,u)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}((x_1,\ldots,x_{n-1}),V,t',u')$$

where

$$(t',u') = \begin{cases} ((t-1)^+,u-1) & \text{if } D_n \subseteq V \\ (t,\min\{u,n-1\}) & \text{if } D_n \cap V = \varnothing \\ ((t-1)^+,\min\{u,n-1\}) & \text{otherwise} \end{cases}$$

Example

$$D_1 = \{a,b\}$$
$$D_2 = \{a,b,c\}$$
$$D_3 = \{a,\mathrm{d}\}$$
$$D_4 = \{c,d\}$$
$$D_5 = \{d\}$$

$$\text{among}((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u)$$
$$\text{among}((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1)$$
$$\text{among}((x_1,x_2,x_3),\{c,d\},(t-2)^+,u-2)$$
$$\text{among}((x_1,x_2),\{c,d\},(t-3)^+,\min\{u-2,2\})$$
$$\text{among}((x_1),\{c,d\},(t-4)^+,\min\{u-2,1\})$$

Projecting Among Constraint

Projection of $\text{among}((x_1,\ldots,x_n),V,t,u)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}((x_1,\ldots,x_{n-1}),V,t',u')$$

where

$$(t',u') = \begin{cases} ((t-1)^+,u-1) & \text{if } D_n \subseteq V \\ (t,\min\{u,n-1\}) & \text{if } D_n \cap V = \varnothing \\ ((t-1)^+,\min\{u,n-1\}) & \text{otherwise} \end{cases}$$

Example

$$D_1 = \{a,b\}$$
$$D_2 = \{a,b,c\}$$
$$D_3 = \{a,\mathrm{d}\}$$
$$D_4 = \{c,d\}$$
$$D_5 = \{d\}$$

$$\text{among}((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u)$$
$$\text{among}((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1)$$
$$\text{among}((x_1,x_2,x_3),\{c,d\},(t-2)^+,u-2)$$
$$\text{among}((x_1,x_2),\{c,d\},(t-3)^+,\min\{u-2,2\})$$
$$\text{among}((x_1),\{c,d\},(t-4)^+,\min\{u-2,1\})$$
$$\text{among}((),\{c,d\},(t-4)^+,\min\{u-2,0\})$$

Projecting Among Constraint

Projection of $\text{among}\big((x_1,\ldots,x_n),V,t,u\big)$ onto $x_1,\ldots,x_{n-1}$ is

$$\text{among}\big((x_1,\ldots,x_{n-1}),V,t',u'\big)$$

where

$$(t',u') = \begin{cases} \big((t-1)^+,u-1\big) & \text{if } D_n \subseteq V \\ \big(t,\min\{u,n-1\}\big) & \text{if } D_n \cap V = \varnothing \\ \big((t-1)^+,\min\{u,n-1\}\big) & \text{otherwise} \end{cases}$$

Example

$$D_1 = \{a,b\}$$
$$D_2 = \{a,b,c\}$$
$$D_3 = \{a,\mathrm{d}\}$$
$$D_4 = \{c,d\}$$
$$D_5 = \{d\}$$

$$\text{among}\big((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u\big)$$
$$\text{among}\big((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1\big)$$
$$\text{among}\big((x_1,x_2,x_3),\{c,d\},(t-2)^+,u-2\big)$$
$$\text{among}\big((x_1,x_2),\{c,d\},(t-3)^+,\min\{u-2,2\}\big)$$
$$\text{among}\big((x_1),\{c,d\},(t-4)^+,\min\{u-2,1\}\big)$$
$$\text{among}\big((),\{c,d\},(t-4)^+,\min\{u-2,0\}\big)$$

Feasible if and only if $(t-4)^+ \le \min\{u-2,0\}$

Projecting Sequence Constraint

- Sequence constraint used for assembly line load balancing and the like.
  - For example, at most 3 of every 10 cars on the line require an air conditioner.
  - Equivalent to overlapping among constraints.

Projecting Sequence Constraint

- Projection is based on an integrality property.
  - The coefficient matrix of the inequality formulation has **consecutive ones** property.
  - So projection of the convex hull of the feasible set is an **integral polyhedron**.
    - **Polyhedral projection** therefore suffices.
    - Straightforward (but tedious) application of **Fourier elimination** yields the projection.

