Static analysis of memory manipulations by abstract interpretation

Algorithmics of tropical polyhedra, and application to abstract interpretation

Xavier ALLAMIGEON

EADS Innovation Works, SE/IS – Suresnes, France
CEA, LIST MeASI – Gif-sur-Yvette, France

November 30th, 2009

Context: bugs are everywhere

Software is omnipresent in highly critical systems:

Context: bugs are everywhere

Software is omnipresent in highly critical systems:

Context: bugs are everywhere

Software is omnipresent in highly critical systems:

Context: bugs are everywhere

Software is omnipresent in highly critical systems:

Context: bugs are everywhere

Software is omnipresent in highly critical systems:

Bugs in software may have disastrous consequences!

Context: bugs are everywhere

Software is omnipresent in highly critical systems:

Bugs in software may have disastrous consequences!

Memory manipulations

Refers to the manipulation of complex data structures in memory

- arrays, matrices
- character strings (`"Hello World!"`)
- lists, trees, *etc*

$\Longrightarrow$ widely used in modern programming languages: C, C++, Java, *etc*

Memory manipulations

Refers to the manipulation of complex data structures in memory

- arrays, matrices
- character strings (`"Hello World!"`)
- lists, trees, *etc*

$\Longrightarrow$ widely used in modern programming languages: C, C++, Java, *etc*

Memory manipulations is **error-prone** and **dangerous**

Memory manipulations

Refers to the manipulation of complex data structures in memory

- arrays, matrices
- character strings (`"Hello World!"`)
- lists, trees, *etc*

⟹ widely used in modern programming languages: C, C++, Java, *etc*

Memory manipulations is **error-prone** and **dangerous**, *e.g.* buffer overflows

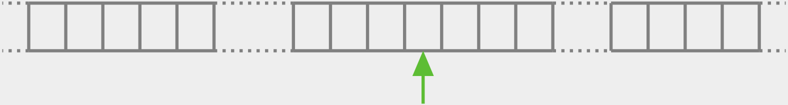
Memory manipulations

Refers to the manipulation of complex data structures in memory

- arrays, matrices
- character strings (`"Hello World!"`)
- lists, trees, *etc*

$\Longrightarrow$ widely used in modern programming languages: C, C++, Java, *etc*

Memory manipulations is **error-prone** and **dangerous**, *e.g.* buffer overflows



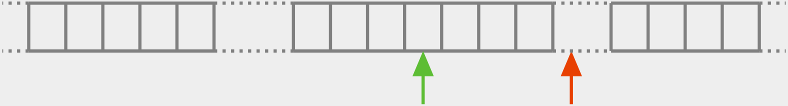
Memory manipulations

Refers to the manipulation of complex data structures in memory

- arrays, matrices
- character strings (`"Hello World!"`)
- lists, trees, *etc*

⟹ widely used in modern programming languages: C, C++, Java, *etc*

Memory manipulations is **error-prone** and **dangerous**, *e.g.* buffer overflows



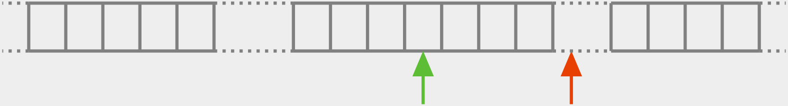
Memory manipulations

Refers to the manipulation of complex data structures in memory

- arrays, matrices
- character strings (`"Hello World!"`)
- lists, trees, *etc*

⟹ widely used in modern programming languages: C, C++, Java, *etc*

Memory manipulations is **error-prone** and **dangerous**, *e.g.* buffer overflows



Buffer overflows may lead to:

- software crashes (SEGFAULT)
- security holes, execution of arbitrary codes

What is this thesis about?

Static analysis of memory manipulations by abstract interpretation

= automatically analyzing the memory manipulations performed by a program

What is this thesis about?

Static analysis of memory manipulations by abstract interpretation

= automatically analyzing the memory manipulations performed by a program

Static analysis =

- automatic analysis technique
- the program is *not* executed (analysis on the source code)

What is this thesis about?

Static analysis of memory manipulations by abstract interpretation

= automatically analyzing the memory manipulations performed by a program

Static analysis =

- automatic analysis technique
- the program is *not* executed (analysis on the source code)

Abstract interpretation = compute properties which hold for all behaviors of the program

What is this thesis about?

Static analysis of memory manipulations by abstract interpretation

= automatically analyzing the memory manipulations performed by a program

Static analysis =

- automatic analysis technique
- the program is *not* executed (analysis on the source code)

Abstract interpretation = compute properties which hold for all behaviors of the program

- determines an **over**-approximation of the set of all behaviors
  $\Longrightarrow$ can not miss any bug

What is this thesis about?

Static analysis of memory manipulations by abstract interpretation

= automatically analyzing the memory manipulations performed by a program

Static analysis =

- automatic analysis technique
- the program is *not* executed (analysis on the source code)

Abstract interpretation = compute properties which hold for all behaviors of the program

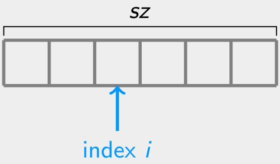
- determines an **over**-approximation of the set of all behaviors
  $\Longrightarrow$ can not miss any bug
- if not precise enough, it is not able to show the absence of bugs
  $\Longrightarrow$ false alarm

What is this thesis about? (2)

Static analysis of memory manipulations by abstract interpretation

Our approach:

Analyzing memory manipulations → Determining numerical properties



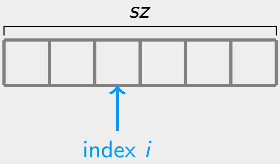
no buffer overflow iff $0 \leq i < sz$

What is this thesis about? (2)

Static analysis of memory manipulations by abstract interpretation

Our approach:

Analyzing memory manipulations → Determining numerical properties



no buffer overflow iff $0 \leq i < sz$

Automatically determining numerical invariants on:

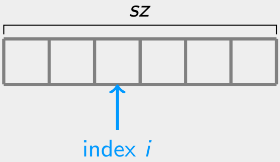
- the size of memory blocks

What is this thesis about? (2)

Static analysis of memory manipulations by abstract interpretation

Our approach:

Analyzing memory manipulations → Determining numerical properties



no buffer overflow iff $0 \leq i < sz$

Automatically determining numerical invariants on:

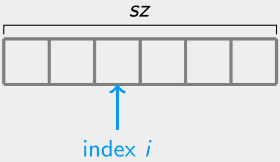
- the size of memory blocks
- the indexes of memory accesses

What is this thesis about? (2)

Static analysis of memory manipulations by abstract interpretation

Our approach:

Analyzing memory manipulations ⟶ Determining numerical properties



no buffer overflow iff $0 \leq i < sz$

Automatically determining numerical invariants on:

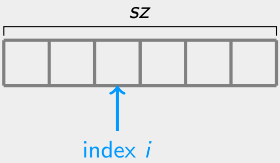
- the size of memory blocks
- the indexes of memory accesses
- the length of the strings:

What is this thesis about? (2)

Static analysis of memory manipulations by abstract interpretation

Our approach:

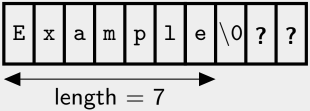
Analyzing memory manipulations → Determining numerical properties



no buffer overflow iff $0 \leq i < sz$

Automatically determining numerical invariants on:

- the size of memory blocks
- the indexes of memory accesses
- the length of the strings: *index of the first \0 character*



Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $v_1, \dots, v_d$. For instance:

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1 \boldsymbol{v}_1 + \cdots + a_d \boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

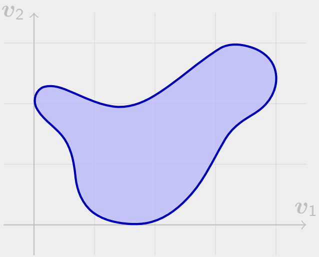
- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1\boldsymbol{v}_1 + \cdots + a_d\boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]
- provides a set of *abstract primitives* allowing to automatically compute **sound** properties

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1\boldsymbol{v}_1 + \dots + a_d\boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]
- provides a set of *abstract primitives* allowing to automatically compute **sound** properties

Different levels of precision:

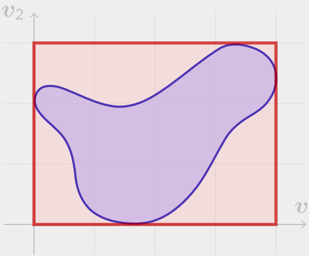


Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1\boldsymbol{v}_1 + \cdots + a_d\boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]
- provides a set of *abstract primitives* allowing to automatically compute **sound** properties

Different levels of precision:



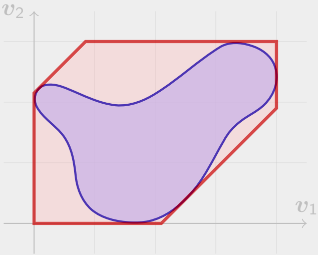
- intervals

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1\boldsymbol{v}_1 + \dots + a_d\boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]
- provides a set of *abstract primitives* allowing to automatically compute **sound** properties

Different levels of precision:



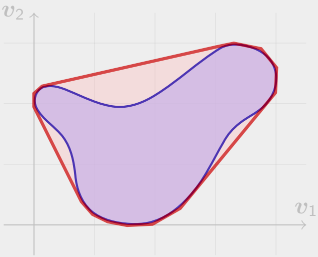
- intervals
- zones

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$. For instance:
  - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
  - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1\boldsymbol{v}_1 + \cdots + a_d\boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]
- provides a set of *abstract primitives* allowing to automatically compute **sound** properties

Different levels of precision:



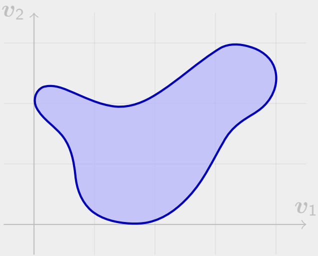
- intervals
- zones
- convex polyhedra

Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$. For instance:
    - intervals $a \leq \boldsymbol{v}_i \leq b$ [Cousot and Cousot, 1977]
    - zones $\boldsymbol{v}_i - \boldsymbol{v}_j \geq a$ [Miné, 2001]
    - convex polyhedra $a_1\boldsymbol{v}_1 + \dots + a_d\boldsymbol{v}_d \leq b$ [Cousot and Halbwachs, 1978]
- provides a set of *abstract primitives* allowing to automatically compute **sound** properties

Different levels of precision:



- intervals
- zones
- convex polyhedra

Remark: most existing numerical abstract domains are convex

The need of non-convex abstract domains

```
assume ($\mathtt{n} \geq 1$);
$\mathtt{s}$ := malloc($\mathtt{n}$);
$\mathtt{i}$ := 0;
while $\mathtt{i} \leq \mathtt{n} - 2$ do
    $\mathtt{s}[\mathtt{i}]$ := read();
    $\mathtt{i} := \mathtt{i} + 1$;
done;
$\mathtt{s}[\mathtt{i}]$ := \0;
$\mathtt{upper}$ := malloc($\mathtt{n}$);
$\mathtt{memcpy}(\mathtt{upper}, \mathtt{s}, \mathtt{n})$;
$\mathtt{i}$ := 0;
while $\mathtt{upper}[\mathtt{i}] \neq$ \0 do
    c := $\mathtt{upper}[\mathtt{i}]$;
    if $(c \geq 97) \wedge (c \leq 122)$ then
        $\mathtt{upper}[\mathtt{i}] := c - 32$;
    end;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
```

iterates up to the first \0

Typical memory manipulating program:

- reads a string $\mathtt{s}$ from standard input
- copies it in $\mathtt{upper}$ and capitalizes it

The need of non-convex abstract domains

```
assume $(\mathtt{n} \geq 1)$;
$\mathtt{s} :=$ malloc$(\mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{i} \leq \mathtt{n} - 2$ do
    $\mathtt{s}[\mathtt{i}] :=$ read();
    $\mathtt{i} := \mathtt{i} + 1$;
done;
$\mathtt{s}[\mathtt{i}] := \backslash 0$;
$\mathtt{upper} :=$ malloc$(\mathtt{n})$;
$\mathtt{memcpy}(\mathtt{upper}, \mathtt{s}, \mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{upper}[\mathtt{i}] \neq \backslash 0$ do
    $\mathtt{c} := \mathtt{upper}[\mathtt{i}]$;
    if $(\mathtt{c} \geq 97) \wedge (\mathtt{c} \leq 122)$ then
        $\mathtt{upper}[\mathtt{i}] := \mathtt{c} - 32$;
    end;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
```

Typical memory manipulating program:

- reads a string $\mathtt{s}$ from standard input
- copies it in $\mathtt{upper}$ and capitalizes it

