A Folding Algorithm for Eliminating Existential Variables from Constraint Logic Programs

V. Senni[1] A. Pettorossi[1] M. Proietti[2]

[1]DISP - University of Roma “Tor Vergata”

[2]IASI - CNR

ICLP 2008

Program Transformation for Logic Programs

Transforming **programs** while preserving **semantics**

$$\begin{array}{ccccc} P_1 & \Longrightarrow & \ldots & \Longrightarrow & P_n \\ M(P_1) & = & \ldots & = & M(P_n) \end{array}$$

UNFOLDING

$$\begin{array}{l} \gamma\colon\ H \leftarrow K \wedge R \\ \delta\colon\ K \leftarrow B \end{array} \quad \Longrightarrow \quad \eta\colon\ H \leftarrow B \wedge R$$

FOLDING

$$\begin{array}{l} \gamma\colon\ H \leftarrow B \wedge R \\ \delta\colon\ K \leftarrow B \end{array} \quad \Longrightarrow \quad \eta\colon\ H \leftarrow K \wedge R$$

Existential Variables

An Existential Variable occurs in the *body* of a clause and *not in its head*

(recall, $\forall X\ (p \leftarrow q(X)) \ \equiv \ p \leftarrow \exists X\ q(X)$)

By folding we can eliminate existential variables and, thus,

- *reduce nondeterminism*
- *avoid intermediate data structures*

Optimization by Existential Variables Elimination (1)

Example

String matching: $S = L :: (P :: R)$

1. $match(P, S) \leftarrow a(L, PR, S) \wedge a(P, R, PR)$
2. $a([\,], X, X) \leftarrow$
3. $a([X|Xs], Y, [X|Zs]) \leftarrow a(Xs, Y, Zs)$

existential variables

Optimization by Existential Variables Elimination (1)

Example

String matching: $S = L :: (P :: R)$

1. $match(P, S) \leftarrow a(L, PR, S) \wedge a(P, R, PR)$
2. $a([\,], X, X) \leftarrow$
3. $a([X|Xs], Y, [X|Zs]) \leftarrow a(Xs, Y, Zs)$

existential variables

$\Downarrow$ **UNFOLD 1**

4. $match(P, S) \leftarrow a(P, R, S)$
5. $match(P, [X|S]) \leftarrow a(L, PR, S) \wedge a(P, R, PR)$

Optimization by Existential Variables Elimination (1)

Example

String matching: $S = L :: (P :: R)$

1. $match(P, S) \leftarrow a(L, PR, S) \wedge a(P, R, PR)$
2. $a([\,], X, X) \leftarrow$
3. $a([X|Xs], Y, [X|Zs]) \leftarrow a(Xs, Y, Zs)$

existential variables

$\Downarrow$ **UNFOLD 1**

4. $match(P, S) \leftarrow a(P, R, S)$
5. $match(P, [X|S]) \leftarrow a(L, PR, S) \wedge a(P, R, PR)$

$\Downarrow$ **FOLD 5**

4. $match(P, S) \leftarrow a(P, R, S)$
6. $match(P, [X|S]) \leftarrow match(P, S)$

Optimization by Existential Variables Elimination (2)

4. $match(P, S) \leftarrow a(P, R, S)$
6. $match(P, [X|S]) \leftarrow match(P, S)$

$\Downarrow$ **DEFINE; UNFOLD; FOLD**

7. $match(P, S) \leftarrow prefix(P, S)$
6. $match(P, [X|S]) \leftarrow match(P, S)$
8. $prefix([\,], S) \leftarrow$
9. $prefix([X|P], [X|S]) \leftarrow prefix(P, S)$

no existential variables

By eliminating the existential variables

- we *avoided the construction of* the intermediate list *PR*
- we *reduced nondeterminism*

Not every possible application of folding ensure the elimination of existential variables.

