Interval Analysis on DAGs for global Optimization 1

Arnold Neumaier
and
Hermann Schichl

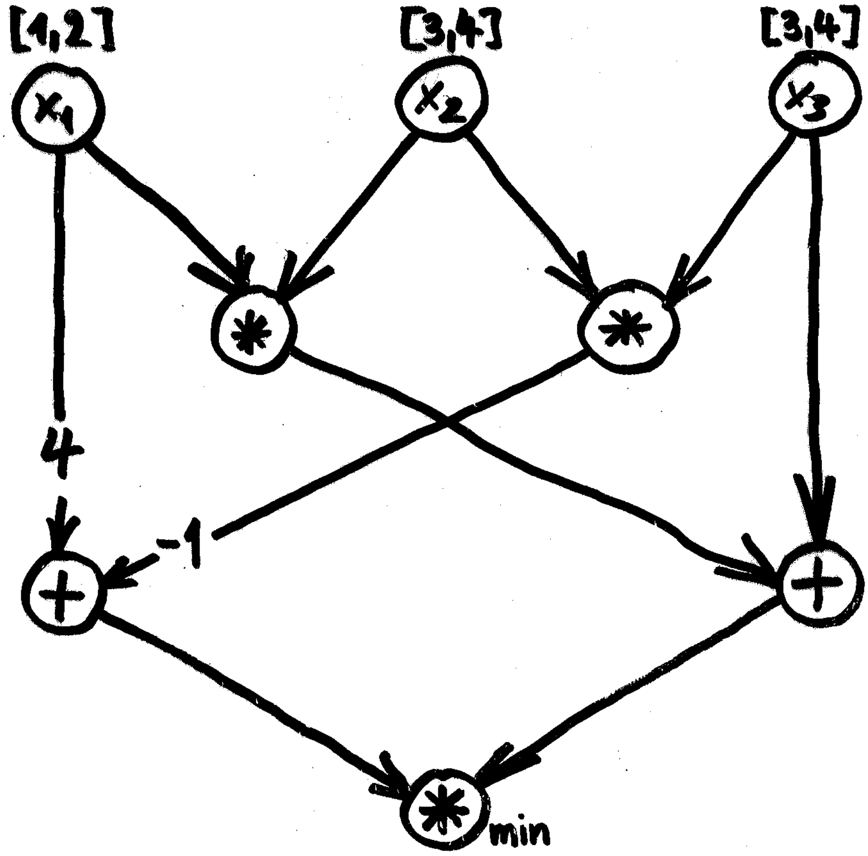
Institut für Mathematik,
Universität Wien,
Austria

email: Arnold.Neumaier@univie.ac.at,
Hermann.Schichl@esi.ac.at

http://www.mat.univie.ac.at/~neum

COCONUT Project
http://www.mat.univie.ac.at/coconut

Optimization Problem



$$\min \; (4x_1 - x_2 x_3)(x_1 x_2 + x_3)$$
$$\text{s.t.} \quad x_1 \in [1,2]$$
$$x_2 \in [3,4]$$
$$x_3 \in [3,4]$$

Directed Acyclic Graph

- directed graph

  i.e. edges have direction, represented by arrows

- acyclic

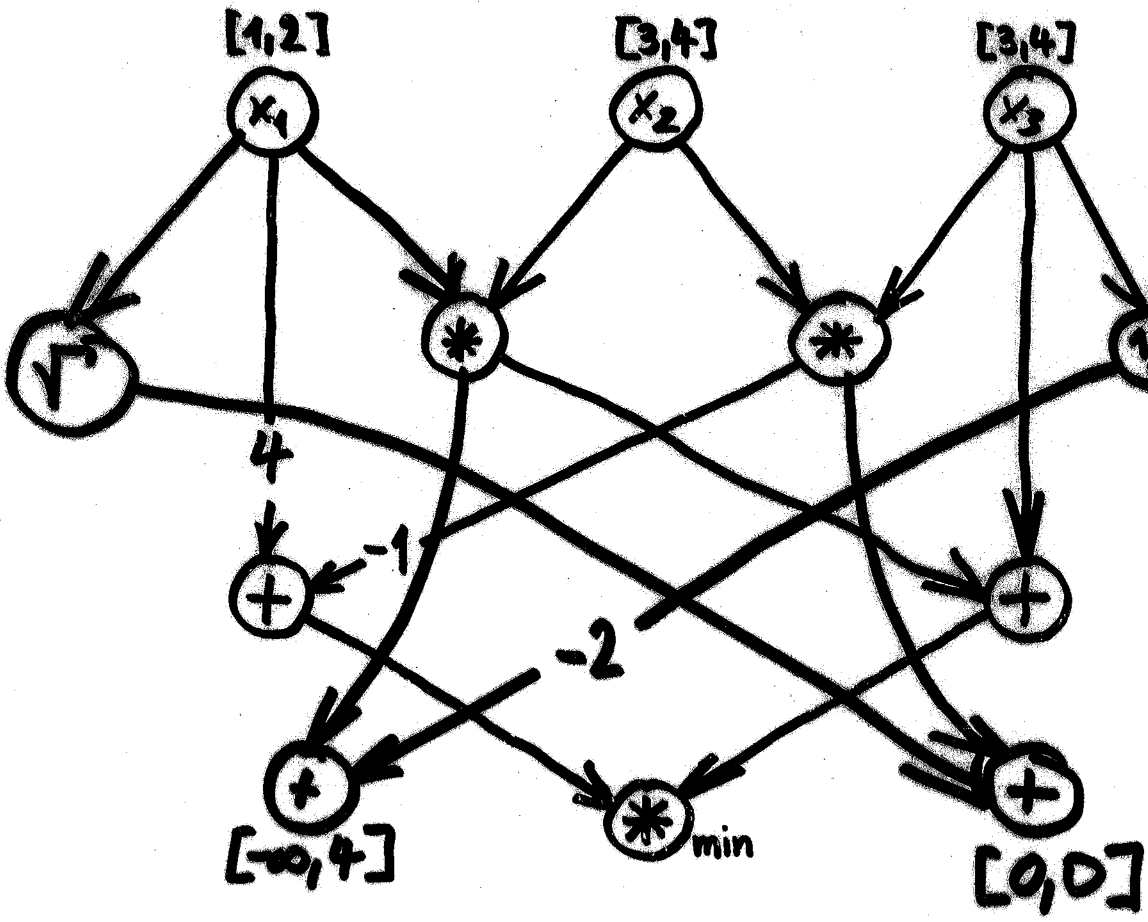
  i.e. impossible to return to the start node of a graph walk when following arrows

  acyclicity $\Leftrightarrow$ existence of a monotone numbering of the nodes (i.e. arrows always point towards bigger nodes)