Convex abstract domains: raise a false alarm

iterates up to the first $\backslash 0$

The need of non-convex abstract domains

```
assume $(\mathtt{n} \geq 1)$;
$\mathtt{s} := \text{malloc}(\mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{i} \leq \mathtt{n} - 2$ do
    $\mathtt{s}[\mathtt{i}] := \text{read}()$;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
$\mathtt{s}[\mathtt{i}] := \backslash 0$;
$\mathtt{upper} := \text{malloc}(\mathtt{n})$;
$\mathtt{memcpy}(\mathtt{upper}, \mathtt{s}, \mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{upper}[\mathtt{i}] \neq \backslash 0$ do
    $\mathtt{c} := \mathtt{upper}[\mathtt{i}]$;
    if $(\mathtt{c} \geq 97) \wedge (\mathtt{c} \leq 122)$ then
        $\mathtt{upper}[\mathtt{i}] := \mathtt{c} - 32$;
    end;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
```

Typical memory manipulating program:

- reads a string s from standard input
- copies it in upper and capitalizes it

Convex abstract domains: raise a false alarm

iterates up to the first \0

The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

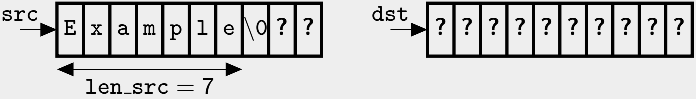
```
int i := 0;
for i = 0 to n-1 do
  dst[i] := src[i];
done;
```

The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

```
int i := 0;
for i = 0 to n-1 do
  dst[i] := src[i];
done;
```

- if $\mathtt{n} > \mathtt{len\_src}$,

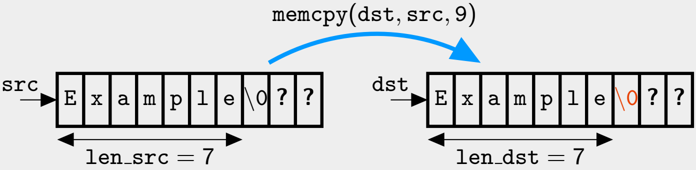


The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

```
int i := 0;
for i = 0 to n-1 do
  dst[i] := src[i];
done;
```

- if $\mathtt{n} > \mathtt{len\_src}$,



$\mathtt{memcpy}(\mathtt{dst}, \mathtt{src}, 9)$

src

E	x	a	m	p	l	e	\0	?	?

$\mathtt{len\_src} = 7$

dst

E	x	a	m	p	l	e	\0	?	?

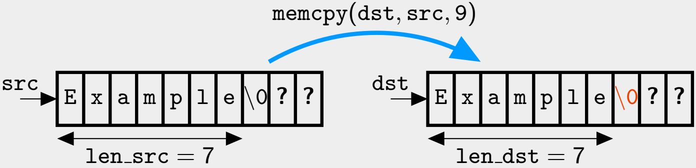
$\mathtt{len\_dst} = 7$

The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

```
1: int i := 0;
2: for i = 0 to n-1 do
3:   dst[i] := src[i];
4: done;
```

- if $\texttt{n} > \texttt{len\_src}$, $\texttt{len\_dst} = \texttt{len\_src}$

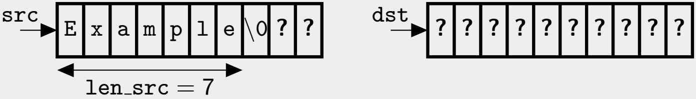


The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

```
int i := 0;
for i = 0 to n-1 do
  dst[i] := src[i];
done;
```

- if $\texttt{n} > \texttt{len\_src}$, $\texttt{len\_dst} = \texttt{len\_src}$
- if $\texttt{n} \leq \texttt{len\_src}$,

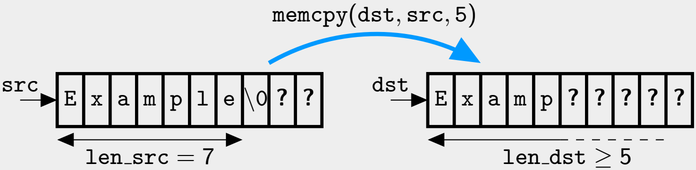


The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

```
int i := 0;
for i = 0 to n-1 do
  dst[i] := src[i];
done;
```

- if $\mathtt{n} > \mathtt{len\_src}$, $\mathtt{len\_dst} = \mathtt{len\_src}$
- if $\mathtt{n} \leq \mathtt{len\_src}$,

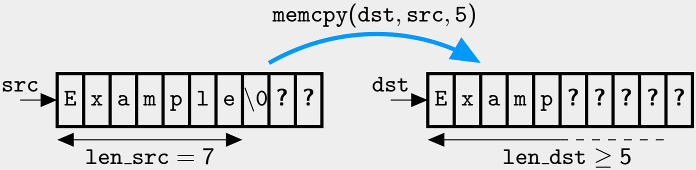


The need of non-convex abstract domains (2)

`memcpy(dst,src,n)` copies the first `n` characters of `src` to `dst`:

```
int i := 0;
for i = 0 to n-1 do
  dst[i] := src[i];
done;
```

- if $\mathtt{n} > \mathtt{len\_src}$, $\mathtt{len\_dst} = \mathtt{len\_src}$
- if $\mathtt{n} \leq \mathtt{len\_src}$, $\mathtt{len\_dst} \geq \mathtt{n}$

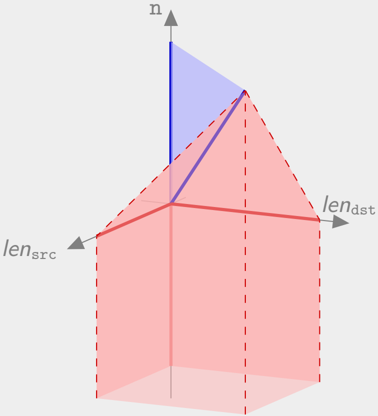


The need of non-convex abstract domains (3)

Disjunction of two cases:

- if $\texttt{n} > \texttt{len\_src}$, $\texttt{len\_dst} = \texttt{len\_src}$
- if $\texttt{n} \leq \texttt{len\_src}$, $\texttt{len\_dst} \geq \texttt{n}$

Not convex at all



Existing disjunctive techniques:

- disjunctive completion [Cousot and Cousot, 1979, Giacobazzi and Ranzato, 1998, Bagnara et al., 2006]
- trace partitioning [Mauborgne and Rival, 2005, Rival and Mauborgne, 2007]

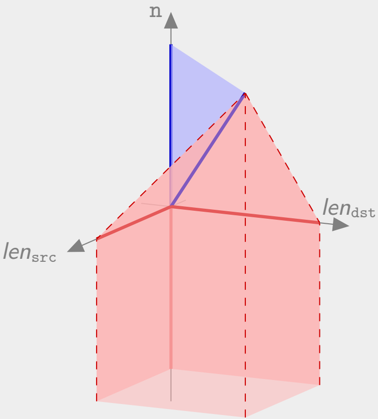
$\Longrightarrow$ not satisfactory

The need of non-convex abstract domains (3)

Disjunction of two cases:

- if $\mathtt{n} > \mathtt{len\_src}$, $\mathtt{len\_dst} = \mathtt{len\_src}$
- if $\mathtt{n} \leq \mathtt{len\_src}$, $\mathtt{len\_dst} \geq \mathtt{n}$

$$\iff \qquad \min(\mathit{len}_{\mathtt{src}}, \mathtt{n}) = \min(\mathit{len}_{\mathtt{dst}}, \mathtt{n})$$



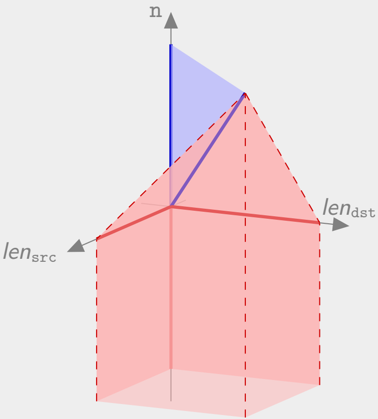
The need of non-convex abstract domains (3)

Disjunction of two cases:

- if $\mathtt{n} > \mathtt{len\_src}$, $\mathtt{len\_dst} = \mathtt{len\_src}$
- if $\mathtt{n} \leq \mathtt{len\_src}$, $\mathtt{len\_dst} \geq \mathtt{n}$

$$\iff \quad \min(\mathit{len}_{\mathtt{src}}, \mathtt{n}) = \min(\mathit{len}_{\mathtt{dst}}, \mathtt{n})$$

$$\iff \quad \max(-\mathit{len}_{\mathtt{src}}, -\mathtt{n}) = \max(-\mathit{len}_{\mathtt{dst}}, -\mathtt{n})$$



The need of non-convex abstract domains (3)

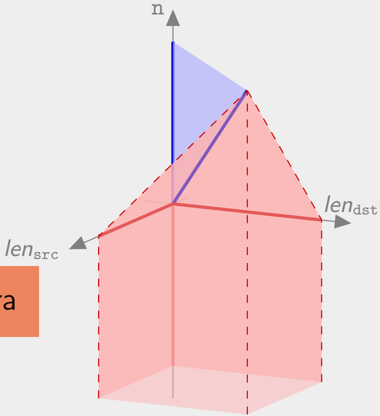
Disjunction of two cases:

- if $\texttt{n} > \texttt{len\_src}$, $\texttt{len\_dst} = \texttt{len\_src}$
- if $\texttt{n} \leq \texttt{len\_src}$, $\texttt{len\_dst} \geq \texttt{n}$

$$\iff \quad \min(len_{\texttt{src}}, \texttt{n}) = \min(len_{\texttt{dst}}, \texttt{n})$$

$$\iff \quad \max(-len_{\texttt{src}}, -\texttt{n}) = \max(-len_{\texttt{dst}}, -\texttt{n})$$

a linear equality ... in tropical algebra



Tropical algebra

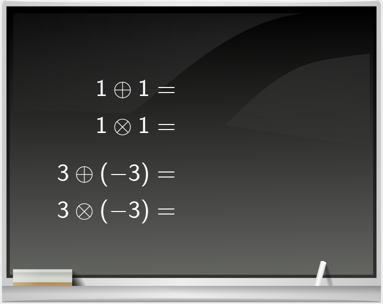
Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$

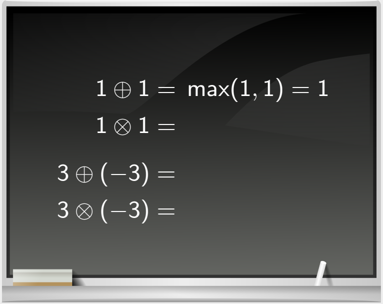


$$\begin{aligned} 1 \oplus 1 &= \\ 1 \otimes 1 &= \\ 3 \oplus (-3) &= \\ 3 \otimes (-3) &= \end{aligned}$$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$



$$1 \oplus 1 = \max(1,1) = 1$$

$$1 \otimes 1 =$$

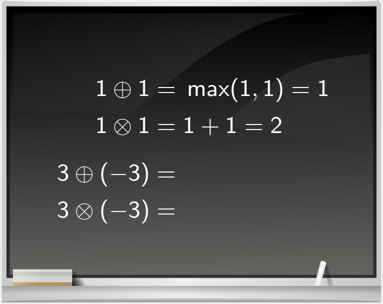
$$3 \oplus (-3) =$$

$$3 \otimes (-3) =$$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$



$$1 \oplus 1 = \max(1, 1) = 1$$

$$1 \otimes 1 = 1 + 1 = 2$$

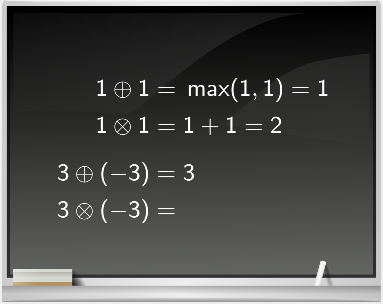
$$3 \oplus (-3) =$$

$$3 \otimes (-3) =$$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$



$$1 \oplus 1 = \max(1,1) = 1$$

$$1 \otimes 1 = 1 + 1 = 2$$

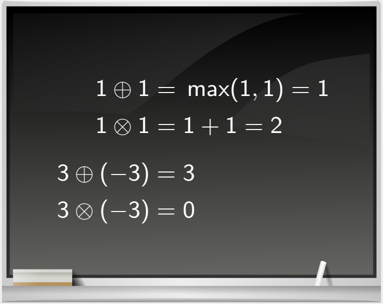
$$3 \oplus (-3) = 3$$

$$3 \otimes (-3) =$$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$



$$1 \oplus 1 = \max(1,1) = 1$$

$$1 \otimes 1 = 1 + 1 = 2$$

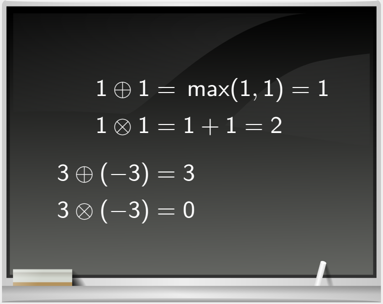
$$3 \oplus (-3) = 3$$

$$3 \otimes (-3) = 0$$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$
- $\mathbb{0} \overset{def}{=} -\infty$ is the zero element
- $\mathbb{1} \overset{def}{=} 0$ is the unit element



$$1 \oplus 1 = \max(1,1) = 1$$

$$1 \otimes 1 = 1 + 1 = 2$$

$$3 \oplus (-3) = 3$$

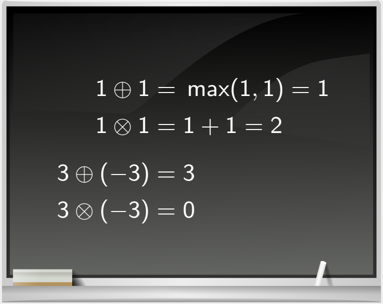
$$3 \otimes (-3) = 0$$

Tropical algebra

Tropical algebra refers to the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{ -\infty \}$ where:

- the addition $x \oplus y$ is $\max(x, y)$
- the multiplication $x \otimes y$ is $x + y$
- $\mathbb{0} \overset{def}{=} -\infty$ is the zero element
- $\mathbb{1} \overset{def}{=} 0$ is the unit element

The addition has **no inverse**! $\Longrightarrow$ semi-ring



$$1 \oplus 1 = \max(1, 1) = 1$$
$$1 \otimes 1 = 1 + 1 = 2$$

$$3 \oplus (-3) = 3$$
$$3 \otimes (-3) = 0$$

Tropical polyhedra

- *Tropical affine inequality* =

$$\alpha_0 + (\alpha_1 \times \boldsymbol{x}_1) + \cdots + (\alpha_d \times \boldsymbol{x}_d) \leq \beta_0 + (\beta_1 \times \boldsymbol{x}_1) + \cdots + (\beta_d \times \boldsymbol{x}_d)$$

Tropical polyhedra

- *Tropical affine inequality* =

$$\alpha_0 \oplus (\alpha_1 \otimes \boldsymbol{x}_1) \oplus \ldots \oplus (\alpha_d \otimes \boldsymbol{x}_d) \leq \beta_0 \oplus (\beta_1 \otimes \boldsymbol{x}_1) \oplus \ldots \oplus (\beta_d \otimes \boldsymbol{x}_d)$$

Tropical polyhedra

- *Tropical affine inequality* =

$$\max(\alpha_0, \alpha_1 + \boldsymbol{x}_1, \dots, \alpha_d + \boldsymbol{x}_d) \leq \max(\beta_0, \beta_1 + \boldsymbol{x}_1, \dots, \beta_d \boldsymbol{x}_d)$$

Tropical polyhedra

- *Tropical affine inequality* =

$$\alpha_0 \oplus (\alpha_1 \otimes \boldsymbol{x}_1) \oplus \ldots \oplus (\alpha_d \otimes \boldsymbol{x}_d) \leq \beta_0 \oplus (\beta_1 \otimes \boldsymbol{x}_1) \oplus \ldots \oplus (\beta_d \otimes \boldsymbol{x}_d)$$

- *Tropical polyhedra* = system of tropical affine inequalities

$$\max(-\mathit{len}_{\mathtt{src}}, -\mathtt{n}) = \max(-\mathit{len}_{\mathtt{dst}}, -\mathtt{n})$$

$$\iff \begin{cases} (-\mathit{len}_{\mathtt{src}}) \oplus (-\mathtt{n}) \leq (-\mathit{len}_{\mathtt{dst}}) \oplus (-\mathtt{n}) \\ (-\mathit{len}_{\mathtt{dst}}) \oplus (-\mathtt{n}) \leq (-\mathit{len}_{\mathtt{src}}) \oplus (-\mathtt{n}) \end{cases}$$

Tropical polyhedra

- *Tropical affine inequality* =

$$\alpha_0 \oplus (\alpha_1 \otimes \boldsymbol{x}_1) \oplus \ldots \oplus (\alpha_d \otimes \boldsymbol{x}_d) \leq \beta_0 \oplus (\beta_1 \otimes \boldsymbol{x}_1) \oplus \ldots \oplus (\beta_d \otimes \boldsymbol{x}_d)$$

- *Tropical polyhedra* = system of tropical affine inequalities

$$\max(-\mathit{len}_{\mathtt{src}}, -\mathtt{n}) = \max(-\mathit{len}_{\mathtt{dst}}, -\mathtt{n})$$

$$\iff \begin{cases} (-\mathit{len}_{\mathtt{src}}) \oplus (-\mathtt{n}) \leq (-\mathit{len}_{\mathtt{dst}}) \oplus (-\mathtt{n}) \\ (-\mathit{len}_{\mathtt{dst}}) \oplus (-\mathtt{n}) \leq (-\mathit{len}_{\mathtt{src}}) \oplus (-\mathtt{n}) \end{cases}$$

Idea: build a numerical domain based on tropical polyhedra

Tropical polyhedra (2)

Very studied in the litterature:

- Zimmermann [Zimmermann, 1977]
- Cuninghame-Green [Cuninghame-Green, 1979]
- Cohen, Gaubert, and Quadrat [Cohen et al., 2001, 2004]
- Nitica and Singer [Nitica and Singer, 2007]
- Briec, Horvath, and Rubinov [Briec and Horvath, 2004, Briec et al., 2005]
- Develin and Sturmfels [Develin and Sturmfels, 2004], Joswig [Joswig, 2005], Yu [Develin and Yu, 2007]
- Gaubert and Katz [Gaubert and Katz, 2006, 2007, 2009]

Tropical polyhedra (2)

Very studied in the litterature:

- Zimmermann [Zimmermann, 1977]
- Cuninghame-Green [Cuninghame-Green, 1979]
- Cohen, Gaubert, and Quadrat [Cohen et al., 2001, 2004]
- Nitica and Singer [Nitica and Singer, 2007]
- Briec, Horvath, and Rubinov [Briec and Horvath, 2004, Briec et al., 2005]
- Develin and Sturmfels [Develin and Sturmfels, 2004], Joswig [Joswig, 2005], Yu [Develin and Yu, 2007]
- Gaubert and Katz [Gaubert and Katz, 2006, 2007, 2009]

Algorithmics of tropical polyhedra: **little studied**

Tropical polyhedra (2)

Very studied in the litterature:

- Zimmermann [Zimmermann, 1977]
- Cuninghame-Green [Cuninghame-Green, 1979]
- Cohen, Gaubert, and Quadrat [Cohen et al., 2001, 2004]
- Nitica and Singer [Nitica and Singer, 2007]
- Briec, Horvath, and Rubinov [Briec and Horvath, 2004, Briec et al., 2005]
- Develin and Sturmfels [Develin and Sturmfels, 2004], Joswig [Joswig, 2005], Yu [Develin and Yu, 2007]
- Gaubert and Katz [Gaubert and Katz, 2006, 2007, 2009]

Algorithmics of tropical polyhedra: **little studied**

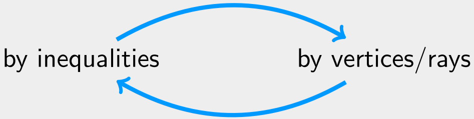
by inequalities by vertices/rays

Tropical polyhedra (2)

Very studied in the litterature:

- Zimmermann [Zimmermann, 1977]
- Cuninghame-Green [Cuninghame-Green, 1979]
- Cohen, Gaubert, and Quadrat [Cohen et al., 2001, 2004]
- Nitica and Singer [Nitica and Singer, 2007]
- Briec, Horvath, and Rubinov [Briec and Horvath, 2004, Briec et al., 2005]
- Develin and Sturmfels [Develin and Sturmfels, 2004], Joswig [Joswig, 2005], Yu [Develin and Yu, 2007]
- Gaubert and Katz [Gaubert and Katz, 2006, 2007, 2009]

Algorithmics of tropical polyhedra: **little studied**



by inequalities

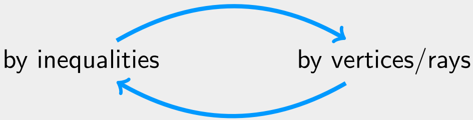
by vertices/rays

Tropical polyhedra (2)

Very studied in the litterature:

- Zimmermann [Zimmermann, 1977]
- Cuninghame-Green [Cuninghame-Green, 1979]
- Cohen, Gaubert, and Quadrat [Cohen et al., 2001, 2004]
- Nitica and Singer [Nitica and Singer, 2007]
- Briec, Horvath, and Rubinov [Briec and Horvath, 2004, Briec et al., 2005]
- Develin and Sturmfels [Develin and Sturmfels, 2004], Joswig [Joswig, 2005], Yu [Develin and Yu, 2007]
- Gaubert and Katz [Gaubert and Katz, 2006, 2007, 2009]

Algorithmics of tropical polyhedra: **little studied**


by inequalities

by vertices/rays

- central operation for computing with tropical polyhedra
- tropical analogue of vertex/facet enumeration problem

Contents: algorithmics of tropical polyhedra, and application to abstract interpretation

Goal of this thesis:

- build a new numerical abstract domain based on tropical polyhedra
- study the algorithmics of tropical polyhedra

Contents: algorithmics of tropical polyhedra, and application to abstract interpretation

Goal of this thesis:

- build a new numerical abstract domain based on tropical polyhedra
- study the algorithmics of tropical polyhedra

1. A better insight into tropical polyhedra

2. Algorithmics of tropical polyhedra

3. Tropical polyhedra based numerical domains

4. Conclusion

Contents

1. A better insight into tropical polyhedra
   - Representation by inequalities
   - Representation by generating set
   - Tropical Minkowski-Weyl theorem
2. Algorithmics of tropical polyhedra
3. Tropical polyhedra based numerical domains
4. Conclusion

Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}^d_{\max}$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}^d_{\max}$ of

$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

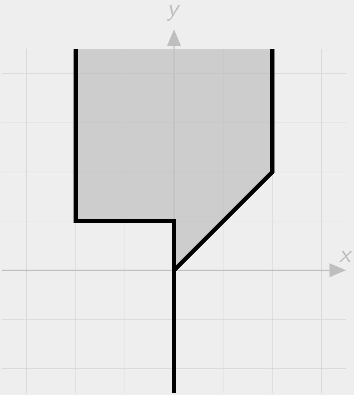
- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}^d_{\max}$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}^d_{\max}$ of

$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$$x \leq \max(y, 0)$$
$$0 \leq x + 2$$
$$x \leq 2$$
$$0 \leq \max(x, y - 1)$$



Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}^d_{\max}$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}^d_{\max}$ of

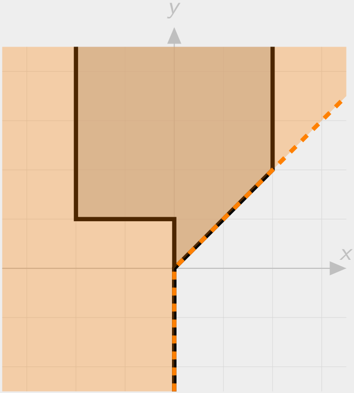
$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$x \leq \max(y, 0)$

$0 \leq x + 2$

$x \leq 2$

$0 \leq \max(x, y - 1)$



Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

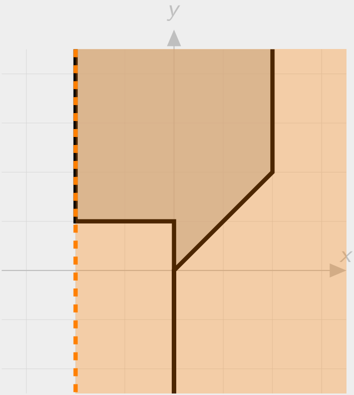
- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}_{\max}^d$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}_{\max}^d$ of

$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$$x \leq \max(y, 0)$$
$$0 \leq x + 2$$
$$x \leq 2$$
$$0 \leq \max(x, y - 1)$$



Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}^d_{\max}$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}^d_{\max}$ of

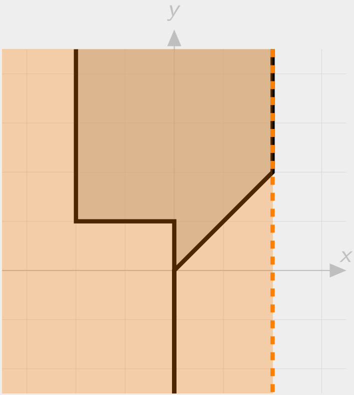
$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$x \leq \max(y, 0)$