Folding for Constraint Logic Programs

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

EXISTENTIAL VARIABLE ELIMINATION

$Vars(K\vartheta) \subseteq Vars(H)$

Folding for Constraint Logic Programs

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

$\Downarrow \equiv$

γ': $H \leftarrow e \wedge (d \wedge B)\vartheta \wedge R$

EXISTENTIAL VARIABLE ELIMINATION

$Vars(K\vartheta) \subseteq Vars(H)$

$\gamma \overset{\equiv}{\Longrightarrow} \gamma'$ If

(1) $\mathrm{AC}_\wedge \models G \leftrightarrow B\vartheta \wedge R$

(equivalence modulo Associativity and Commutativity of $\wedge$)

(2) $\mathcal{D} \models \forall\, (c \leftrightarrow e \wedge d\vartheta)$

Folding for Constraint Logic Programs

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

$\Downarrow \equiv$

γ': $H \leftarrow e \wedge (d \wedge B)\vartheta \wedge R$

$\Downarrow$ **FOLDING**

η: $H \leftarrow e \wedge K\vartheta \wedge R$

EXISTENTIAL VARIABLE ELIMINATION

$Vars(K\vartheta) \subseteq Vars(H)$

$\gamma \overset{\equiv}{\Longrightarrow} \gamma'$ If

(1) $\mathrm{AC}_\wedge \models G \leftrightarrow B\vartheta \wedge R$

(equivalence modulo Associativity and Commutativity of $\wedge$)

(2) $\mathcal{D} \models \forall\, (c \leftrightarrow e \wedge d\vartheta)$

An Example of CLP Folding

$$\gamma: \quad p(X) \leftarrow 1-X>0 \wedge X+Z-1\geq 0 \wedge q(Z)$$
$$\delta: \quad s(Y) \leftarrow W>0 \wedge Y+2W-3\geq 0 \wedge q(W)$$

EXISTENTIAL VARIABLE ELIMINATION
$Vars(s(Y)\vartheta) \subseteq \{X\}$

$$\Downarrow \equiv$$

$$\gamma': \quad p(Y) \leftarrow 1-X>0 \wedge (W>0 \wedge Y+2W-3\geq 0 \wedge q(W))\{Y/2X+1, W/Z\}$$

$$\Downarrow \textbf{FOLDING}$$

$$\eta: \quad p(Y) \leftarrow 1-X>0 \wedge s(Y)\{Y/2X+1, W/Z\}$$

An Example of CLP Folding

$$\gamma: \quad p(X) \leftarrow 1-X>0 \wedge X+Z-1\geq 0 \wedge q(Z)$$
$$\delta: \quad s(Y) \leftarrow W>0 \wedge Y+2W-3\geq 0 \wedge q(W)$$

EXISTENTIAL VARIABLE ELIMINATION
$Vars(s(Y)\vartheta) \subseteq \{X\}$

$$\Downarrow \equiv$$

$$\gamma': \quad p(Y) \leftarrow 1-X>0 \wedge (W>0 \wedge Y+2W-3\geq 0 \wedge q(W))\{Y/2X+1, W/Z\}$$

$$\Downarrow \text{ FOLDING}$$

$$\eta: \quad p(Y) \leftarrow 1-X>0 \wedge s(Y)\{Y/2X+1, W/Z\}$$

We can show that, in this case, $\gamma \overset{\equiv}{\Longrightarrow} \gamma'$ since conditions (1) and (2) of the definition of $\equiv$ are satisfied

But, in general,

$$H \leftarrow c \wedge G \overset{\equiv}{\Longrightarrow} H \leftarrow ? \wedge (d \wedge B)? \wedge ?$$

how do we fill the question marks ?

Our Contribution

A Folding Algorithm which, given

$$\begin{array}{lrcl} \gamma: & H & \leftarrow & c \wedge G \\ \delta: & K & \leftarrow & d \wedge B \end{array}$$

computes e, ϑ, R such that

$$H \leftarrow c \wedge G \stackrel{\equiv}{\Longrightarrow} H \leftarrow e \wedge (d \wedge B)\vartheta \wedge R$$

and

EXISTENTIAL VARIABLE ELIMINATION

$Vars(K\vartheta) \subseteq Vars(H)$

A Two-Step Folding Algorithm

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

A Two-Step Folding Algorithm

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

$\Downarrow \equiv$ **Step 1: GOAL MATCHING** $\mathrm{AC}_{\wedge} \models G \leftrightarrow B\alpha \wedge R$

$H \leftarrow c \wedge B\alpha \wedge R$

A Two-Step Folding Algorithm

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

$\Downarrow \equiv$ **Step 1: GOAL MATCHING** $\mathsf{AC}_{\wedge} \models G \leftrightarrow B\alpha \wedge R$