- has roots no arrows emanating from

  leaves no arrows pointing to

Optimization Problem = DAG

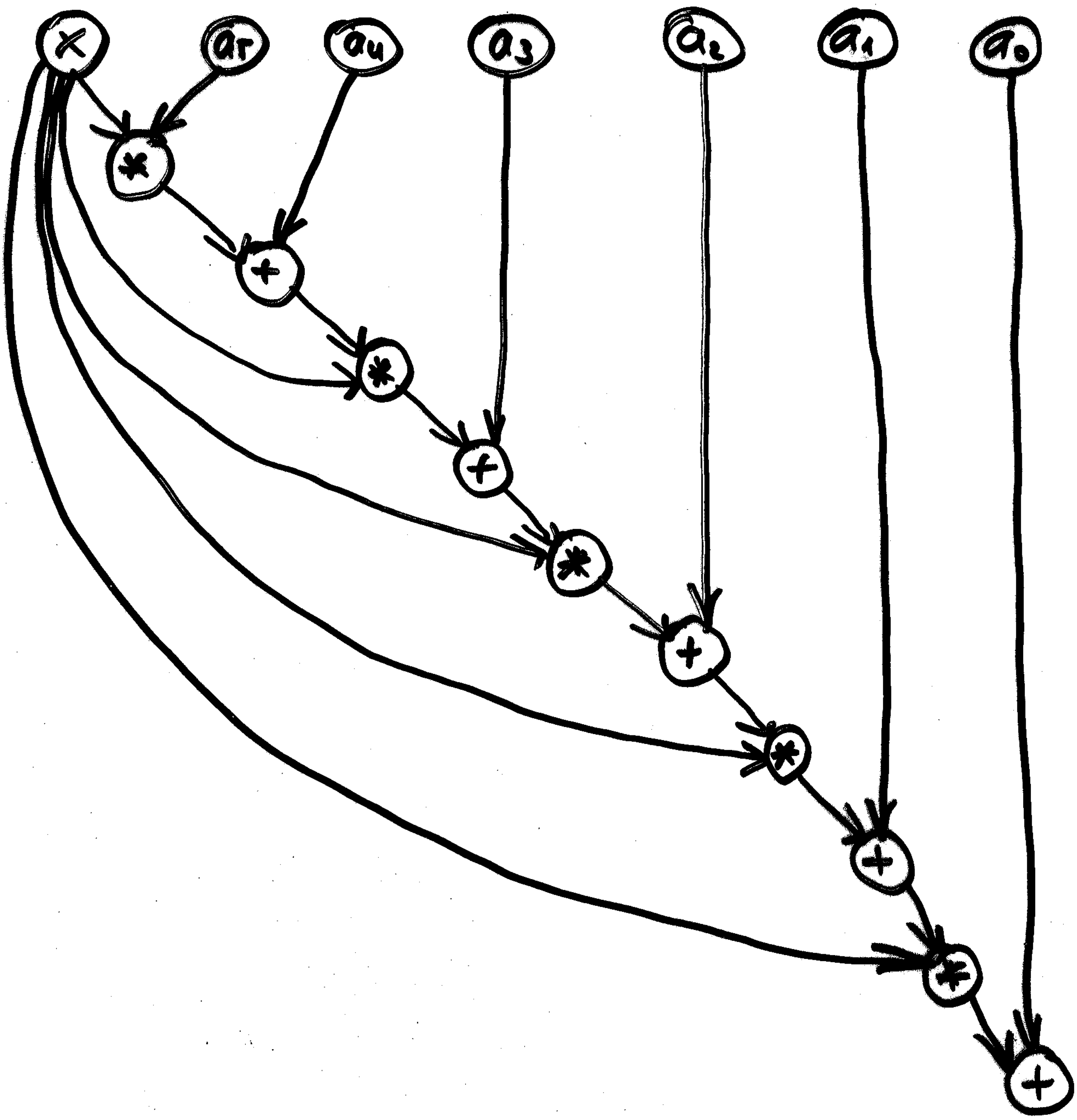


$$
\begin{aligned}
\min \quad & (4x_1 - x_2x_3)(x_1x_2 + x_3) \\
\text{s.t.} \quad & x_1 \in [1,2] \\
& x_2 \in [3,4] \\
& x_3 \in [3,4] \\
& x_1x_2 - 2x_3^2 \le 4 \\
& x_2x_3 + \sqrt{x_1} = 0
\end{aligned}
$$

'Every' global optimization problem can be represented by ONE DAG.

Example for a DAG:

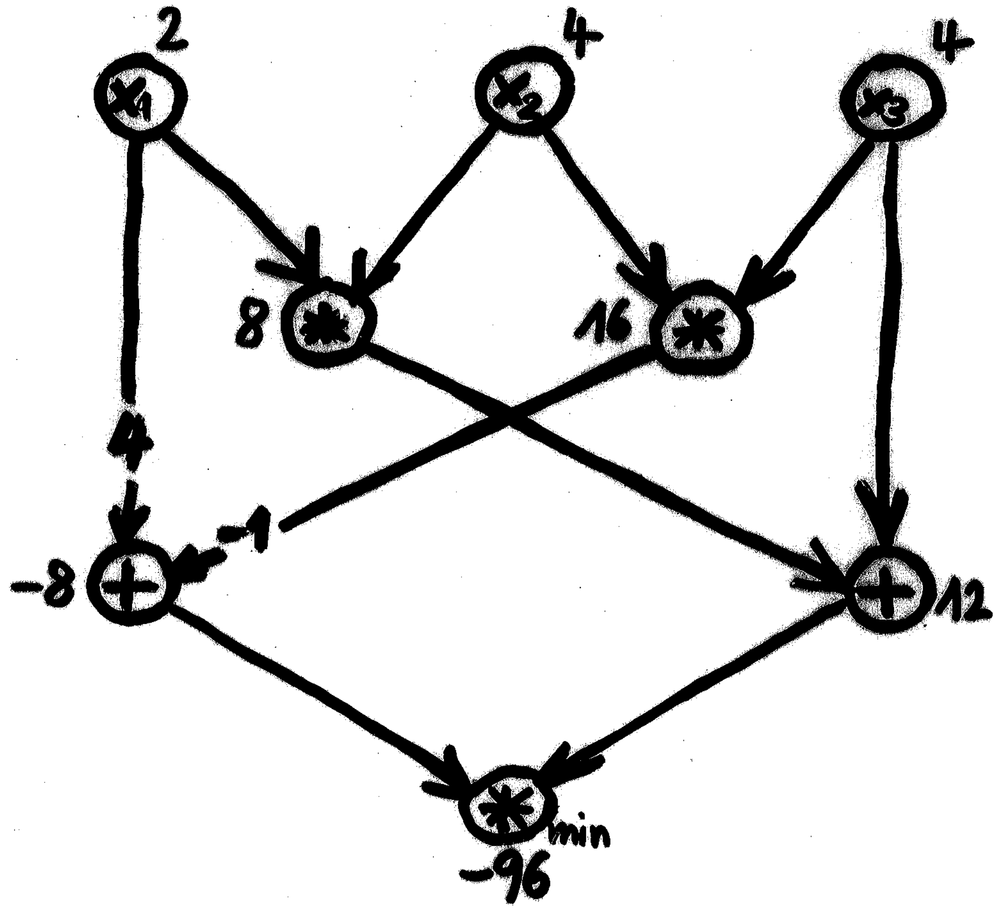
Horner scheme:



$((((a_5 x + a_4)x + a_3)x + a_2)x + a_1)x + a_0$

Forward Evaluation

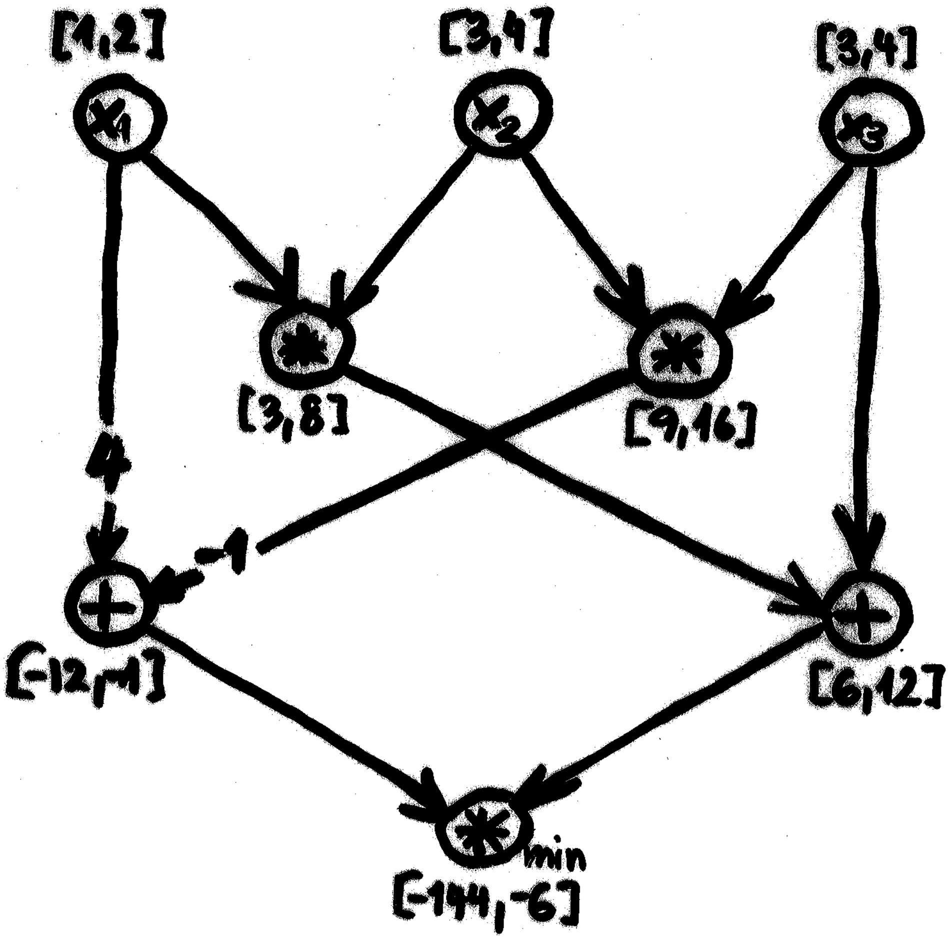
real function values



$$f(x) = \underbrace{\underbrace{(\underbrace{4\underset{2}{x_1}}_{8} - \underbrace{\underset{4}{x_2}\,\underset{4}{x_3}}_{16})}_{-8}\;\underbrace{(\underbrace{\underset{2}{x_1}\,\underset{4}{x_2}}_{8} + \underset{4}{x_3})}_{12}}_{-96}$$

Forward Evaluation

interval evaluation:



$f(x) = (4x_1 - x_2x_3)(x_1x_2 + x_3)$

[1,2] [3,4] [3,4] [1,2] [3,4] [3,4]

[4,8] [9,16] [3,8]

[-12,-1] [6,12]

[-144,-6]

Slopes 1

- linear approximation

$$f(x) = f(z) + f[z,x](x-z)$$

Krawczyk-Neumaier, Kolev

in 1D the slope is unique if continuous

$$f[z,x] = \frac{f(x)-f(z)}{x-z} \quad x \neq z$$

$$f[z,z] = f'(z)$$

- enclosure of the range

$$f(x) \in f(z) + f[z,\mathbf{x}](\mathbf{x}-z) \quad \forall x \in \mathbf{x}$$

has <u>quadratic approximation property</u>.