Projecting Sequence Constraint

- Projection is based on an integrality property.
  - The coefficient matrix of the inequality formulation has **consecutive ones** property.
  - So projection of the convex hull of the feasible set is an **integral polyhedron**.
    - **Polyhedral projection** therefore suffices.
    - Straightforward (but tedious) application of **Fourier elimination** yields the projection.

- Projection onto any subset of variables is a **generalized sequence constraint**.
  - Complexity of projecting out x_k is $O(kq)$, where q = length of the overlapping sequences.

Following standard convention, we assume without loss of generality that the `sequence` constraint applies to 0-1 variables $x_1, \ldots, x_n$ [23, 46]. It enforces overlapping constraints of the form

$$\texttt{among}((x_{\ell-q+1}, \ldots, x_\ell), \{1\}, L_\ell, U_\ell) \tag{5}$$

Theorem 4. *Given any $k \in \{0, \ldots, n\}$, the projection of the* `sequence` *constraint defined by (5) onto $(x_1, \ldots, x_k)$ is described by a generalized* `sequence` *constraint that enforces constraints of the form*

$$\textit{among}\Big((x_i, \ldots, x_\ell), \{1\}, L^\ell_{\ell-i+1}, U^\ell_{\ell-i+1}\Big) \tag{6}$$

where $i = \ell - q + 1, \ldots, \ell$ for $\ell = q, \ldots, k$ and $i = 1, \ldots, \ell$ for $\ell = 1, \ldots, q-1$. The projection of the `sequence` *constraint onto $(x_1, \ldots, x_{k-1})$ is given by (6) with $L^\ell_{\ell-i+1}$ replaced by $\hat{L}^\ell_{\ell-i+1}$ and $U^\ell_{\ell-i+1}$ by $\hat{U}^\ell_{\ell-i+1}$, where*

$$\begin{aligned}
\hat{L}^\ell_i &= \begin{cases} \max\{L^\ell_i, L^k_{i+k-\ell} - U^k_{k-\ell}\}, \ \textit{for } i = 1, \ldots, q-k+\ell, \\ L^\ell_i, \ \textit{for } i = q-k+\ell+1, \ldots, q \end{cases} \\
\hat{U}^\ell_i &= \begin{cases} \min\{U^\ell_i, U^k_{i+k-\ell} - L^k_{k-\ell}\}, \ \textit{for } i = 1, \ldots, q-k+\ell, \\ U^\ell_i, \ \textit{for } i = q-k+\ell+1, \ldots, q \end{cases}
\end{aligned} \tag{7}$$

Projecting Sequence Constraint

Example $\text{among}\big((x_{t-3},\ldots,x_t),\{1\},2,2\big), \quad t = 4,5,6$

$x_1, x_3, x_4, x_6 \in \{0,1\}, \quad x_2, x_5 \in \{1\}$

To project out x_6, add constraint

$$\text{among}\big((x_3,x_4,x_5),\{1\},1,1\big)$$

Projecting Sequence Constraint

Example $\text{among}\big((x_{t-3},\ldots,x_t),\{1\},2,2\big), \quad t=4,5,6$

$x_1,x_3,x_4,x_6 \in \{0,1\}, \quad x_2,x_5 \in \{1\}$

To project out x_6, add constraint

$$\text{among}\big((x_3,x_4,x_5),\{1\},1,1\big)$$

To project out x_5, add constraints

$$\text{among}\big((x_2,x_3,x_4),\{1\},1,1\big) \quad \text{among}\big((x_3,x_4),\{1\},0,0\big)$$

Projecting Sequence Constraint

Example $\text{among}\big((x_{t-3},\ldots,x_t),\{1\},2,2\big), \quad t = 4,5,6$

$x_1, x_3, x_4, x_6 \in \{0,1\}, \quad x_2, x_5 \in \{1\}$

To project out x_6, add constraint

$$\text{among}\big((x_3,x_4,x_5),\{1\},1,1\big)$$

To project out x_5, add constraints

$$\text{among}\big((x_2,x_3,x_4),\{1\},1,1\big) \quad \text{among}\big((x_3,x_4),\{1\},0,0\big)$$