$H \leftarrow c \wedge B\alpha \wedge R$

$\Downarrow \equiv$ **Step 2: CONSTRAINT MATCHING** $\mathcal{D} \models \forall\,(c \leftrightarrow e \wedge d\alpha\beta)$

$H \leftarrow e \wedge d\alpha\beta \wedge B\alpha\beta \wedge R$

A Two-Step Folding Algorithm

γ: $H \leftarrow c \wedge G$

δ: $K \leftarrow d \wedge B$

$\Downarrow \equiv$ **Step 1: GOAL MATCHING** $\mathsf{AC}_{\wedge} \models G \leftrightarrow B\alpha \wedge R$

$H \leftarrow c \wedge B\alpha \wedge R$

$\Downarrow \equiv$ **Step 2: CONSTRAINT MATCHING** $\mathcal{D} \models \forall\,(c \leftrightarrow e \wedge d\alpha\beta)$

$H \leftarrow e \wedge d\alpha\beta \wedge B\alpha\beta \wedge R$

$\Downarrow$ **FOLDING**

η: $H \leftarrow e \wedge K\vartheta \wedge R$ $\vartheta = \alpha\beta$

Step 1 - Goal Matching

Problem: find α and R such that

$$\mathrm{AC}_\wedge \models G \leftrightarrow B\alpha \wedge R$$

(+ additional conditions on the substitution α)

Algorithm: start from B/G and apply the following rewritings

$$
\begin{array}{lcl}
L_1 \wedge B' \,/\, G_1 \wedge L_2 \wedge G_2 & \Longrightarrow & L_1 \,/\, L_2,\ B' \,/\, G_1 \wedge G_2 \\
a(s_1, \ldots, s_n) \,/\, a(t_1, \ldots, t_n) & \Longrightarrow & s_1 \,/\, t_1, \ldots, s_n \,/\, t_n \\
a(s_1, \ldots, s_n) \,/\, X & \Longrightarrow & \textbf{fail} \\
X \,/\, Y, \mathit{true}/R & \Longrightarrow & \textbf{fail} \quad \text{if } Y \text{ is a variable in } R \\
& \vdots &
\end{array}
$$

$$\boxed{\{B/G\} \Longrightarrow^* \alpha \cup \{\mathit{true}/R\}}$$

Step 2 - Constraint Matching

Problem: find a constraint e and a substitution β such that

$$\mathcal{D} \models \forall\, (c \leftrightarrow e \wedge d\beta)$$

(1) e can be computed from c by *projecting out* the variables not occuring in H:

$$e = \mathit{project}(c, \mathit{Vars}(H))$$

(2) computing β is more problematic, even if we stick to *linear* constraints on the domain of *real numbers* $\mathbb{R}$ or *rational numbers* $\mathbb{Q}$

We Cannot Use Tarski's Quantifier Elimination

Find f such that $\mathcal{D} \models \forall X\ (c(X) \leftrightarrow e(X) \wedge d(Y)\underbrace{\{Y/f(X)\}}_{\beta})$

which is equivalent to

find m and q such that $\mathcal{D} \models \forall X\ (c(X) \leftrightarrow e(X) \wedge d(mX + q))$

We Cannot Use Tarski's Quantifier Elimination

Find f such that $\mathcal{D} \models \forall X\ (c(X) \leftrightarrow e(X) \wedge d(Y)\underbrace{\{Y/f(X)\}}_{\beta})$

which is equivalent to

find m and q such that $\mathcal{D} \models \forall X\ (c(X) \leftrightarrow e(X) \wedge d(mX+q))$

Case of real numbers $\mathbb{R}$

$$\forall X\ (c(X) \leftrightarrow e(X) \wedge d(mX+q))$$

$$\downarrow \text{ by Cylindric Algebraic Decomposition}$$

$$\varphi(m,q)$$

φ is a ***nonlinear*** quantifier-free formula and, in general, it is ***impossible*** to compute explicitly m and q

Case of rational numbers $\mathbb{Q}$

$$\forall X\ (c(X) \leftrightarrow e(X) \wedge d(mX+q))$$

$$\downarrow$$

$$\ldots$$

The existence of a general procedure for the elimination of quantifiers is an open problem