$0 \leq x + 2$

$x \leq 2$

$0 \leq \max(x, y - 1)$



Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

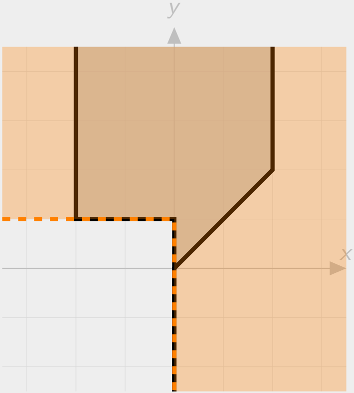
- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}_{\max}^d$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}_{\max}^d$ of

$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$$x \leq \max(y, 0)$$
$$0 \leq x + 2$$
$$x \leq 2$$
$$0 \leq \max(x, y - 1)$$



Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

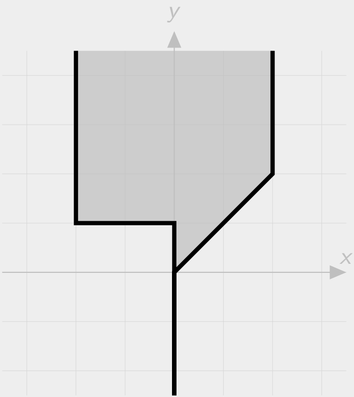
Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}^d_{\max}$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}^d_{\max}$ of

$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$$\begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \\ \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \oplus \begin{pmatrix} \mathbb{0} \\ \mathbb{1} \\ \mathbb{0} \\ \mathbb{1} \end{pmatrix}$$

$$\leq \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ 2 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \\ \mathbb{1} & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \oplus \begin{pmatrix} \mathbb{1} \\ \mathbb{0} \\ 2 \\ \mathbb{0} \end{pmatrix}$$



Tropical polyhedra as system of inequalities

Tropical polyhedra are the analogues of convex polyhedra in tropical algebra

Two possible representations:

- as the solutions of a system of tropical affine inequalities,
- or as the convex hull of a finitely many points and rays.

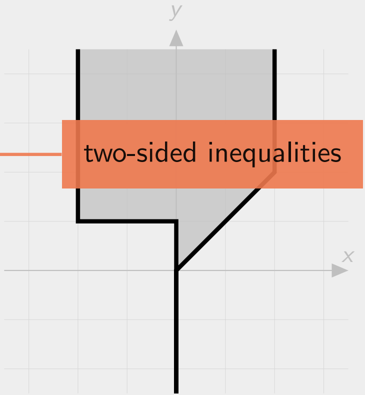
Definition (Inequality form)

A tropical polyhedron of $\mathbb{R}^d_{\max}$ is the set of the solutions $\boldsymbol{x} \in \mathbb{R}^d_{\max}$ of

$$A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$$

$$\begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \\ \mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \oplus \begin{pmatrix} \mathbb{0} \\ \mathbb{1} \\ \mathbb{0} \\ \mathbb{1} \end{pmatrix}$$

$$\leq \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ 2 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \\ \mathbb{1} & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \oplus \begin{pmatrix} \mathbb{1} \\ \mathbb{0} \\ 2 \\ \mathbb{0} \end{pmatrix}$$

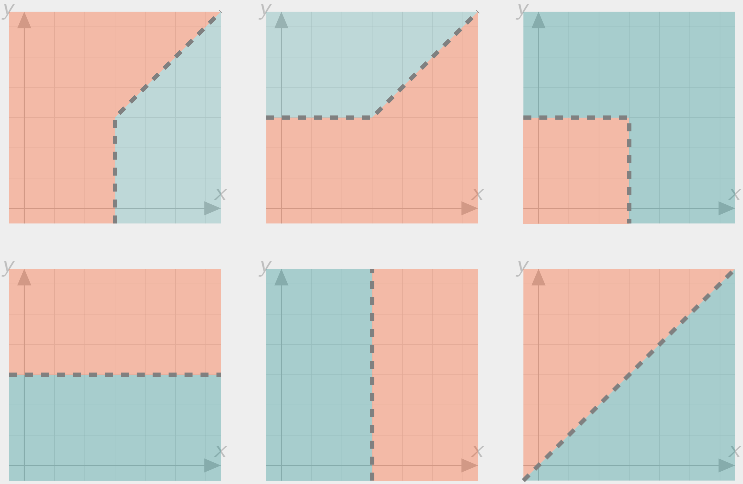


two-sided inequalities

Tropical halfspaces

Tropical halfspace = set of the solutions $x \in \mathbb{R}^d_{\max}$ of an affine inequality

$$ax \oplus c \leq bx \oplus d$$



Contents

1 A better insight into tropical polyhedra

Representation by inequalities

Representation by generating set

Tropical Minkowski-Weyl theorem

2 Algorithmics of tropical polyhedra

3 Tropical polyhedra based numerical domains

4 Conclusion

Tropical polyhedra = convex hull of generators

Definition (Generating representation)

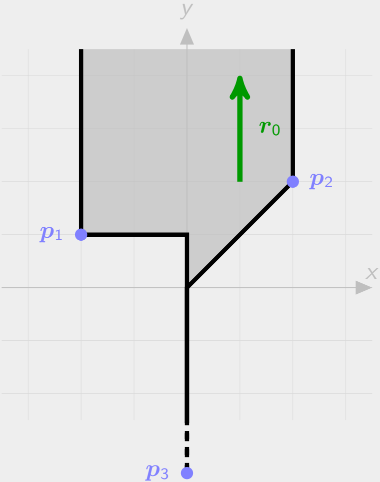
A tropical polyhedron is formed by the combinations of finitely many:

- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

where $\bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}$.



Tropical polyhedra = convex hull of generators

Definition (Generating representation)

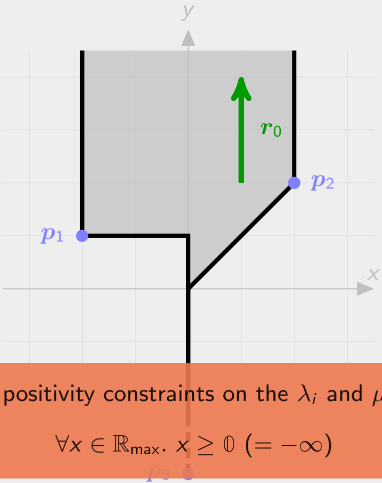
A tropical polyhedron is formed by the combinations of finitely many:

- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

where $\bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}$.



no positivity constraints on the λ_i and μ_j:

$$\forall x \in \mathbb{R}_{\max}.\ x \geq \mathbb{0}\ (= -\infty)$$

Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:

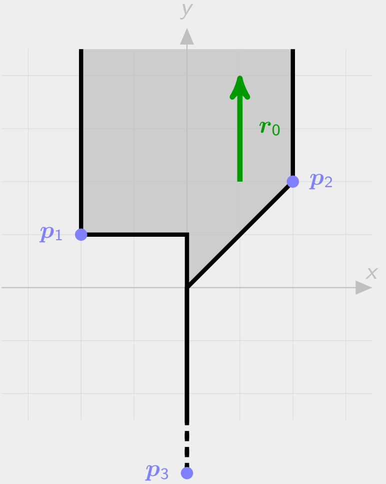
- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

$$\text{where } \bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}.$$

The couple (P, R) is a *generating representation*.



Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:

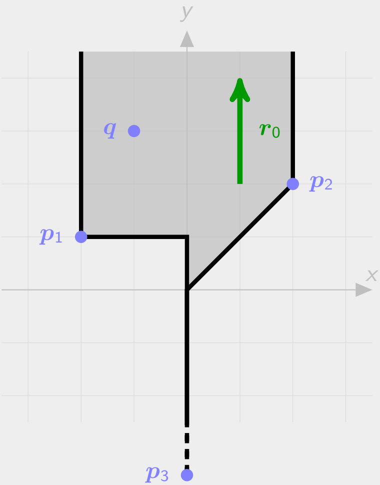
- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

where $\bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}$.

The couple (P, R) is a *generating representation*.



Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:

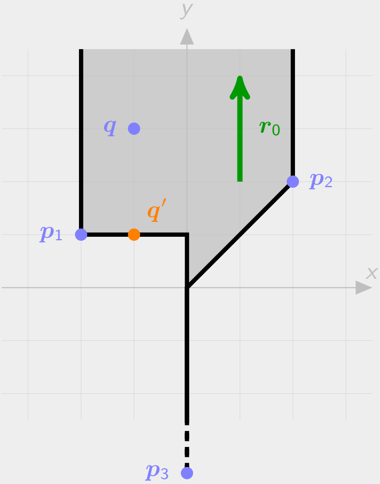
- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

where $\bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}$.

The couple (P, R) is a *generating representation*.



Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:

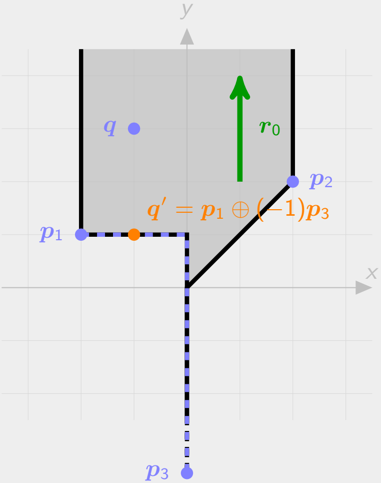
- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

where $\bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}$.

The couple (P, R) is a *generating representation*.



Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:

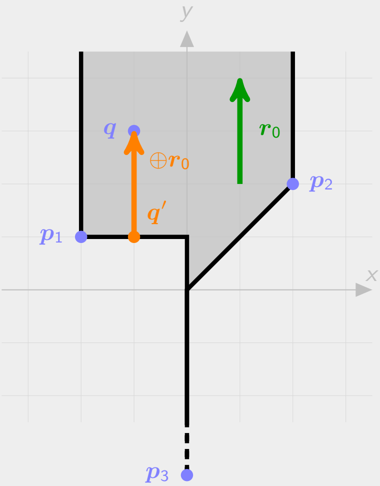
- points $\boldsymbol{p}_i \in P$,
- and of rays $\boldsymbol{r}_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i \boldsymbol{p}_i \oplus \bigoplus_{j=1}^{q} \mu_j \boldsymbol{r}_j$$

where $\bigoplus_{i=1}^{p} \lambda_i = \mathbb{1}$.

The couple (P, R) is a *generating representation*.



Contents

1. A better insight into tropical polyhedra
   - Representation by inequalities
   - Representation by generating set
   - Tropical Minkowski-Weyl theorem
2. Algorithmics of tropical polyhedra
3. Tropical polyhedra based numerical domains
4. Conclusion

Tropical Minkowski-Weyl theorem

Theorem ([Gaubert and Katz, 2006])

The two definitions of tropical polyhedra:

- *as the solution of a system of tropical affine inequalities*
- *as the convex hull of points and rays*

are equivalent.

Tropical Minkowski-Weyl theorem

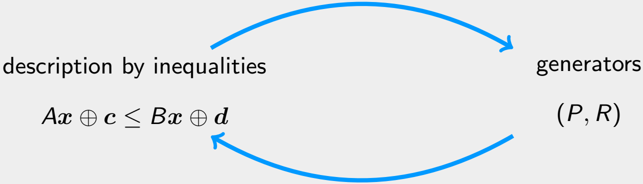
Theorem ([Gaubert and Katz, 2006])

The two definitions of tropical polyhedra:

- *as the solution of a system of tropical affine inequalities*
- *as the convex hull of points and rays*

are equivalent.