- most general with interval center

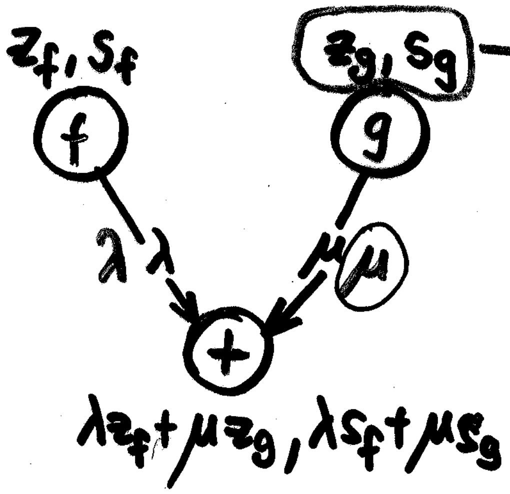
$$f(\mathbf{x}) \subseteq f(\mathbf{z}) + f[\mathbf{z},\mathbf{x}](\mathbf{x}-\mathbf{z})$$

the special case $\mathbf{x} = \mathbf{z}$ gives the interval derivative

$$f[\mathbf{z},\mathbf{z}] = f'(\mathbf{z})$$

Slopes 2

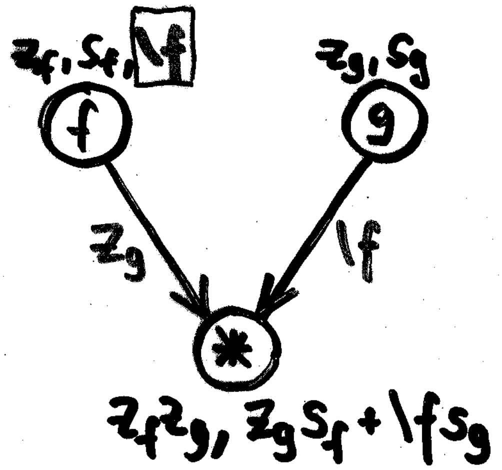
- slopes can be calculated automatically.
- on the DAG the recursive computation proceeds as follows:



z_g ... center of g
s_g ... slope of g

$$h = \lambda f + \mu g$$
$$z_h = \lambda z_f + \mu z_g$$
$$s_h = \lambda s_f + \mu s_g$$

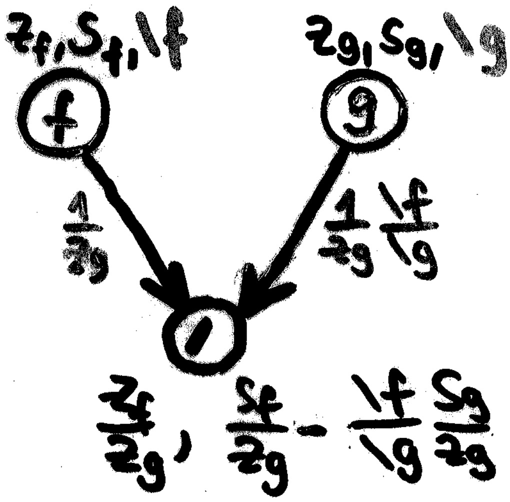
dependence of s_h on s_g



$$h = fg$$
$$z_h = z_f z_g$$
$$s_h = z_g s_f + \backslash f s_g$$

$\boxed{\backslash f}$ .. already known enclosure of f

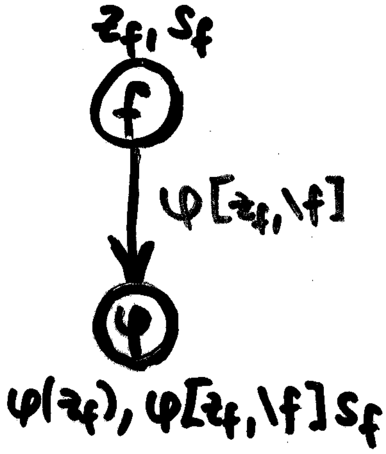
Slopes 3



$$h = f/g$$

$$z_h = z_f / z_g$$

$$s_h = \frac{s_f}{z_g} - \frac{\backslash f}{\backslash g}\frac{s_g}{z_g}$$



$$h = \varphi(f)$$

$$z_h = \varphi(z_f)$$

$$s_h = \varphi[z_f, \backslash f]\, s_f$$

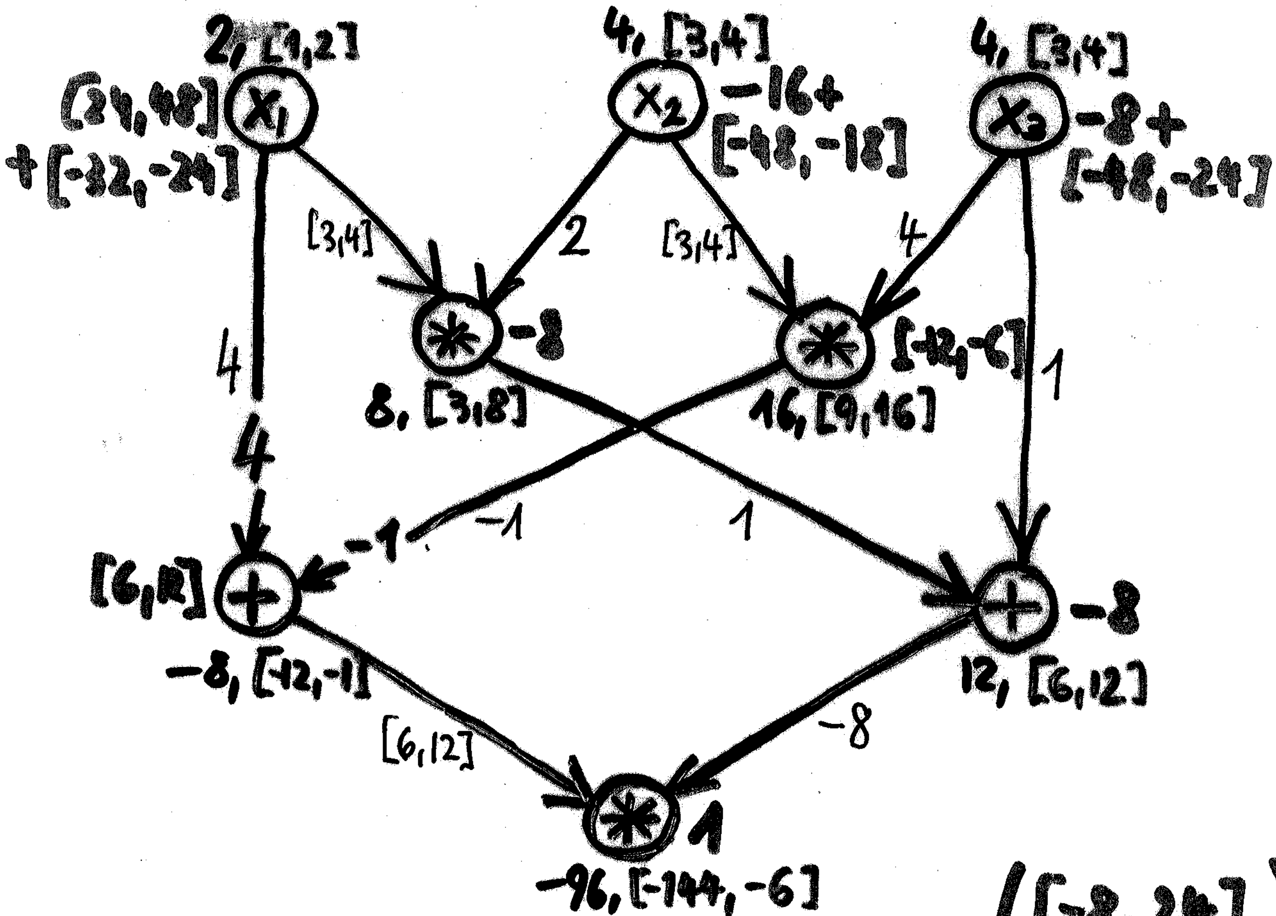
- Slopes for elementary functions:
  - Kolev: optimal for concave, convex