To project out x_4, add constraints

$$\text{among}\big((x_1),\{1\},1,1\big) \quad \text{among}\big((x_1,x_2,x_3),\{1\},1,2\big)$$
$$\text{among}\big((x_2,x_3),\{1\},0,1\big) \quad \text{among}\big((x_3),\{1\},0,0\big)$$

Projecting Sequence Constraint

Example $\text{among}\big((x_{t-3},\ldots,x_t),\{1\},2,2\big), \quad t = 4,5,6$

$x_1, x_3, x_4, x_6 \in \{0,1\}, \quad x_2, x_5 \in \{1\}$

To project out x_6, add constraint

$$\text{among}\big((x_3,x_4,x_5),\{1\},1,1\big)$$

To project out x_5, add constraints

$$\text{among}\big((x_2,x_3,x_4),\{1\},1,1\big) \quad \text{among}\big((x_3,x_4),\{1\},0,0\big)$$

To project out x_4, add constraints

$$\text{among}\big((x_1),\{1\},1,1\big) \qquad \text{among}\big((x_1,x_2,x_3),\{1\},1,2\big)$$
$$\text{among}\big((x_2,x_3),\{1\},0,1\big) \quad \text{among}\big((x_3),\{1\},0,0\big)$$

To project out x_3, fix $(x_1,x_2) = (1,1)$

Projecting Regular Constraint

- Versatile constraint used for employee shift scheduling, etc.
  - For example, a worker can change shifts only after a day off.
  - Formulates constraint as **deterministic finite automaton**.
  - Or as **regular language** expression.

Projecting Regular Constraint

- Projection can be read from state transition graph.
  - Complexity of projecting onto $x_1, \ldots, x_k$ for all k is $O(nm^2)$, where n = number of variables, m = max number of states per stage.

Projecting Regular Constraint

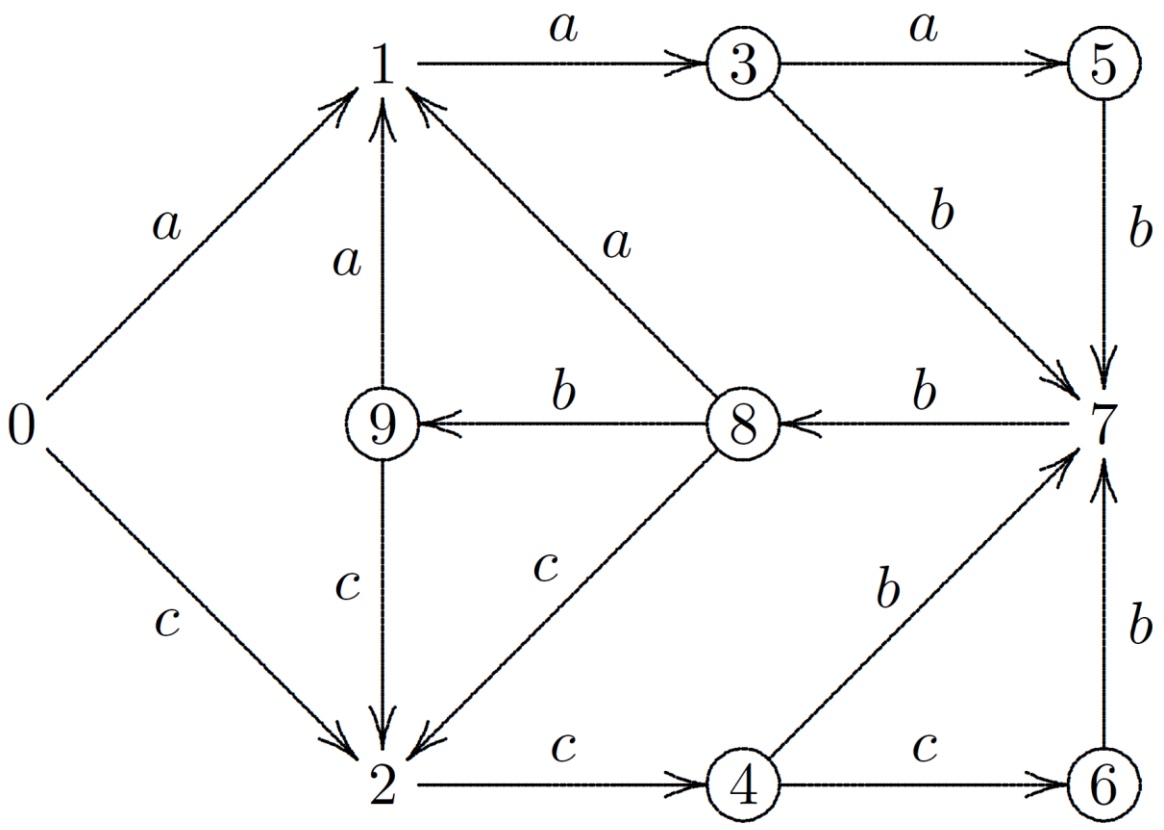
- Projection can be read from state transition graph.
  - Complexity of projecting onto $x_1, \ldots, x_k$ for all k is $O(nm^2)$, where n = number of variables, m = max number of states per stage.