Underlying algorithmic problems:



description by inequalities

$\boldsymbol{Ax} \oplus \boldsymbol{c} \leq \boldsymbol{Bx} \oplus \boldsymbol{d}$

generators

(P, R)

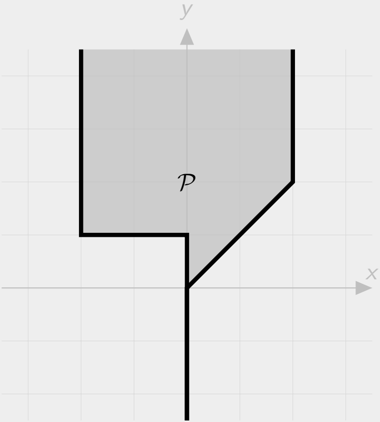
Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
   - Tropical double description method
   - Combinatorial characterization of extreme points
   - From inequalities to generators
   - Some other results
3. Tropical polyhedra based numerical domains
4. Conclusion

Tropical double description method

= **incremental** method computing generators from inequalities

$$x \leq y \oplus \mathbb{1}$$
$$\mathbb{1} \leq 2x$$
$$x \leq 2$$
$$\mathbb{1} \leq x \oplus (-1)y$$

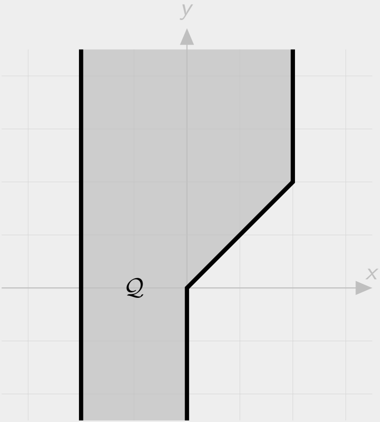


Tropical double description method

$=$ **incremental** method computing generators from inequalities

$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathbb{1} \leq x \oplus (-1)y$$

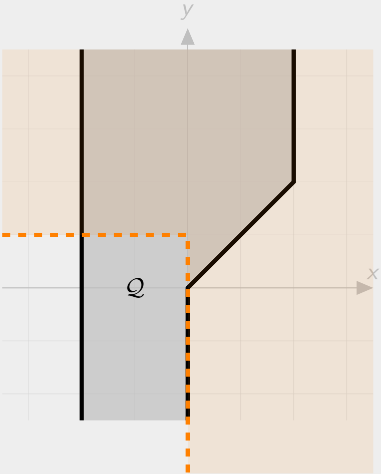


Tropical double description method

= **incremental** method computing generators from inequalities

$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$



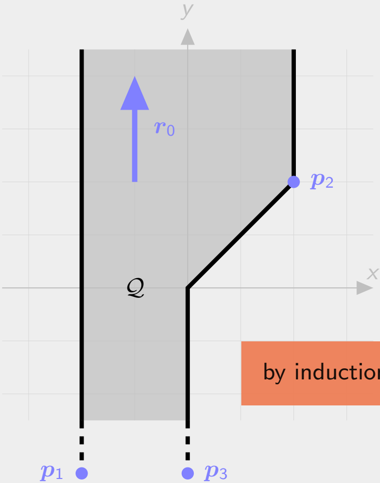
Tropical double description method

$=$ **incremental** method computing generators from inequalities

$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$

Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$



by induction

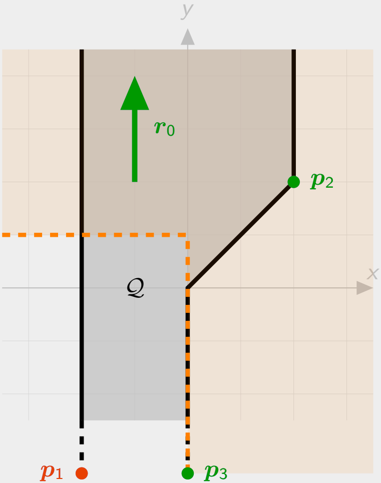
Tropical double description method

= **incremental** method computing generators from inequalities

$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$

Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$



Tropical double description method

$=$ **incremental** method computing generators from inequalities

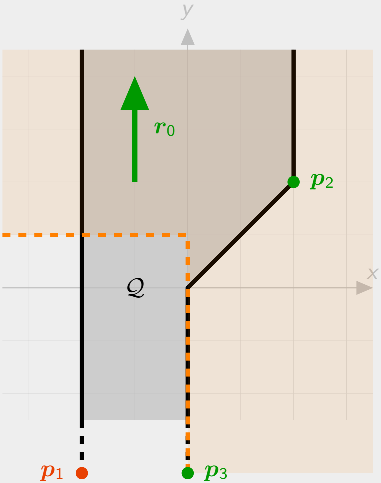
$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$

Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$

Intersection of $\mathcal{Q}$ and $\mathcal{H}$ generated by:

- generators of $\mathcal{Q}$ in $\mathcal{H}$



Tropical double description method

= **incremental** method computing generators from inequalities

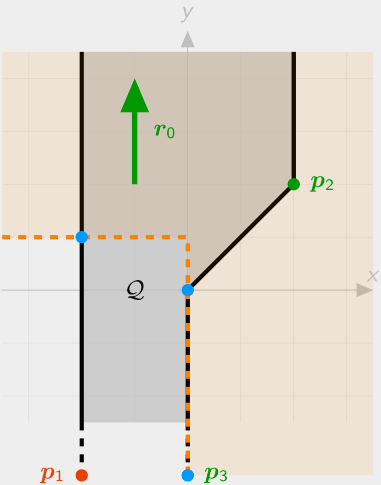
$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$

Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$

Intersection of $\mathcal{Q}$ and $\mathcal{H}$ generated by:

- generators of $\mathcal{Q}$ in $\mathcal{H}$
- combinations of green and red generators of $\mathcal{Q}$ lying on the boundary of $\mathcal{H}$

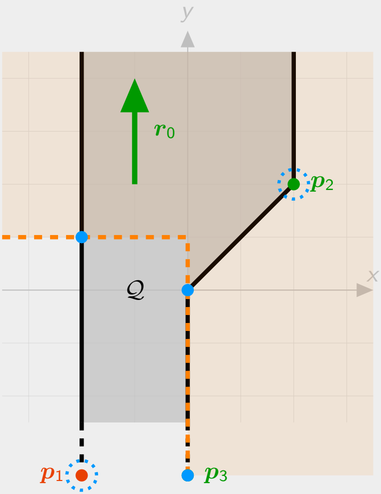


Tropical double description method

$=$ **incremental** method computing generators from inequalities

$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$



Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$

Intersection of $\mathcal{Q}$ and $\mathcal{H}$ generated by:

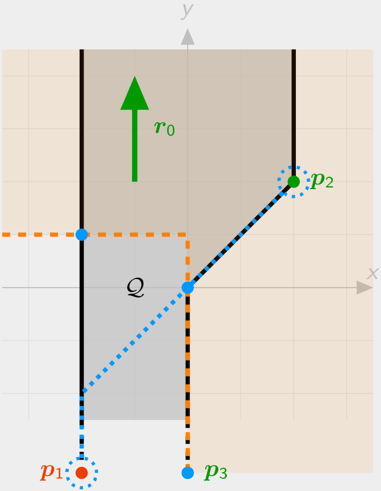
- generators of $\mathcal{Q}$ in $\mathcal{H}$
- combinations of green and red generators of $\mathcal{Q}$ lying on the boundary of $\mathcal{H}$

Tropical double description method

= **incremental** method computing generators from inequalities

$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$



Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$

Intersection of $\mathcal{Q}$ and $\mathcal{H}$ generated by:

- generators of $\mathcal{Q}$ in $\mathcal{H}$
- combinations of green and red generators of $\mathcal{Q}$ lying on the boundary of $\mathcal{H}$

Tropical double description method

= **incremental** method computing generators from inequalities

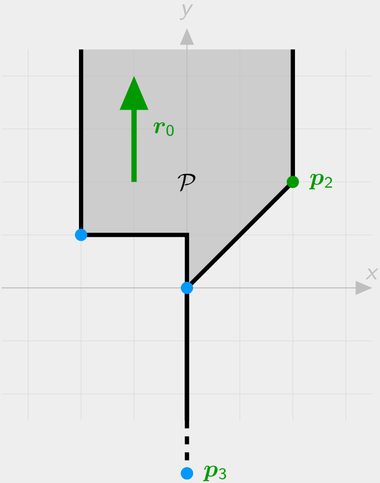
$$\mathcal{Q} : \begin{cases} x \leq y \oplus \mathbb{1} \\ \mathbb{1} \leq 2x \\ x \leq 2 \end{cases}$$

$$\mathcal{H} : \quad \mathbb{1} \leq x \oplus (-1)y$$

Generators of $\mathcal{Q}$: $\boldsymbol{r}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3$

Intersection of $\mathcal{Q}$ and $\mathcal{H}$ generated by:

- generators of $\mathcal{Q}$ in $\mathcal{H}$
- combinations of green and red generators of $\mathcal{Q}$ lying on the boundary of $\mathcal{H}$



Tropical double description method (2)

Theorem (Elementary step of the DDM, Allamigeon et al. (STACS'10))

Consider:

- *a tropical polyhedron* $\mathcal{Q}$ *of generating represention* $(\{\boldsymbol{p}^i\}, \{\boldsymbol{r}^j\})$
- *a tropical halfspace* $\mathcal{H}$ *defined by* $\boldsymbol{ax} \oplus c \leq \boldsymbol{bx} \oplus d$

Then $\mathcal{Q} \cap \mathcal{H}$ *is generated by* (Q, S) *where*

Tropical double description method (2)

Theorem (Elementary step of the DDM, Allamigeon et al. (STACS'10))

Consider:

- *a tropical polyhedron* $\mathcal{Q}$ *of generating represention* $(\{\boldsymbol{p}^i\}, \{\boldsymbol{r}^j\})$
- *a tropical halfspace* $\mathcal{H}$ *defined by* $\boldsymbol{a}\boldsymbol{x} \oplus c \leq \boldsymbol{b}\boldsymbol{x} \oplus d$

Then $\mathcal{Q} \cap \mathcal{H}$ *is generated by* (Q, S) *where*

$$
\begin{aligned}
Q = {} & \left\{ \boldsymbol{p}^i \mid \boldsymbol{a}\boldsymbol{p}^i \oplus c \leq \boldsymbol{b}\boldsymbol{p}^i \oplus d \right\} \\
\cup {} & \left\{ \lambda \boldsymbol{p}^i \oplus \mu \boldsymbol{p}^j \;\middle|\; \begin{array}{l} \boldsymbol{a}\boldsymbol{p}^i \oplus c \leq \boldsymbol{b}\boldsymbol{p}^i \oplus d \text{ and } \boldsymbol{a}\boldsymbol{p}^j \oplus c > \boldsymbol{b}\boldsymbol{p}^j \oplus d \\ \lambda = \kappa^{-1}(\boldsymbol{a}\boldsymbol{p}^j \oplus c), \mu = \kappa^{-1}(\boldsymbol{b}\boldsymbol{p}^i \oplus d), \kappa = \boldsymbol{a}\boldsymbol{p}^j \oplus c \oplus \boldsymbol{b}\boldsymbol{p}^i \oplus d \end{array} \right\} \\
\cup {} & \left\{ \boldsymbol{p}^i \oplus \alpha \boldsymbol{r}^j \mid \boldsymbol{a}\boldsymbol{p}^i \oplus c \leq \boldsymbol{b}\boldsymbol{p}^i \oplus d \text{ and } \boldsymbol{a}\boldsymbol{r}^j > \boldsymbol{b}\boldsymbol{r}^j,\ \alpha = (\boldsymbol{a}\boldsymbol{r}^j)^{-1}(\boldsymbol{b}\boldsymbol{p}^i \oplus d) \right\} \\
\cup {} & \left\{ \beta \boldsymbol{r}^i \oplus \boldsymbol{p}^j \mid \boldsymbol{a}\boldsymbol{r}^i < \boldsymbol{b}\boldsymbol{r}^i \text{ and } \boldsymbol{a}\boldsymbol{p}^j \oplus c > \boldsymbol{b}\boldsymbol{p}^j \oplus d,\ \beta = (\boldsymbol{b}\boldsymbol{r}^i)^{-1}(\boldsymbol{a}\boldsymbol{p}^j \oplus c) \right\}
\end{aligned}
$$

$$
R = \left\{ \boldsymbol{r}^i \mid \boldsymbol{a}\boldsymbol{r}^i \leq \boldsymbol{b}\boldsymbol{r}^i \right\} \cup \left\{ (\boldsymbol{a}\boldsymbol{r}^j)\boldsymbol{r}^i \oplus (\boldsymbol{b}\boldsymbol{r}^i)\boldsymbol{r}^j \mid \boldsymbol{a}\boldsymbol{r}^i \leq \boldsymbol{b}\boldsymbol{r}^i \text{ and } \boldsymbol{a}\boldsymbol{r}^j > \boldsymbol{b}\boldsymbol{r}^j \right\}
$$

Tropical double description method (2): homogenized version

Theorem (Elementary step of the DDM, Allamigeon et al. (STACS'10))

Consider:

- *a tropical cone $\mathcal{C}$ of generating represention $G = (\boldsymbol{g}^i)_i$*
- *a tropical linear halfspace $\mathcal{H}$ defined by $\boldsymbol{ax} \leq \boldsymbol{bx}$*

Then $\mathcal{C} \cap \mathcal{H}$ is generated by:

$$\left\{ \boldsymbol{g}^i \mid \boldsymbol{ag}^i \leq \boldsymbol{bg}^i \right\} \cup \left\{ (\boldsymbol{ag}^j)\boldsymbol{g}^i \oplus (\boldsymbol{bg}^i)\boldsymbol{g}^j \mid \boldsymbol{ag}^i \leq \boldsymbol{bg}^i \text{ and } \boldsymbol{ag}^j > \boldsymbol{bg}^j \right\}$$

Tropical double description method (3)