    $$\varphi[z, \backslash x] = \square \{ \varphi[z, \underline{x}], \varphi[z, \overline{x}] \} \quad \text{if } z \in \backslash x$$

  - otherwise

    $$\varphi[z, \backslash x] \subseteq \varphi'(\backslash x) \quad \text{if } z \in \backslash x$$

Backward Evaluation of Slopes



$$S = \begin{pmatrix} [-8, 24] \\ [-64, -34] \\ [-56, -32] \end{pmatrix}$$

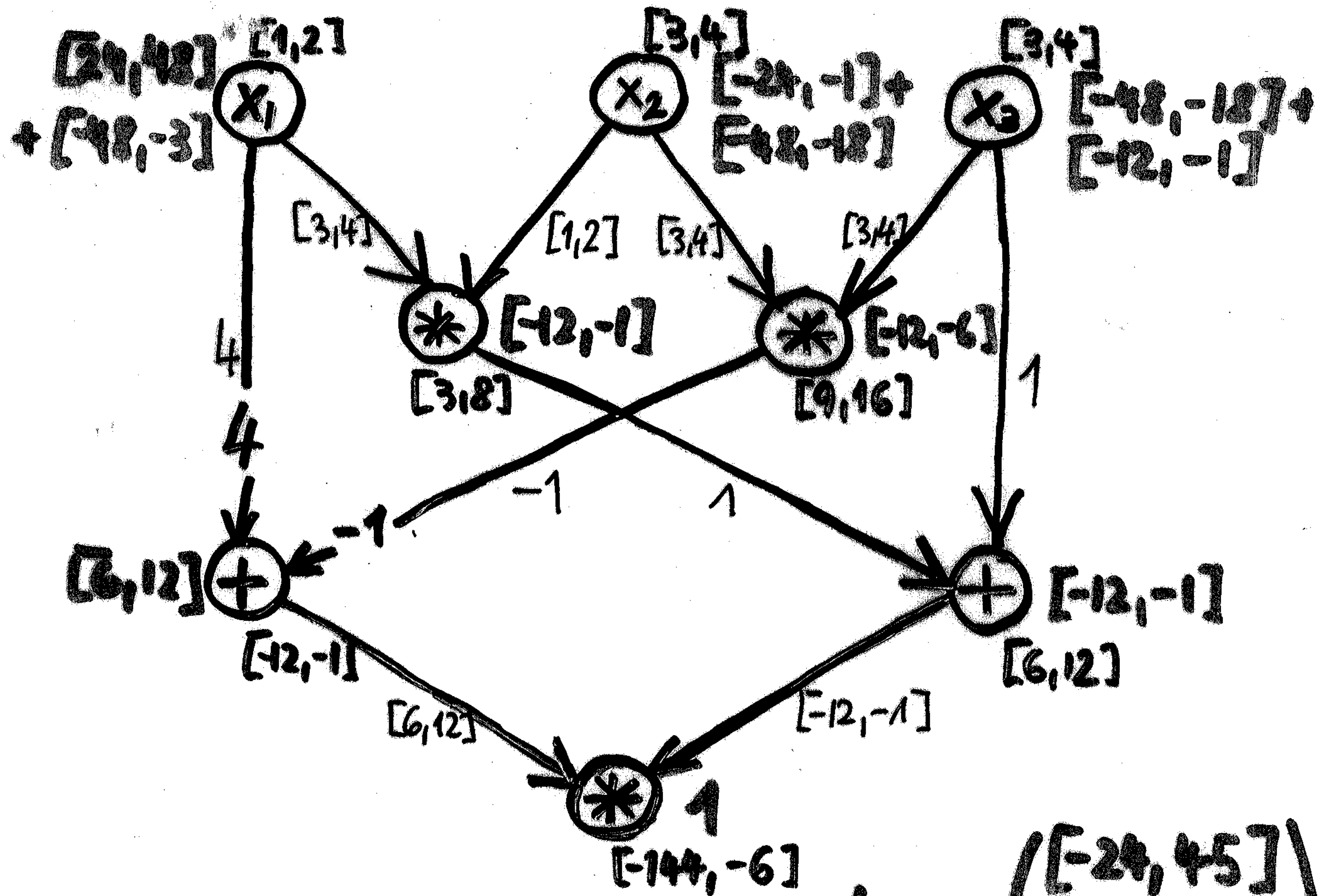
$$f(x) = (4x_1 - x_2 x_3)(x_1 x_2 + x_3)$$

- due to Bliek '91
- chain rule

$$(f \circ g)[z, \mathbf{x}] \subseteq f[g(z), g(\mathbf{x})]\, g[z, \mathbf{x}]$$

suggests backward evaluation like for gradients.