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow e \wedge \underbrace{(Z>0 \wedge Y+2Z-3\geq 0)\{Y/2X+1\}}_{d\ \beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓
2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓
2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow e \wedge \underbrace{(Z>0 \wedge Y+2Z-3\geq 0)}_{d}\{Y/2X+1\}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓
2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

   $Y = X-\mathbf{Z}+2$ ✗

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓
2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$ $\qquad$ $2X+2Z-2\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

   $Y = X-\mathbf{Z}+2$ ✗

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓
2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

   $Y = X-\mathbf{Z}+2$ ✗

   $2X+2Z-2\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $2X+2Z-2 = Y+2Z-3$

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow e \wedge \underbrace{(Z>0 \wedge Y+2Z-3\geq 0)}_{d}\{Y/2X+1\}_{\beta}$
2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓
2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

   $Y = X-\mathbf{Z}+2$ ✗

   $2X+2Z-2\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $2X+2Z-2 = Y+2Z-3$

   $Y = 2X+1$ ✓

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$

2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓

2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

   $Y = X-\mathbf{Z}+2$ ✗

   $2X+2Z-2\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $2X+2Z-2 = Y+2Z-3$

   $Y = 2X+1$ ✓

3. $1-X>0 \wedge X+Z-1\geq 0 \rightarrow Z>0$

   $Z\geq X-1 \wedge X-1>0$ ✓

An Example of Constraint Matching

Given c and d,

compute e and β such that

1. $\underbrace{1-X>0 \wedge X+Z-1\geq 0}_{c} \leftrightarrow \underbrace{1-X>0}_{e} \wedge (\underbrace{Z>0 \wedge Y+2Z-3\geq 0}_{d})\underbrace{\{Y/2X+1\}}_{\beta}$

2. $Vars(Y\beta) \subseteq \{X\}$ (EXISTENTIAL VARIABLE ELIMINATION)

$$e = project(1-X>0 \wedge X+Z-1\geq 0, \{X\}) = 1-X>0$$

1. $1-X>0 \leftrightarrow 1-X>0$ ✓

2. $X+Z-1\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $X+Z-1 = Y+2Z-3$

   $Y = X-\mathbf{Z}+2$ ✗

   $2X+2Z-2\geq 0 \leftrightarrow Y+2Z-3\geq 0$

   $2X+2Z-2 = Y+2Z-3$

   $Y = 2X+1$ ✓

3. $1-X>0 \wedge X+Z-1\geq 0 \rightarrow Z>0$

   $Z\geq X-1 \wedge X-1>0$ ✓

$$\beta = \{Y/2X+1\}$$

A Three-Step Algorithm for Finding β (1)

Step 1. Find an injection from **orange constraints** to **green constraints**

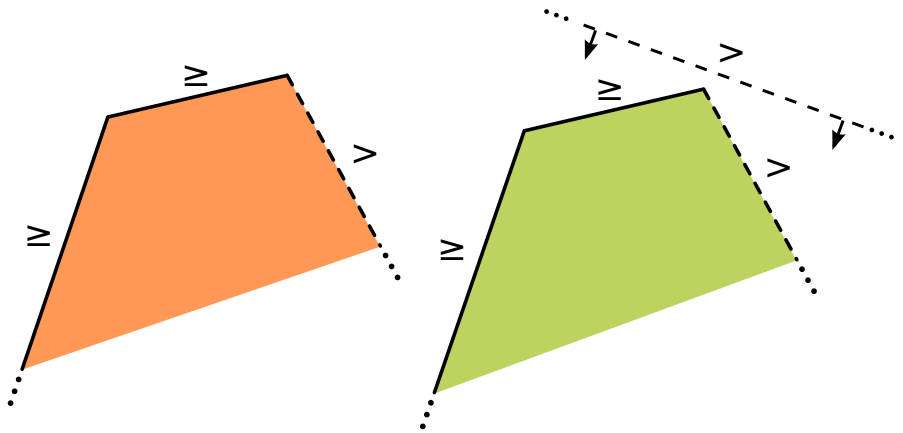
$$\underbrace{c_1 \wedge \ldots \wedge c_k}_{\overrightarrow{X}} \leftrightarrow \underbrace{e_1 \wedge \ldots \wedge e_m}_{\overrightarrow{X}} \wedge \underbrace{d_1 \wedge \ldots \wedge d_n}_{\overrightarrow{Y}}$$

Variables:

Step 1. Find an injection from **orange constraints** to **green constraints**

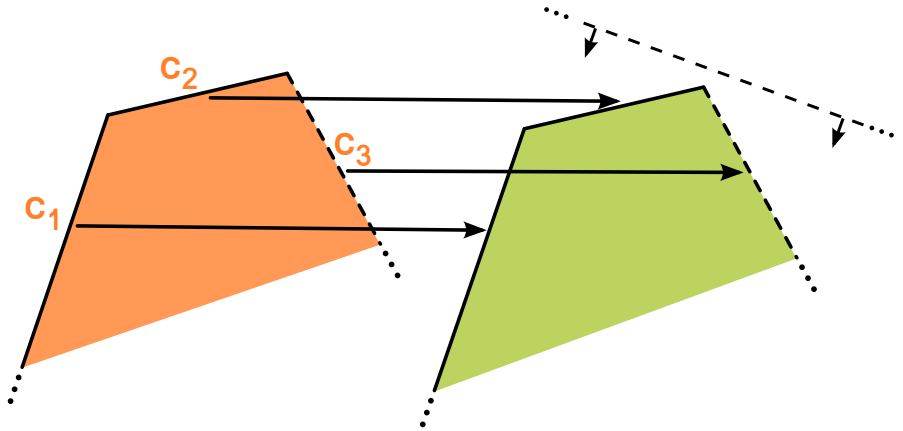
Variables:

$$\underbrace{c_1 \wedge \ldots \wedge c_k}_{\overrightarrow{X}} \leftrightarrow \underbrace{e_1 \wedge \ldots \wedge e_m}_{\overrightarrow{X}} \wedge \underbrace{d_1 \wedge \ldots \wedge d_n}_{\overrightarrow{Y}}$$



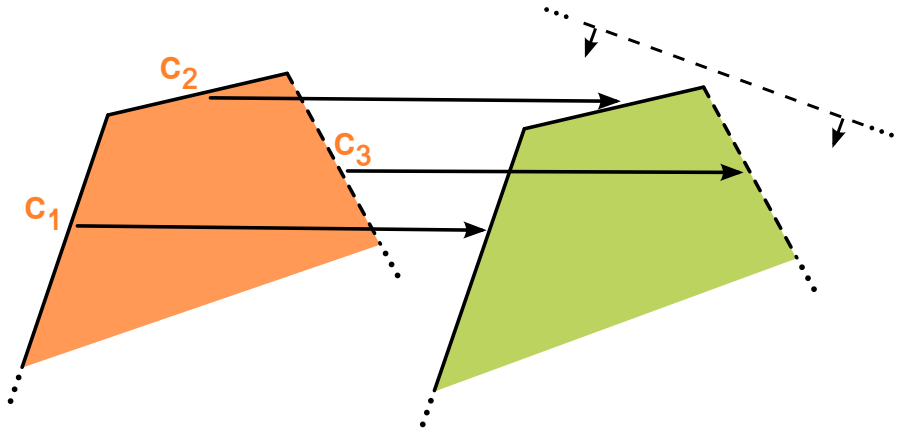
Step 1. Find an injection from **orange constraints** to **green constraints**

Variables: $\underbrace{c_1 \wedge \ldots \wedge c_k}_{\vec{X}} \leftrightarrow \underbrace{e_1 \wedge \ldots \wedge e_m}_{\vec{X}} \wedge \underbrace{d_1 \wedge \ldots \wedge d_n}_{\vec{Y}}$



Step 1. Find an injection from **orange constraints** to **green constraints**

Variables: $\underbrace{c_1 \wedge \ldots \wedge c_k}_{\vec{X}} \leftrightarrow \underbrace{e_1 \wedge \ldots \wedge e_m}_{\vec{X}} \wedge \underbrace{d_1 \wedge \ldots \wedge d_n}_{\vec{Y}}$



We obtain a set of formulas of the form

- $(p \geq 0 \leftrightarrow q \geq 0)$ (INJECTION)
- $(p_1 \geq 0 \wedge \ldots \wedge p_k \geq 0 \rightarrow q \geq 0)$ (EXTRA CONSTRAINTS)

A Three-Step Algorithm for Finding β (2)

Step 2. Replace *formulas* by sets of *equations* and *inequations* on variables in $\overrightarrow{X}$ and $\overrightarrow{Y}$, and some new variables $\mathbf{V}, \mathbf{V_1}, \ldots, \mathbf{V_{k+1}}, \ldots$