- Shift scheduling example
  - Assign each worker to shift $x_i \in \{a,b,c\}$ on each day $i = 1,\ldots,7$.
  - Must work any given shift 2 or 3 days in a row.
  - No direct transition between shifts a and c.
  - Variable domains: $D_1 = D_5 = \{a,c\}$, $D_2 = \{a,b,c\}$, $D_3 = D_6 = D_7 = \{a,b\}$, $D_4 = \{b,c\}$

Projecting Regular Constraint

Deterministic finite automaton for this problem instance:

◯ = absorbing state

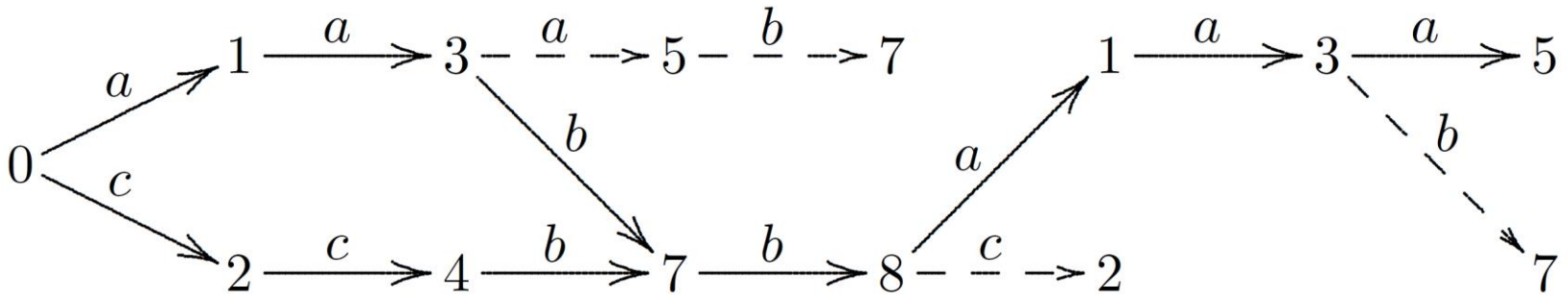


Regular language expression:

(((aa|aaa)(bb|bbb))*|((cc|ccc)(bb|bbb))*)*(c|(aa|aaa)|(cc|ccc))

Projecting Regular Constraint

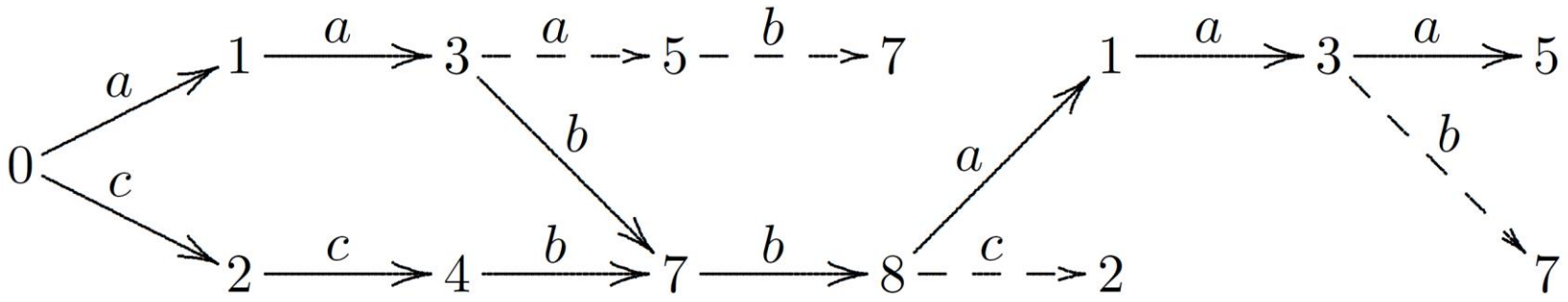
State transition graph for 7 stages
Dashed lines lead to unreachable states.