Interval Derivative



$$f'(x) = \begin{pmatrix} [-24,45] \\ [-72,-19] \\ [-60,-19] \end{pmatrix}$$

$$f(x) = (4x_1 - x_2x_3)(x_1x_2 + x_3)$$

- uses the well known chain rule

$$(f \circ g)'(x) \subseteq f'(g(x))\, g'(x)$$

- is not better than slope

$$\begin{pmatrix} [-24,45] \\ [-72,-19] \\ [-60,-19] \end{pmatrix} \supsetneq \begin{pmatrix} [-8,24] \\ [-64,-34] \\ [-56,-32] \end{pmatrix}$$

Constraint Propagation 1

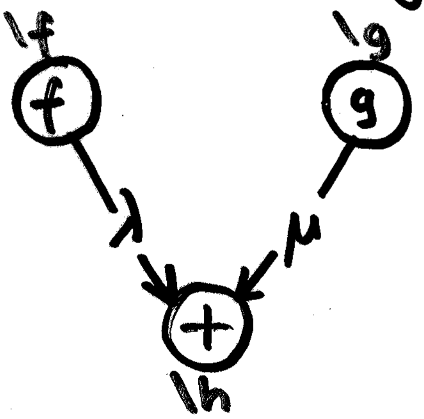
- search good feasible point

  here: $f(2,4,4) = -96 = f_{best}$

- introduce constraint

  $$f(x) \le f_{best}$$

- propagate ranges through the DAG until no significant improvement happens



$$h = \lambda f + \mu g$$

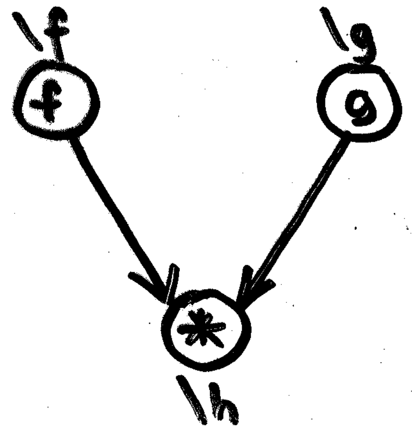
forward propagation

$$\backslash h := (\lambda \backslash f + \mu \backslash g) \cap \backslash h$$

backward propagation

$$\backslash f := \tfrac{1}{\lambda}(\backslash h - \mu \backslash g) \cap \backslash f$$

$$\backslash g := \tfrac{1}{\mu}(\backslash h - \lambda \backslash f) \cap \backslash g$$



$$h = fg$$

forward propagation

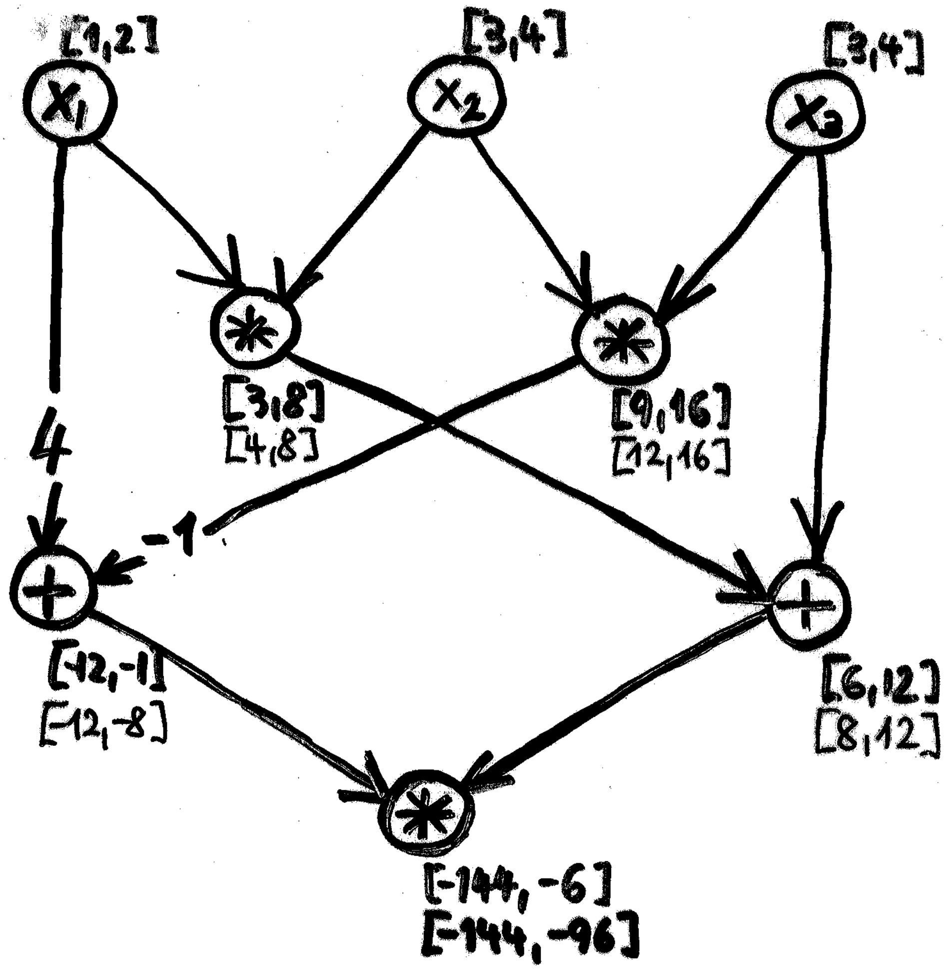
$$\backslash h := (\backslash f \cdot \backslash g) \cap \backslash h$$