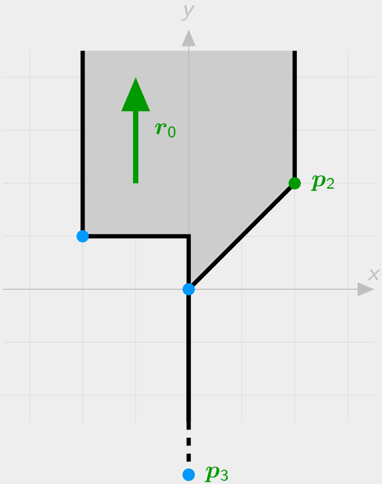
This method may yield non-extreme generators:

Definition

extreme = not a combination of the other generators

Tropical double description method (3)

This method may yield non-extreme generators:

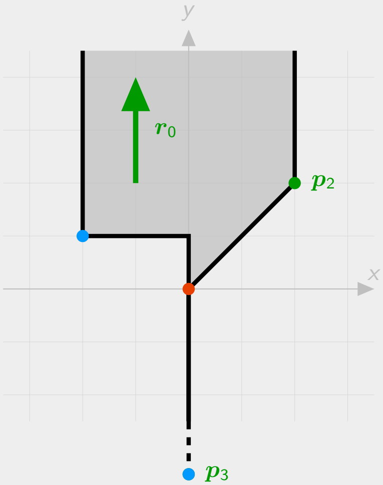


Definition

extreme = not a combination of the other generators

Tropical double description method (3)

This method may yield non-extreme generators:

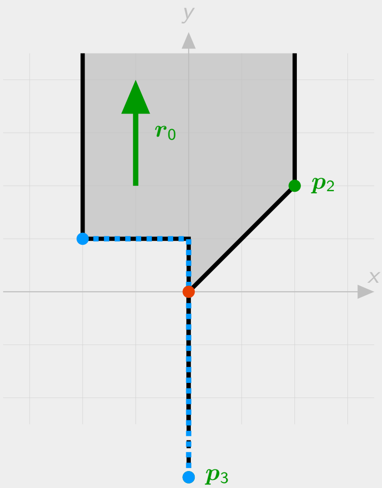


Definition

extreme = not a combination of the other generators

Tropical double description method (3)

This method may yield non-extreme generators:

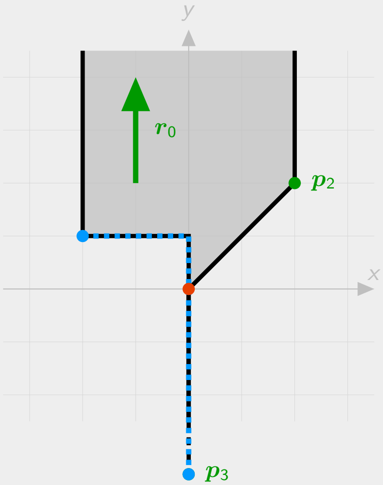


Definition

extreme = not a combination of the other generators

Tropical double description method (3)

This method may yield non-extreme generators:



Definition

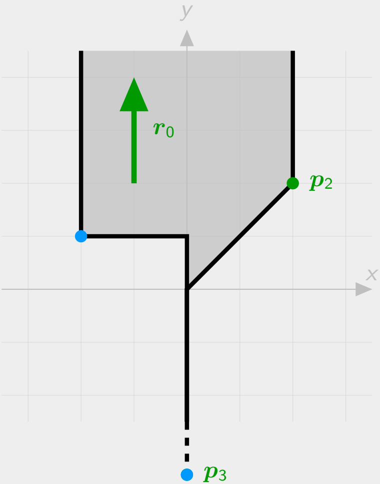
extreme = not a combination of the other generators

Non-extreme generators

- redundant and useless

Tropical double description method (3)

This method may yield non-extreme generators:



Definition

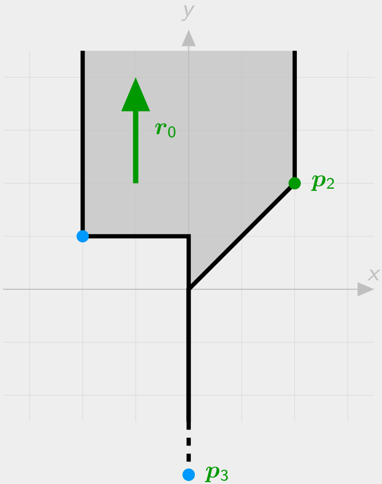
extreme = not a combination of the other generators

Non-extreme generators

- redundant and useless

Tropical double description method (3)

This method may yield non-extreme generators:



Definition

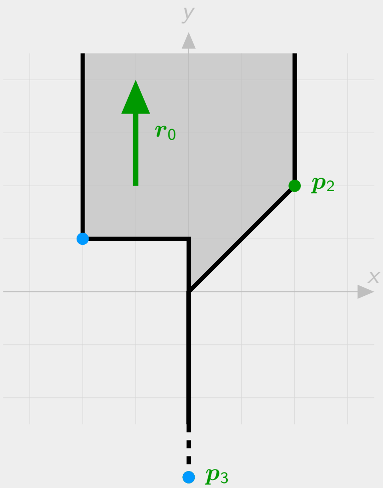
extreme = not a combination of the other generators

Non-extreme generators

- redundant and useless
- may considerably degrade the performance of the DDM
  double exponential complexity

Tropical double description method (3)

This method may yield non-extreme generators:



Definition

extreme = not a combination of the other generators

Non-extreme generators

- redundant and useless
- may considerably degrade the performance of the DDM
  double exponential complexity

Non-extreme generators must be eliminated at *each step* of the induction

Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
   - Tropical double description method
   - Combinatorial characterization of extreme points
   - From inequalities to generators
   - Some other results
3. Tropical polyhedra based numerical domains
4. Conclusion

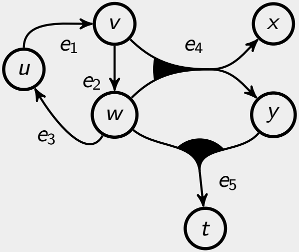
Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $Ax \oplus c \leq Bx \oplus d$ $\longrightarrow$ Reachability problem is a **directed hypergraph**

Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $\boldsymbol{A x} \oplus \boldsymbol{c} \leq \boldsymbol{B x} \oplus \boldsymbol{d}$ ⟶ Reachability problem is a **directed hypergraph**

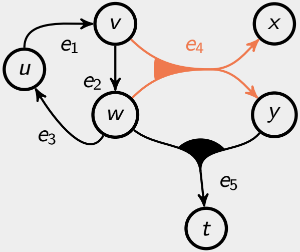
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $\boldsymbol{A x} \oplus \boldsymbol{c} \leq \boldsymbol{B x} \oplus \boldsymbol{d}$ ⟶ Reachability problem is a **directed hypergraph**

Directed hypergraphs = generalization of directed graphs:

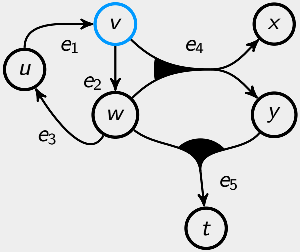


$\{ v, w \} \longrightarrow \{ x, y \}$

Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $\boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{c} \leq \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{d}$ ⟶ Reachability problem is a **directed hypergraph**

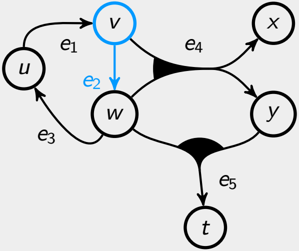
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $\boldsymbol{A x} \oplus \boldsymbol{c} \leq \boldsymbol{B x} \oplus \boldsymbol{d}$ ⟶ Reachability problem is a **directed hypergraph**

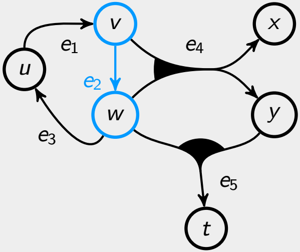
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $\boldsymbol{Ax} \oplus \boldsymbol{c} \leq \boldsymbol{Bx} \oplus \boldsymbol{d}$ ⟶ Reachability problem is a **directed hypergraph**

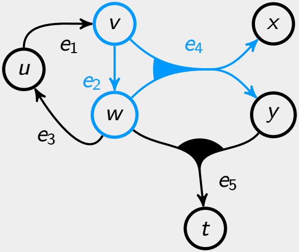
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $Ax \oplus c \leq Bx \oplus d$ ⟶ Reachability problem is a **directed hypergraph**

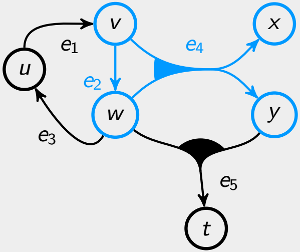
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $Ax \oplus c \leq Bx \oplus d$ ⟶ Reachability problem is a **directed hypergraph**

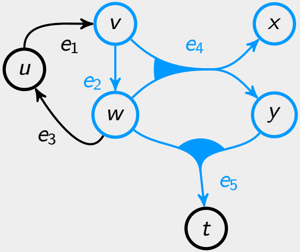
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $Ax \oplus c \leq Bx \oplus d$ ⟶ Reachability problem is a **directed hypergraph**

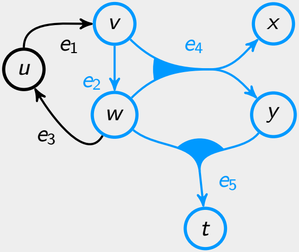
Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points

Extremality in a tropical polyhedron $\boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{c} \leq \boldsymbol{B}\boldsymbol{x} \oplus \boldsymbol{d}$ ⟶ Reachability problem is a **directed hypergraph**

Directed hypergraphs = generalization of directed graphs:



Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by $\begin{cases} A_1 \boldsymbol{x} \oplus \boldsymbol{c}_1 \leq B_1 \boldsymbol{x} \oplus \boldsymbol{d}_1 \\ \vdots \\ A_p \boldsymbol{x} \oplus \boldsymbol{c}_p \leq B_p \boldsymbol{x} \oplus \boldsymbol{d}_p \end{cases}$ **Is $\boldsymbol{p} \in \mathcal{P}$ extreme?**

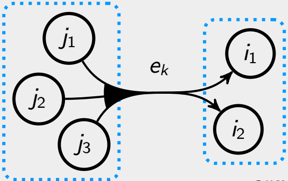
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by $\begin{cases} A_1 \boldsymbol{x} \oplus \boldsymbol{c}_1 \leq B_1 \boldsymbol{x} \oplus \boldsymbol{d}_1 \\ \vdots \\ A_p \boldsymbol{x} \oplus \boldsymbol{c}_p \leq B_p \boldsymbol{x} \oplus \boldsymbol{d}_p \end{cases}$ **Is $\boldsymbol{p} \in \mathcal{P}$ extreme?**

Definition

The *tangent directed hypergraph* $\mathcal{H}(\boldsymbol{p})$ at the point $\boldsymbol{p}$ is formed by the hyperedges

for each k such that $A_k \boldsymbol{p} \oplus \boldsymbol{c}_k = B_k \boldsymbol{p} \oplus \boldsymbol{d}_k$



$\arg\max(B_k \boldsymbol{p} \oplus \boldsymbol{d}_k)$

$\arg\max(A_k \boldsymbol{p} \oplus \boldsymbol{c}_k)$

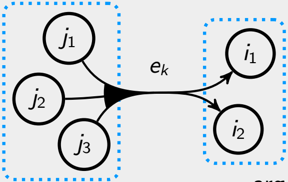
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by $\begin{cases} A_1 \boldsymbol{x} \oplus \boldsymbol{c}_1 \leq B_1 \boldsymbol{x} \oplus \boldsymbol{d}_1 \\ \vdots \\ A_p \boldsymbol{x} \oplus \boldsymbol{c}_p \leq B_p \boldsymbol{x} \oplus \boldsymbol{d}_p \end{cases}$ **Is $\boldsymbol{p} \in \mathcal{P}$ extreme?**

Definition

The *tangent directed hypergraph* $\mathcal{H}(\boldsymbol{p})$ at the point $\boldsymbol{p}$ is formed by the hyperedges

for each k such that $A_k \boldsymbol{p} \oplus \boldsymbol{c}_k = B_k \boldsymbol{p} \oplus \boldsymbol{d}_k$



$\arg\max(B_k \boldsymbol{p} \oplus \boldsymbol{d}_k)$

$\arg\max(A_k \boldsymbol{p} \oplus \boldsymbol{c}_k)$

Theorem (Allamigeon et al. (STACS'10))

$\boldsymbol{p} \in \mathcal{P}$ *is extreme* $\iff$ $\mathcal{H}(\boldsymbol{p})$ *admits a sink*