$\leftrightarrow$ **ELIM.** $(p \geq 0 \leftrightarrow q \geq 0) \implies \{\mathbf{V}p - q = 0, \mathbf{V} > 0\}$

$\rightarrow$ **ELIM.** $(p_1 \geq 0 \wedge \ldots \wedge p_k \geq 0 \rightarrow q \geq 0) \implies$ (by Farkas' lemma)
$\{\mathbf{V_1}p_1 + \ldots + \mathbf{V_k}p_k + \mathbf{V_{k+1}} - q = 0, \mathbf{V_1} \geq 0, \ldots, \mathbf{V_{k+1}} \geq 0\}$

solve equations w.r.t. $\overrightarrow{Y}$

$=$ **SOLVE** $aY_i + f(Y_{i+1}, \ldots, Y_m, X, \mathbf{V}) = 0 \implies Y_i = \frac{1}{a}(f(Y_{i+1}, \ldots, Y_m, X, \mathbf{V}))$

decompose bilinear polynomials into *linear* polynomials

$=$ **ELIM.** $f_1(\mathbf{V})X_1 + \ldots + f_n(\mathbf{V})X_n = 0 \implies \{f_1(\mathbf{V}) = 0, \ldots, f_n(\mathbf{V}) = 0\}$

A Three-Step Algorithm for Finding β (2)

Step 2. Replace *formulas* by sets of *equations* and *inequations* on variables in $\overrightarrow{X}$ and $\overrightarrow{Y}$, and some new variables $\mathbf{V}, \mathbf{V_1}, \ldots, \mathbf{V_{k+1}}, \ldots$

$\leftrightarrow$ **ELIM.** $(p \geq 0 \leftrightarrow q \geq 0) \implies \{\mathbf{V}p - q = 0, \mathbf{V} > 0\}$

$\rightarrow$ **ELIM.** $(p_1 \geq 0 \wedge \ldots \wedge p_k \geq 0 \rightarrow q \geq 0) \implies$ (by Farkas' lemma)

$$\{\mathbf{V_1}p_1 + \ldots + \mathbf{V_k}p_k + \mathbf{V_{k+1}} - q = 0, \mathbf{V_1} \geq 0, \ldots, \mathbf{V_{k+1}} \geq 0\}$$

solve equations w.r.t. $\overrightarrow{Y}$

$=$ **SOLVE** $aY_i + f(Y_{i+1}, \ldots, Y_m, X, \mathbf{V}) = 0 \implies Y_i = \frac{1}{a}(f(Y_{i+1}, \ldots, Y_m, X, \mathbf{V}))$

decompose bilinear polynomials into *linear* polynomials

$=$ **ELIM.** $f_1(\mathbf{V})X_1 + \ldots + f_n(\mathbf{V})X_n = 0 \implies \{f_1(\mathbf{V}) = 0, \ldots, f_n(\mathbf{V}) = 0\}$