backward propagation

$$\backslash f := (\backslash h / \backslash g) \cap \backslash f$$

$$\backslash g := (\backslash h / \backslash f) \cap \backslash g$$

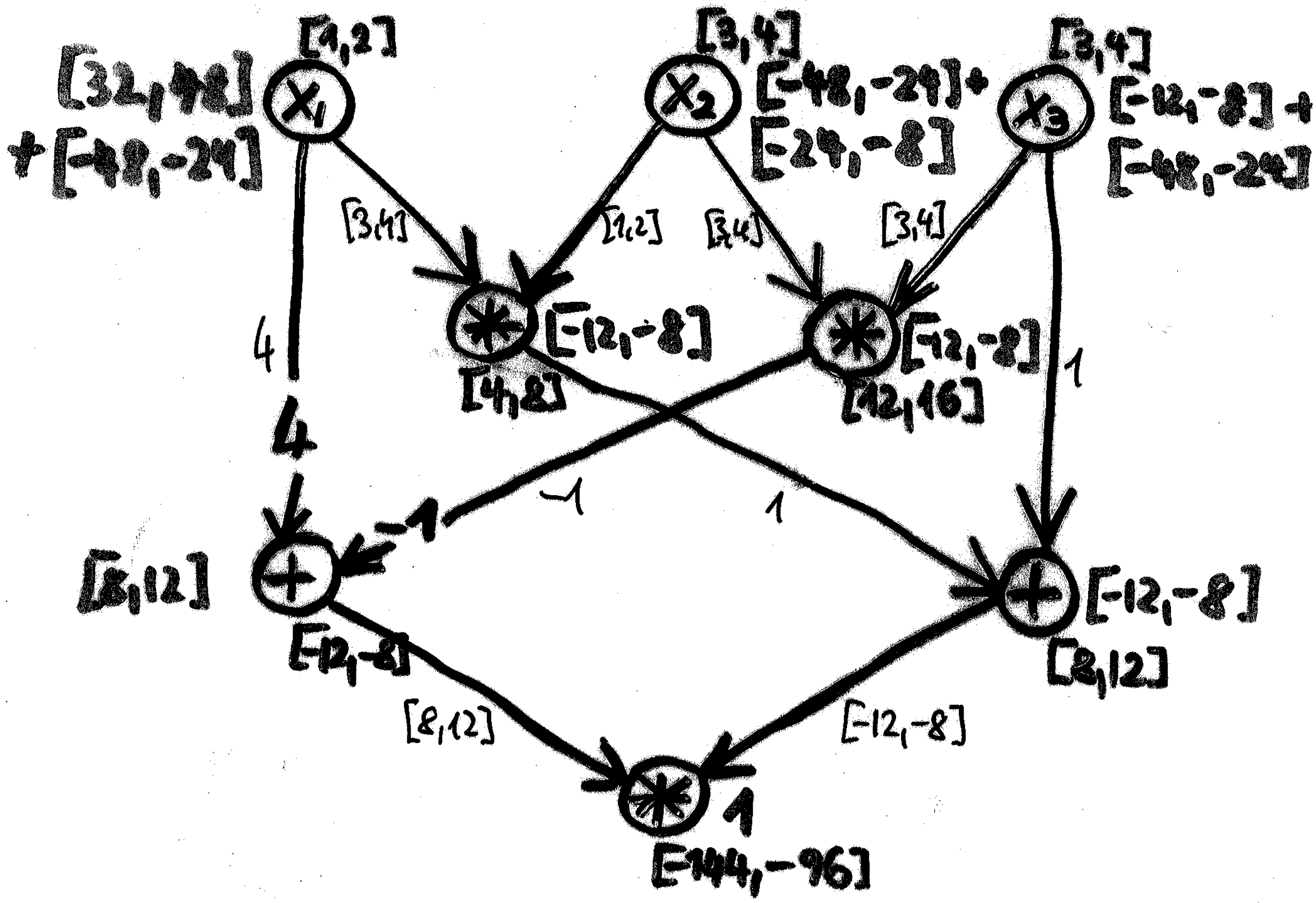
Constraint propagation 2



$$f(x) = (4x_1 - x_2x_3)(x_1x_2 + x_3)$$

- **introduce $f(x) \leq -96$**
- perform backward and forward propagation until no significant change happens