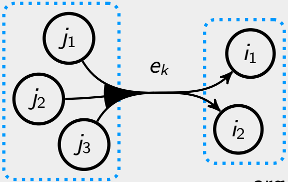
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by $\begin{cases} A_1\boldsymbol{x} \oplus \boldsymbol{c}_1 \leq B_1\boldsymbol{x} \oplus \boldsymbol{d}_1 \\ \vdots \\ A_p\boldsymbol{x} \oplus \boldsymbol{c}_p \leq B_p\boldsymbol{x} \oplus \boldsymbol{d}_p \end{cases}$ **Is $\boldsymbol{p} \in \mathcal{P}$ extreme?**

Definition

The *tangent directed hypergraph* $\mathcal{H}(\boldsymbol{p})$ at the point $\boldsymbol{p}$ is formed by the hyperedges

for each k such that $A_k\boldsymbol{p} \oplus \boldsymbol{c}_k = B_k\boldsymbol{p} \oplus \boldsymbol{d}_k$



$\arg\max(B_k\boldsymbol{p} \oplus \boldsymbol{d}_k)$

$\arg\max(A_k\boldsymbol{p} \oplus \boldsymbol{c}_k)$

Theorem (Allamigeon et al. (STACS'10))

$\boldsymbol{p} \in \mathcal{P}$ *is extreme* $\iff$ $\mathcal{H}(\boldsymbol{p})$ *admits a sink*

reachable from all nodes

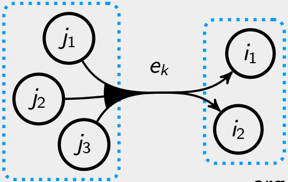
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by $\begin{cases} A_1 \boldsymbol{x} \oplus \boldsymbol{c}_1 \leq B_1 \boldsymbol{x} \oplus \boldsymbol{d}_1 \\ \vdots \\ A_p \boldsymbol{x} \oplus \boldsymbol{c}_p \leq B_p \boldsymbol{x} \oplus \boldsymbol{d}_p \end{cases}$ **Is $\boldsymbol{p} \in \mathcal{P}$ extreme?**

Definition

The *tangent directed hypergraph* $\mathcal{H}(\boldsymbol{p})$ at the point $\boldsymbol{p}$ is formed by the hyperedges

for each k such that $A_k \boldsymbol{p} \oplus \boldsymbol{c}_k = B_k \boldsymbol{p} \oplus \boldsymbol{d}_k$



$\arg\max(B_k \boldsymbol{p} \oplus \boldsymbol{d}_k)$

$\arg\max(A_k \boldsymbol{p} \oplus \boldsymbol{c}_k)$

Stronger property than in the classical case:

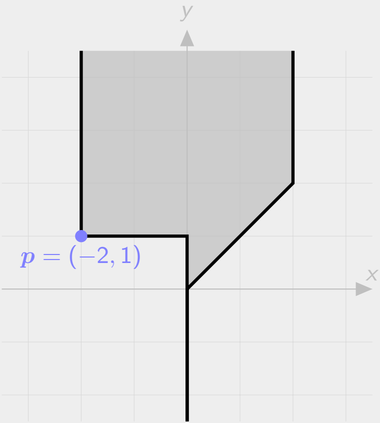
- saturated inequalities
- “maximality” criterion

Theorem (Allamigeon et al. (STACS’10))

$\boldsymbol{p} \in \mathcal{P}$ *is extreme* $\iff$ $\mathcal{H}(\boldsymbol{p})$ *admits a sink*

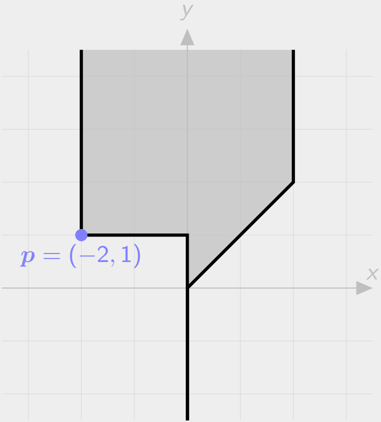
reachable from all nodes

Combinatorial characterization of extreme points (let's practice!)



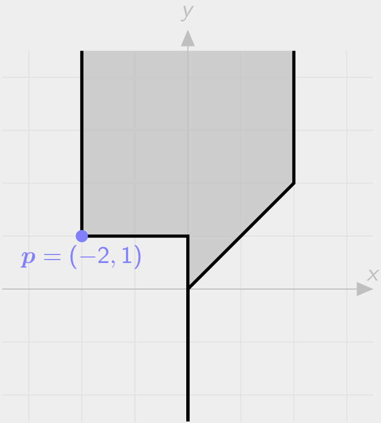
$$x \leq \max(y, 0)$$
$$0 \leq x + 2$$
$$x \leq 2$$
$$0 \leq \max(x, y - 1)$$

Combinatorial characterization of extreme points (let's practice!)



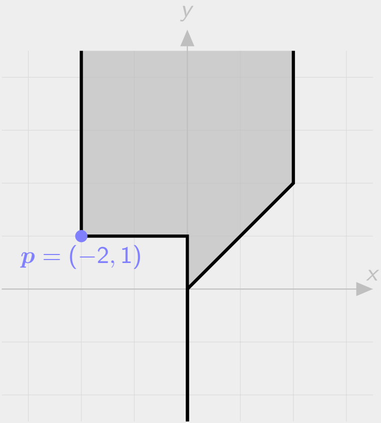
$$\begin{aligned} -2 = x &\leq \max(y, 0) = 1 \\ 0 &\leq x + 2 \\ x &\leq 2 \\ 0 &\leq \max(x, y - 1) \end{aligned}$$

Combinatorial characterization of extreme points (let's practice!)



$$\begin{aligned}
-2 = x &< \max(y, 0) = 1 \\
0 &\leq x + 2 \\
x &\leq 2 \\
0 &\leq \max(x, y - 1)
\end{aligned}$$

Combinatorial characterization of extreme points (let's practice!)



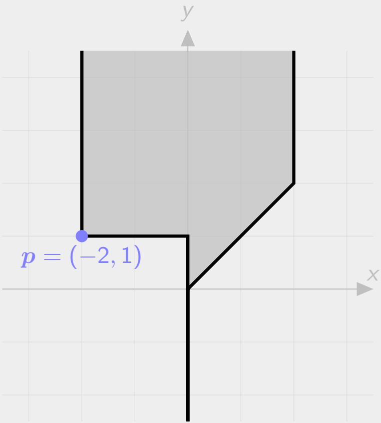
$$-2 = x < \max(y, 0) = 1$$

$$0 \leq x + 2 = 0$$

$$x \leq 2$$

$$0 \leq \max(x, y - 1)$$

Combinatorial characterization of extreme points (let's practice!)



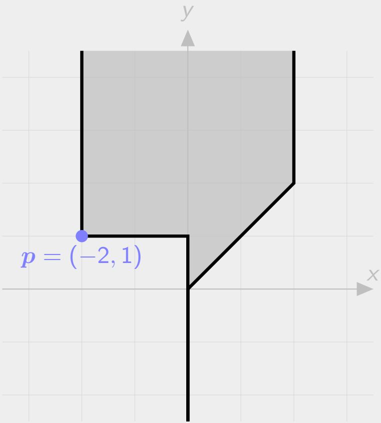
$$-2 = x < \max(y, 0) = 1$$

$$0 = x + 2 = 0$$

$$x \leq 2$$

$$0 \leq \max(x, y - 1)$$

Combinatorial characterization of extreme points (let's practice!)

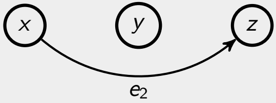


$$-2 = x < \max(y, 0) = 1$$

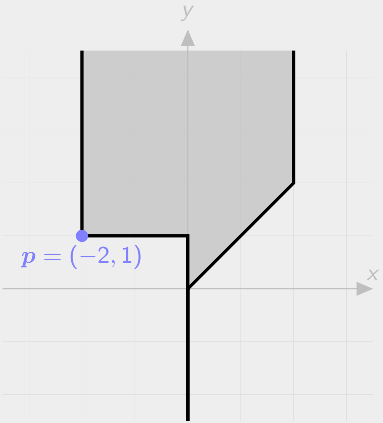
$$0 = x + 2 = 0$$

$$x \leq 2$$

$$0 \leq \max(x, y - 1)$$



Combinatorial characterization of extreme points (let's practice!)

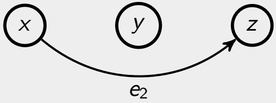


$$-2 = x < \max(y, 0) = 1$$

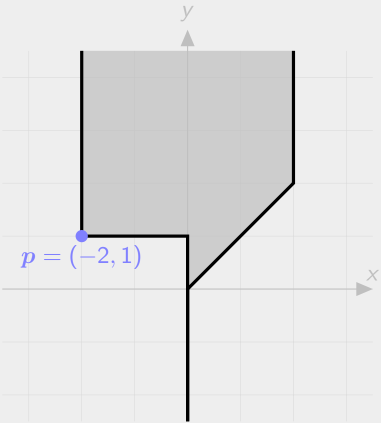
$$0 = x + 2 = 0$$

$$-2 = x \leq 2$$

$$0 \leq \max(x, y - 1)$$



Combinatorial characterization of extreme points (let's practice!)

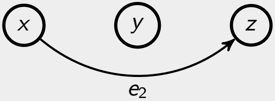


$$-2 = x < \max(y,0) = 1$$

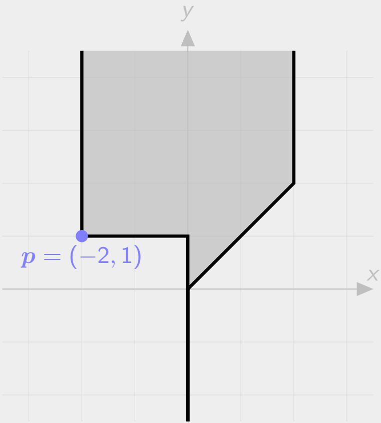
$$0 = x + 2 = 0$$

$$-2 = x < 2$$

$$0 \leq \max(x, y-1)$$



Combinatorial characterization of extreme points (let's practice!)

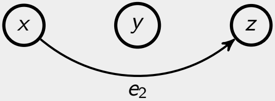


$$-2 = x < \max(y, 0) = 1$$

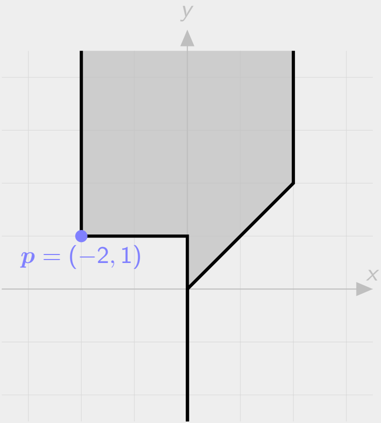
$$0 = x + 2 = 0$$

$$-2 = x < 2$$

$$0 \leq \max(x, y - 1) = 0$$



Combinatorial characterization of extreme points (let's practice!)

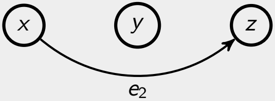


$$-2 = x < \max(y, 0) = 1$$

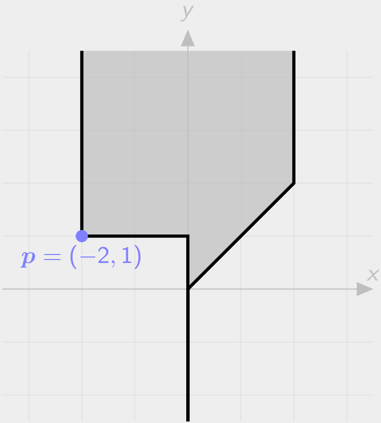
$$0 = x + 2 = 0$$

$$-2 = x < 2$$

$$0 = \max(x, y - 1) = 0$$



Combinatorial characterization of extreme points (let's practice!)

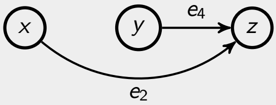


$$-2 = x < \max(y, 0) = 1$$

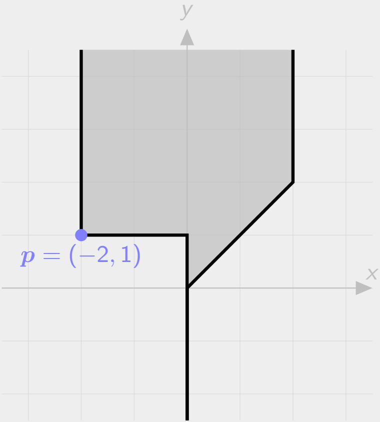
$$0 = x + 2 = 0$$

$$-2 = x < 2$$

$$0 = \max(x, y - 1) = 0$$



Combinatorial characterization of extreme points (let's practice!)