Step 3. Solve the set of *linear* constraints on $\mathbf{V}$'s

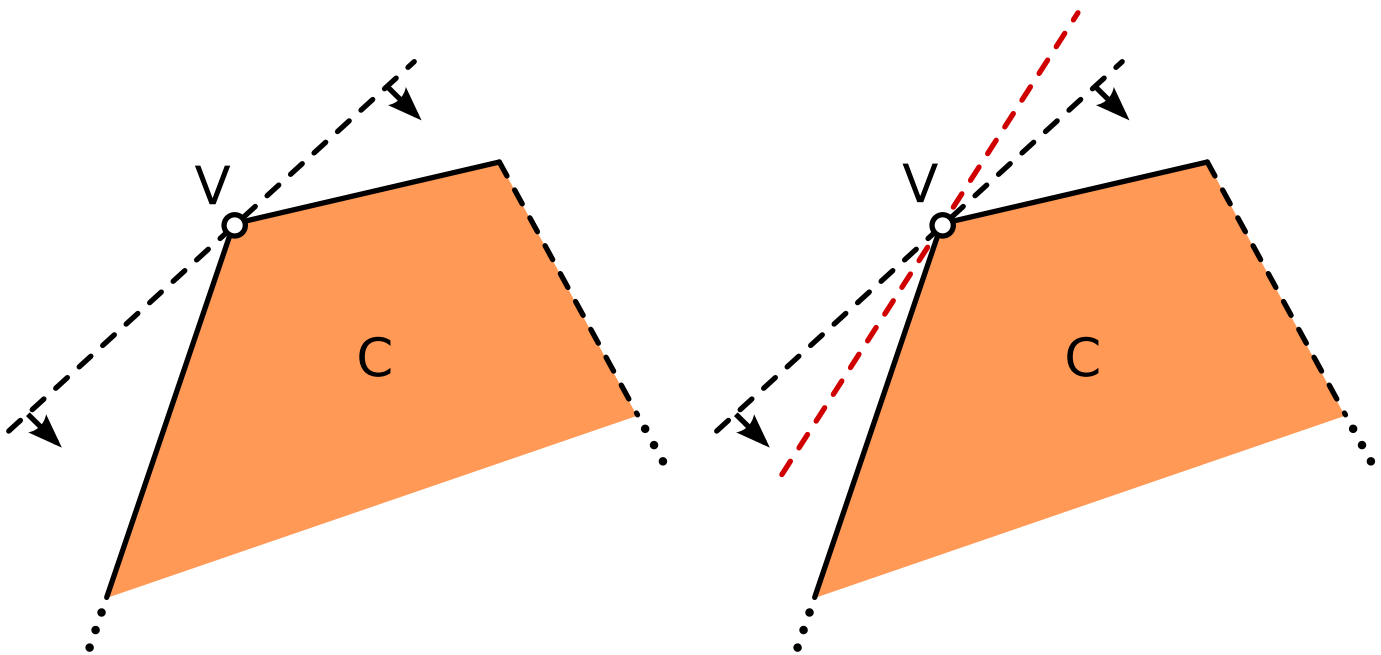
We obtain β of the form $\{Y_1/g_1(X), \ldots, Y_m/g_m(X)\}$

Admissibility

The algorithm is *complete* for *admissible* constraints

Admissible constraints have a *unique* constraint-representation

A non-admissible constraint:



Admissibility is *decidable*

Theorem Proving by Existential Variable Elimination

$$\textbf{prop}: \ \neg \exists L \neg \exists U \neg \exists X \ (X \in L \ \wedge \ X > U)$$

"every list of rational numbers has an upper bound"

(1) Encode **prop** as a *stratified* constraint logic program

1. $\textbf{prop} \leftarrow \neg \, p$
2. $p \leftarrow \mathit{list}(L) \wedge \neg \, q(L)$
3. $q(L) \leftarrow \mathit{list}(L) \wedge \neg \, r(L, U)$
4. $r(L, U) \leftarrow X > U \wedge \mathit{list}(L) \wedge \mathit{member}(X, L)$

(2) Eliminate *existential variables* bottom-up by using the *unfolding* and *folding* rules and obtain a propositional program

1. $\textbf{prop} \leftarrow \neg \, p$
5. $p \leftarrow p1$
6. $p1 \leftarrow p1$

We use *folding* to perform *quantifier elimination*

Implementation Timings and Complexity

Example	$D0$	$D1$	$D2$	$D3$	$D4$
Number of Foldings	1	1	1	1	1
Number of Variables	10	4	8	12	16
Time (in seconds)	0.01	0.01	0.08	3.03	306
Total-Time (in seconds)	0.02	0.02	0.14	4.89	431

Example	$N1$	$N2$	$N3$	$N4$
Number of Foldings	2	4	4	16
Number of Variables	4	8	12	16
Time (in seconds)	0.02	0.08	0.23	1.09
Total-Time (in seconds)	0.03	49	1016	11025

Table: Execution times of the Folding Algorithm for various examples.

- If we assume that constraint projection takes constant time then our Folding Algorithm is in NP
- Since in $\mathbb{Q}$ and $\mathbb{R}$ constraint projection takes $\mathcal{O}(2^v)$ time (where v is the number of projected variables) we get that the complexity is $\mathcal{O}(2^v)$

Open Issues

- Extension to *non-admissible constraints*
- Extension to the domain $\mathbb{I}$ of the *Integers*
- Can our algorithm be lifted to an algorithm for solving the problem of *restricted matching modulo* $\mathbb{Q}$ (or $\mathbb{R}$)?