Interval derivative after CP

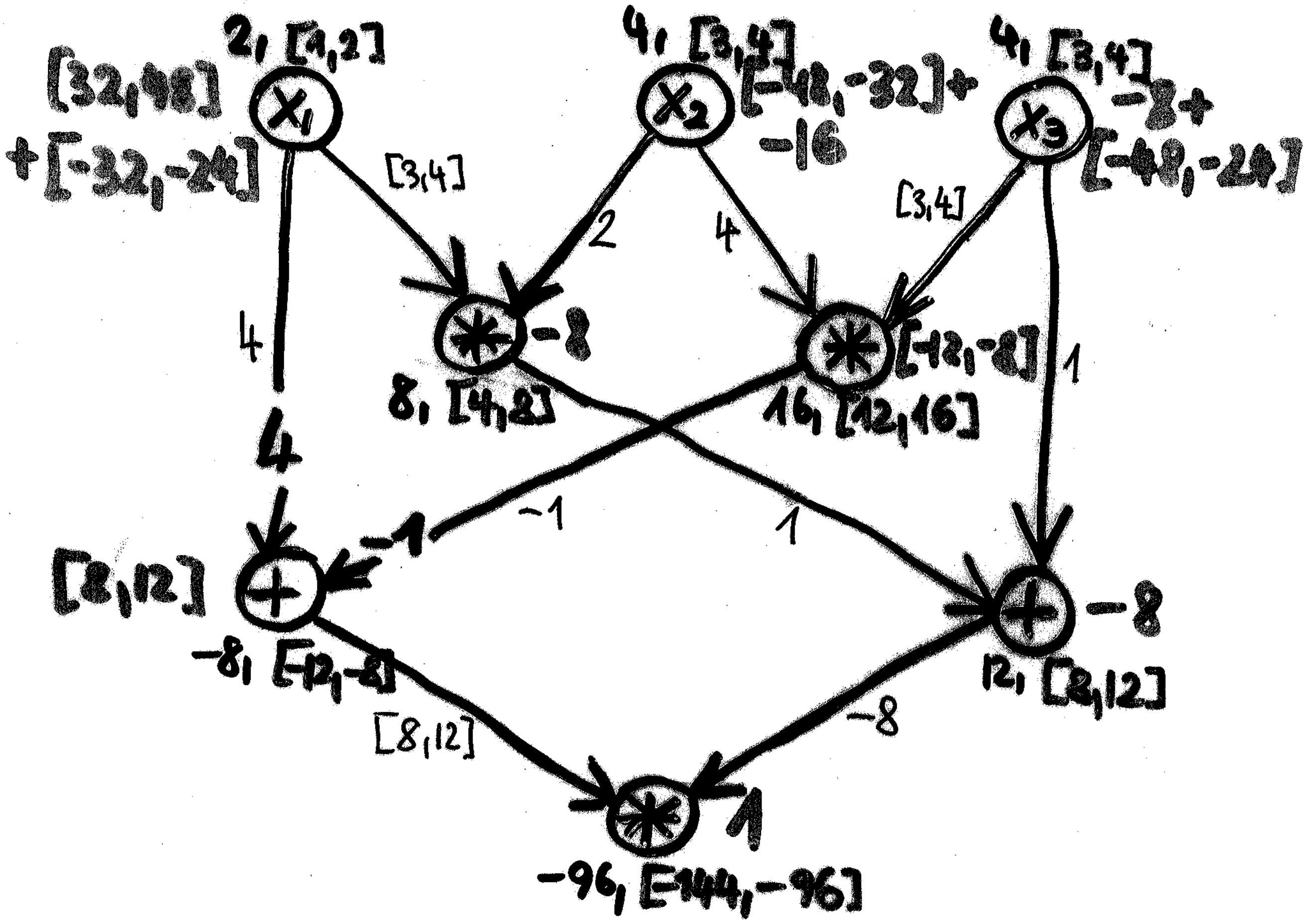


$$f(x) = (4x_1 - x_2x_3)(x_1x_2 + x_3)$$

- keep propagated ranges and use them during calculation
- gives better enclosures than usual method

$$\begin{pmatrix} [-16,24] \\ [-72,-32] \\ [-60,-32] \end{pmatrix} \subsetneq \begin{pmatrix} [-24,45] \\ [-72,-19] \\ [-60,-19] \end{pmatrix}$$

Slopes after CP



$$f(x)=(4x_1-x_2x_3)(x_1x_2+x_3)$$

- improves as well

$$\begin{pmatrix}[0,24]\\ [-64,-48]\\ [-56,-32]\end{pmatrix} \subsetneq \begin{pmatrix}[-8,24]\\ [-64,-34]\\ [-56,-32]\end{pmatrix}$$

Implementation 1

A. Rounding errors and guaranteed enclosures

$$f(x) \overset{?}{\in} f(z) + f[z,\backslash x](\backslash x - z)$$

due to rounding errors this cannot be guaranteed to be an enclosure

$$f(x) \in \backslash f + f[z_f,\backslash x](\backslash x - z_f) \qquad \forall x \in \backslash x$$

with $f(z_f) \in \backslash f$

$$g(y) \in \backslash g + g[z_g,\backslash y](\backslash y - z_g) \qquad \forall y \in \backslash y$$

$$g(f(x)) \in \backslash g + g[z_g,\backslash y](\backslash f + f[z_f,\backslash x](\backslash x - z_f) - z_g)$$

if $f(x) \in \backslash y \quad \forall x \in \backslash x$

$$\subseteq \backslash g + g[z_g,\backslash y](\backslash f - z_g) +$$
$$+ g[z_g,\backslash y] f[z_f,\backslash x](\backslash x - z_f)$$

here should be $z_g \approx f(z_f)$

Implementation 2

B. what to compute forward, what backward?

in addition to before consider

$$h(t) \in \backslash h + h[z_h, \backslash t](\backslash t - z_h)$$

and

$$h(g(f(x))) \subseteq \;\ldots$$

$$\ldots \subseteq \backslash h + \underline{h[z_h, \backslash t](\backslash g - z_h + g[z_g, \backslash y](\backslash f - z_g))} +$$

$$+ h[z_h, \backslash t]\, g[z_g, \backslash y]\, f[z_f, \backslash x](\backslash x - z_f)$$

if everything is computed backward:

$$\rightarrow h[z_h, \backslash t](\backslash g - z_h) + h[z_h, \backslash t]\, g[z_g, \backslash y](\backslash f - z_g)$$

which is a **worse** enclosure by subdistributivity $\quad (\backslash a(\backslash b + \backslash c) \subseteq \backslash a\backslash b + \backslash a\backslash c)$

so the center should be computed in forward mode

Implementation 3

C. Slopes of * and / :

- non unique

$$x_1 x_2 \in z_1 z_2 + \begin{pmatrix} \lambda \backslash x_2 + (1-\lambda) z_2 \\ \lambda z_1 + (1-\lambda) \backslash x_1 \end{pmatrix} \begin{pmatrix} \backslash x_1 - z_1 \\ \backslash x_2 - z_2 \end{pmatrix}$$

$$\frac{x_1}{x_2} \in \frac{z_1}{z_2} + \begin{pmatrix} \frac{\lambda}{z_2} + \frac{(1-\lambda)}{\backslash x_2} \\ -\frac{\lambda}{z_2}\frac{x_1}{x_2} - \frac{1-\lambda}{\backslash x_2}\frac{z_1}{z_2} \end{pmatrix} \begin{pmatrix} \backslash x_1 - z_1 \\ \backslash x_2 - z_2 \end{pmatrix}$$

for $\lambda \in \mathbb{R}$

- which one is best, and in what sense?
  - for / we can use $\frac{x_1}{x_2}$ after CP for $\lambda = 1$, which in addition has no subdistributivity problems in backward evaluation

$$S = \begin{pmatrix} \frac{1}{z_2} \\ -\frac{1}{z_2}\frac{x_1}{x_2} \end{pmatrix}$$

  - for * we can minimize the width of the resulting range

$$\lambda = \begin{cases} 0 & \text{if } \operatorname{rad}(\backslash x_1)|z_2| \le \operatorname{rad}(\backslash x_2)|z_1| \\ 1 & \text{otherwise} \end{cases}$$

- avoid case distinction by ordering products
- Horner scheme gives hint
- sort by ascending complexity of factors (i.e. roughly, increasing overest.)
- set $\lambda = 0$