$j =$	1	2	3	4	5	6	7	8
$D_j =$	$\{a, c\}$	$\{a, b, c\}$	$\{a, b\}$	$\{b, c\}$	$\{a, c\}$	$\{a, b\}$	$\{a, b\}$	

Projecting Regular Constraint

State transition graph for 7 stages
Dashed lines lead to unreachable states.

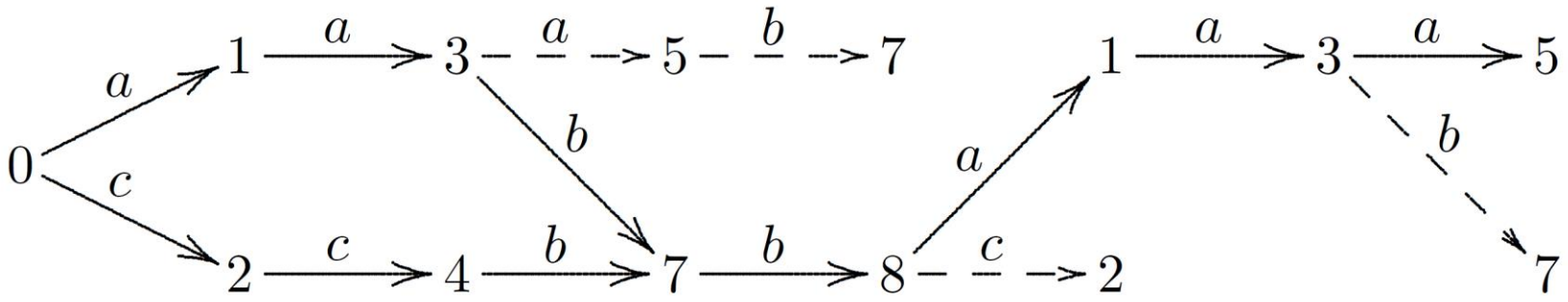


$j =$	1	2	3	4	5	6	7	8
$D_j =$	$\{a,c\}$	$\{a,b,c\}$	$\{a,b\}$	$\{b,c\}$	$\{a,c\}$	$\{a,b\}$	$\{a,b\}$	
$D'_j =$	$\{a,c\}$	$\{a,c\}$	$\{b\}$	$\{b\}$	$\{a\}$	$\{a\}$	$\{a\}$	

Filtered domains

Projecting Regular Constraint

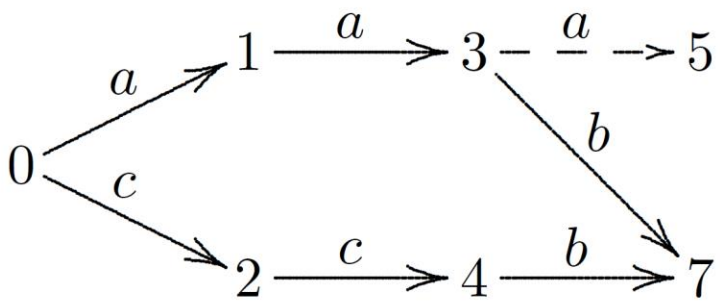
To project onto x_1, x_2, x_3, truncate the graph at stage 4.



$j =$	1	2	3	4	5	6	7	8
$D_j =$	$\{a,c\}$	$\{a,b,c\}$	$\{a,b\}$	$\{b,c\}$	$\{a,c\}$	$\{a,b\}$	$\{a,b\}$	
$D'_j =$	$\{a,c\}$	$\{a,c\}$	$\{b\}$	$\{b\}$	$\{a\}$	$\{a\}$	$\{a\}$	

Projecting Regular Constraint

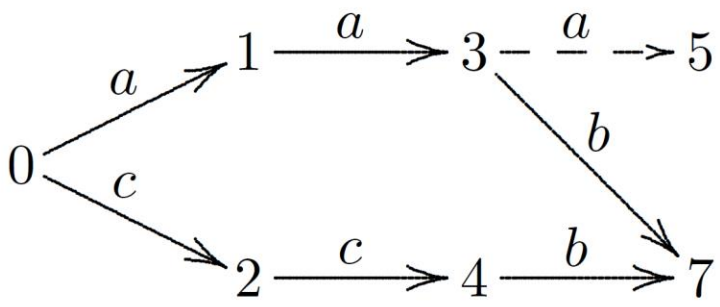
To project onto x_1, x_2, x_3, truncate the graph at stage 4.



$j =$	1	2	3	4
$D_j =$	$\{a, c\}$	$\{a, b, c\}$	$\{a, b\}$	
$D'_j =$	$\{a, c\}$	$\{a, c\}$	$\{b\}$	

Projecting Regular Constraint

To project onto x_1, x_2, x_3, truncate the graph at stage 4.



$$
\begin{array}{lcccc}
j = & 1 & 2 & 3 & 4 \\
D_j = & \{a,c\} & \{a,b,c\} & \{a,b\} & \\
D'_j = & \{a,c\} & \{a,c\} & \{b\} &
\end{array}
$$

Resulting graph can be viewed as a constraint that describes the projection.

Constraint is easily propagated through a relaxed decision diagram.

Projecting Alldiff Constraint

- Used for sequencing and much else.
  - Domain consistency **easily achieved** by matching algorithm and duality theory.

Projecting Alldiff Constraint

- Projection is inherently **complicated**.
  - But it can simplify for small domains.

- The result is a **disjunction** of constraint sets,
  - …each of which contains an alldiff constraint and some atmost constraints.

Projecting Alldiff Constraint

Example $\text{alldiff}\left(x_1, x_2, x_3, x_4, x_5\right)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting Alldiff Constraint

Example $\text{alldiff}(x_1, x_2, x_3, x_4, x_5)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_5, we get

$$\text{alldiff}(x_1, x_2, x_3, x_4), \quad \text{atmost}((x_1, x_2, x_3, x_4), \{a,f,g\}, 2)$$

because x_5 must take one of the values in {*a*,*f*,*g*}, …

Projecting Alldiff Constraint

Example $\text{alldiff}(x_1, x_2, x_3, x_4, x_5)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_5, we get

$$\text{alldiff}(x_1, x_2, x_3, x_4), \quad \text{atmost}((x_1, x_2, x_3, x_4), \{a,f,g\}, 2)$$

because x_5 must take one of the values in {*a*,*f*,*g*}, leaving 2 for other x_i s.

Projecting Alldiff Constraint

Example $\text{alldiff}(x_1, x_2, x_3, x_4, x_5)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_5, we get

$$\text{alldiff}(x_1, x_2, x_3, x_4), \quad \text{atmost}((x_1, x_2, x_3, x_4), \{a,f,g\}, 2)$$

because x_5 must take one of the values in {*a*,*f*,*g*}, leaving 2 for other x_i s.

Projecting out x_4, we note that $x_4 \in \{a,f,g\}$ or $x_4 \notin \{a,f,g\}$.

Projecting Alldiff Constraint

Example $\text{alldiff}(x_1, x_2, x_3, x_4, x_5)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_5, we get

$$\text{alldiff}(x_1, x_2, x_3, x_4), \quad \text{atmost}((x_1, x_2, x_3, x_4), \{a,f,g\}, 2)$$

because x_5 must take one of the values in {*a,f,g*}, leaving 2 for other x_i s.

Projecting out x_4, we note that $x_4 \in \{a,f,g\}$ or $x_4 \notin \{a,f,g\}$.

If $x_4 \in \{a,f,g\}$, we get

$$\text{alldiff}(x_1, x_2, x_3), \quad \text{atmost}((x_1, x_2, x_3), \{a,f,g\}, 1)$$

Projecting Alldiff Constraint

Example $\text{alldiff}(x_1, x_2, x_3, x_4, x_5)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_5, we get

$$\text{alldiff}(x_1, x_2, x_3, x_4), \quad \text{atmost}((x_1, x_2, x_3, x_4), \{a,f,g\}, 2)$$

because x_5 must take one of the values in {*a,f,g*}, leaving 2 for other x_i s.

Projecting out x_4, we note that $x_4 \in \{a,f,g\}$ or $x_4 \notin \{a,f,g\}$.

If $x_4 \in \{a,f,g\}$, we get

$$\text{alldiff}(x_1, x_2, x_3), \quad \text{atmost}((x_1, x_2, x_3), \{a,f,g\}, 1)$$

If $x_4 \notin \{a,f,g\}$, we get $x_4 = e$, and we remove *e* from other domains.

Projecting Alldiff Constraint

Example $\text{alldiff}\left(x_1, x_2, x_3, x_4, x_5\right)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_5, we get

$$\text{alldiff}\left(x_1, x_2, x_3, x_4\right), \quad \text{atmost}\left((x_1, x_2, x_3, x_4), \{a,f,g\}, 2\right)$$

because x_5 must take one of the values in $\{a,f,g\}$, leaving 2 for other x_i s.

Projecting out x_4, we note that $x_4 \in \{a,f,g\}$ or $x_4 \notin \{a,f,g\}$.

If $x_4 \in \{a,f,g\}$, we get

$$\text{alldiff}\left(x_1, x_2, x_3\right), \quad \text{atmost}\left((x_1, x_2, x_3), \{a,f,g\}, 1\right)$$

If $x_4 \notin \{a,f,g\}$, we get $x_4 = e$, and we remove e from other domains.

So the projection is

$$\begin{bmatrix} \text{alldiff}\left(x_1, x_2, x_3\right) \\ \text{atmost}\left((x_1, x_2, x_3), \{a,f,g\}, 1\right) \end{bmatrix} \vee \begin{bmatrix} D_1 = \{a,b,c\} \\ D_2 = \{c,d\} \\ D_3 = \{d,f\} \end{bmatrix}$$

Projecting Alldiff Constraint

Example $\text{alldiff}\left(x_1, x_2, x_3, x_4, x_5\right)$

$$D_1 = \{a,b,c\}, D_2 = \{c,d,e\}, D_3 = \{d,e,f\}, D_4 = \{e,f,g\}, D_5 = \{a,f,g\}$$

Projecting out x_3, we get simply

$$\text{alldiff}\left(x_1, x_2\right)$$

Projecting out x_2, we get the original domain for x_1

$$D_1 = \{a,b,c\}$$

Algorithm 2 Given a projection of $\texttt{alldiff}(x^n)$ onto x^k, this algorithm computes a projection onto x^{k-1}. The projection onto x^k is assumed to be a disjunction of constraint sets, each of which has the form (10). The above algorithm is applied to each disjunct, after which the disjunction of all created constraint sets forms the projection onto x^{k-1}.

For all $i \in I$: if $\texttt{atmost}(x^k, V_i, b_i)$ is redundant then remove i from I.
For all $i \in I$:
    If $D_k \cap V_i \neq \emptyset$ then
        If $b_i > 1$ then
            Create a constraint set consisting of $\texttt{alldiff}(x^{k-1})$,
            $\texttt{atmost}(x^{k-1}, V_{i'}, b_{i'})$ for $i' \in I \setminus \{i\}$, and $\texttt{atmost}\ (x^{k-1}, V_i, b_i - 1)$.
Let $R = D_k \setminus \bigcup_{i \in I} V_i$.
If $|R| > 1$ then
    Create a constraint set consisting of $\texttt{alldiff}(x^{k-1})$,
    $\texttt{atmost}(x^{k-1}, V_{i'}, b_{i'})$ for $i' \in I$, and $\texttt{atmost}\ (x^{k-1}, R, |R| - 1)$.
Else if $|R| = 1$ then
    Let $R = \{v\}$ and remove v from D_j for $j = 1, \ldots, k-1$ and from V_i for $i \in I$.
    If D_j is nonempty for $j = 1, \ldots, k-1$ then
        For all $i' \in I$: if $\texttt{atmost}(x^{k-1}, V_{i'}, b_{i'})$ is redundant then remove i' from i.
        Create a constraint set consisting of $\texttt{alldiff}(x^{k-1})$ and
        $\texttt{atmost}(x^{k-1}, V_{i'}, b_{i'})$ for $i' \in I$.

That's it.