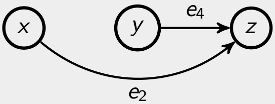


$$-2 = x < \max(y, 0) = 1$$

$$0 = x + 2 = 0$$

$$-2 = x < 2$$

$$0 = \max(x, y - 1) = 0$$



$\mathcal{H}(p)$ has a sink: z

Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest $\textsc{Scc}$

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

Efficient evaluation of the extremality criterion

$$\mathcal{H} \text{ admits a sink} \quad \Longleftrightarrow \quad \mathcal{H} \text{ has a greatest } \textsc{Scc}$$

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal SCCs *can be determined in almost linear time.*

Efficient evaluation of the extremality criterion

$$\mathcal{H} \text{ admits a sink} \quad \Longleftrightarrow \quad \mathcal{H} \text{ has a greatest } \textsc{Scc}$$

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal SCCs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal SCCs in the underlying directed graph (Tarjan)

Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest $\textsc{Scc}$

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

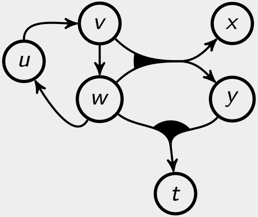
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal $\textsc{Scc}$*s can be determined in almost linear time.*

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal $\textsc{Scc}$s in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

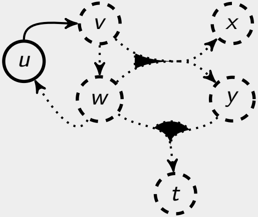
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Sccs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest SCC

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

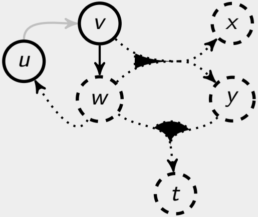
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal SCCs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal SCCs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest SCC
(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

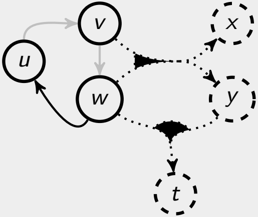
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal SCCs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal SCCs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

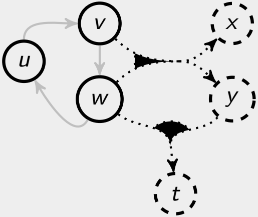
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Sccs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest $\textsc{Scc}$
(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

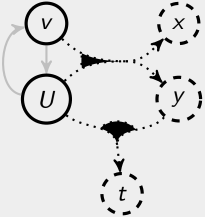
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal $\textsc{Scc}$*s can be determined in almost linear time.*

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal $\textsc{Scc}$s in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest $\textsc{Scc}$

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

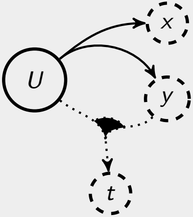
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal $\textsc{Scc}$*s can be determined in almost linear time.*

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal $\textsc{Scc}$s in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

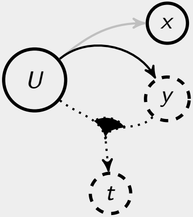
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Sccs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\iff$ $\mathcal{H}$ has a greatest $\textsc{Scc}$

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

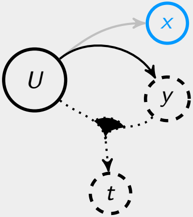
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal $\textsc{Scc}$s can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal $\textsc{Scc}$s in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

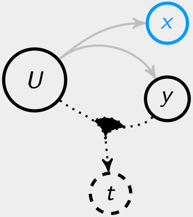
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Sccs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

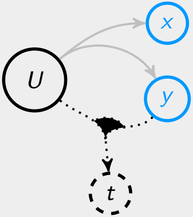
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Sccs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

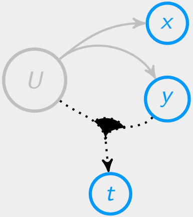
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Sccs can be determined in almost linear time.

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Efficient evaluation of the extremality criterion

$\mathcal{H}$ admits a sink $\Longleftrightarrow$ $\mathcal{H}$ has a greatest Scc

(order induced by the reachability relation: $C_1 \prec C_2$ iff C_1 reaches C_2)

directed graphs decidable in linear time (variation of Tarjan)

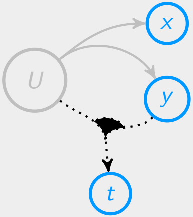
directed hypergraphs

Theorem (Allamigeon et al. (STACS'10))

The maximal Scc*s can be determined in almost linear time.*

Sequence of operations of two kinds:

- merging some nodes in the directed hypergraph
- discovering the maximal Sccs in the underlying directed graph (Tarjan)



Result of independent interest

- no previous work on Sccs in directed hypergraph
- only existing method suboptimal, based on Gallo et al. [1993]

Efficient evaluation of the extremality criterion (2)

```
 1: function HMaxSccCount($\mathcal{H} = (N, E)$)
 2:   $n := 0$, $nb := 0$, $S := []$, $Finished := \emptyset$
 3:   for all $e \in E$ do $r_e := undef$, $c_e := 0$
 4:   for all $u \in N$ do
 5:     $index[u] := undef$, $low[u] := undef$
 6:     $F_u := []$, MakeSet($u$)
 7:   done
 8:   for all $u \in N$ do
 9:     if $index[u] = undef$ then HVisit($u$)
10:   done
11:   return $nb$
12: end

13: function HVisit($u$)
14:   local $U := \text{Find}(u)$, local $F := []$
15:   $index[U] := n$, $low[U] := n$, $n := n + 1$
16:   $ismax[U] := true$, push $U$ on the stack $S$
17:   for all $e \in E_u$ do
18:     if $|T(e)| = 1$ then push $e$ on $F$
19:     else
20:       if $r_e = undef$ then $r_e := u$
21:       local $R_e := \text{Find}(r_e)$
22:       if $R_e$ appears in $S$ then
23:         $c_e := c_e + 1$
24:         if $c_e = |T(e)|$ then
25:           push $e$ on the stack $F_{R_e}$
26:         end
27:       end
28:     end
29:   done
30:   while $F$ is not empty do
31:     pop $e$ from $F$
32:     for all $w \in H(e)$ do
33:       local $W := \text{Find}(w)$
34:       if $index[W] = undef$ then HVisit($w$)
35:       if $W \in Finished$ then
36:         $ismax[U] := false$
37:       else
38:         $low[U] := \min(low[U], low[W])$
39:         $ismax[U] := ismax[U]$ && $ismax[W]$
40:       end
41:     done
42:   done
43:   if $low[U] = index[U]$ then
44:     if $ismax[U] = true$ then
45:       local $i := index[U]$
46:       pop each $e$ from $F_U$ and push it on $F$
47:       pop $V$ from $S$
48:       while $index[V] > i$ do
49:         pop each $e$ from $F_V$ and push it on $F$
50:         $U := \text{Merge}(U, V)$
51:         pop $V$ from $S$
52:       done
53:       $index[U] := i$, push $U$ on $S$
54:       if $F$ is not empty then go to Line 30
55:       $nb := nb + 1$
56:     end
57:     repeat
58:       pop $V$ from $S$, add $V$ to $Finished$
59:     until $index[V] = index[U]$
60:   end
61: end
```

Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
   - Tropical double description method
   - Combinatorial characterization of extreme points
   - From inequalities to generators
   - Some other results
3. Tropical polyhedra based numerical domains
4. Conclusion

From inequalities to generators

InegToGen = combination of

- tropical double description method
- elimination of non-extreme elements by hypergraph-based characterization

From inequalities to generators

$\textsc{IneqToGen}$ = combination of

- tropical double description method
- elimination of non-extreme elements by hypergraph-based characterization

Proposition

The time complexity of $\textsc{IneqToGen}$ *is*

$$O(p^2\ d\ G_{\max}^2)$$

where:

- d = *dimension*
- p = *nb of constraints in* $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$
- $G_{\max}$ = *maximal number of extreme generators in the intermediate polyhedra*

From inequalities to generators

INEQTOGEN = combination of

- tropical double description method
- elimination of non-extreme elements by hypergraph-based characterization

Proposition

The time complexity of INEQTOGEN *is*

$$O(p^2\ d\ G_{\mathrm{max}}^2)$$

where:

- *d = dimension*
- *p = nb of constraints in $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$*
- *G_{max} = maximal number of extreme generators in the intermediate polyhedra* **leading term**, *exponential in d*

Comparison to existing works

Time complexity of $\textsc{IneqToGen} = O(p^2 \; d \; G_{\text{max}}^2)$

Notations

- $d =$ dimension
- $p =$ nb of constraints in $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$
- $G_{\text{max}} =$ maximal number of extreme generators in the intermediate polyhedra **leading term**, exponential in d

Comparison to existing works

Time complexity of $\textsc{IneqToGen} = O(p^2\ d\ G_{\text{max}}^2)$

Tropical world:

- seminal algorithm due to Butkovič and Hegedüs [1984]: double exponential

Notations

- d = dimension
- p = nb of constraints in $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$
- G_{max} = maximal number of extreme generators in the intermediate polyhedra **leading term**, exponential in d

Comparison to existing works

Time complexity of InEqToGen $= O(p^2\ d\ G_{\max}^2)$

Tropical world:

- seminal algorithm due to Butkovič and Hegedüs [1984]: double exponential
- implementation in the Max-plus toolbox of Scilab and ScicosLab, later refined in Allamigeon et al. (SAS'08)

  $$O(p\ d\ G_{\max}^4)$$

  Elimination of non-extreme elements by residuation [see Vorobyev, 1967, Cuninghame-Green, 1976]

Notations

- $d =$ dimension
- $p =$ nb of constraints in $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$
- $G_{\max} =$ maximal number of extreme generators in the intermediate polyhedra **leading term**, exponential in d

Comparison to existing works

Time complexity of $\textsc{IneqToGen} = O(p^2\ d\ G_{\max}^2)$

Tropical world:

- seminal algorithm due to Butkovič and Hegedüs [1984]: double exponential
- implementation in the Max-plus toolbox of Scilab and ScicosLab, later refined in Allamigeon et al. (SAS'08)

$$O(p\ d\ G_{\max}^4)$$

Elimination of non-extreme elements by residuation [see Vorobyev, 1967, Cuninghame-Green, 1976]

Classical world: Motzkin et al. [1953], Fukuda and Prodon [1996]

$$O(p^2\ G_{\max}^3)$$

Notations

- d = dimension
- p = nb of constraints in $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{x} \oplus \boldsymbol{d}$
- $G_{\max}$ = maximal number of extreme generators in the intermediate polyhedra **leading term**, exponential in d

IneqToGen: benchmarks

Implementation in OCaml, experimentations on a 3 GHz Intel Xeon with 3 Gb RAM

	d	p	# final	# inter.	T (s)	T' (s)
rnd100	12	15	32	59	0.24	6.72
rnd100	15	10	555	292	2.87	321.78
rnd100	15	18	152	211	6.26	899.21
rnd30	17	10	1484	627	15.2	4667.9
rnd10	20	8	5153	1273	49.8	50941.9
rnd10	25	5	3999	808	9.9	12177.0
rnd10	25	10	32699	6670	3015.7	—
cyclic	10	20	3296	887	25.8	4957.1
cyclic	15	7	2640	740	8.1	1672.2
cyclic	17	8	4895	1589	44.8	25861.1
cyclic	20	8	28028	5101	690	$\sim$ 45 days
cyclic	25	5	25025	1983	62.6	$\sim$ 8 days
cyclic	30	5	61880	3804	261	—
cyclic	35	5	155040	7695	1232.6	—

- T: IneqToGen
- T': previous algorithm of Scilab and Allamigeon et al. (SAS'08)

Contents

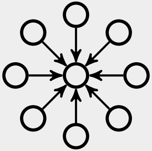
1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
   - Tropical double description method
   - Combinatorial characterization of extreme points
   - From inequalities to generators
   - Some other results
3. Tropical polyhedra based numerical domains
4. Conclusion

From generators to inequalities

GenToIneq =

- dual version of the double description method
- elimination of “non-extreme inequalities” at each step of the induction

Characterizing extreme inequalities is easier:



Maximal number of extreme elements in tropical polyhedron

Theorem (McMullen-type bound, Allamigeon et al. (submitted to JCTA))

The number of extreme elements of a tropical polyhedron in $\mathbb{R}^d_{\max}$ defined by p inequalities is bounded by

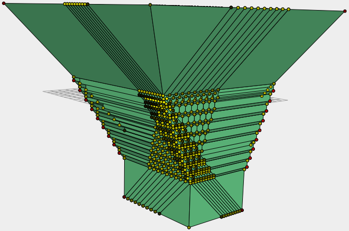
$$U(p+d+1,d) = O\left((p+d+1)^{\lfloor d/2 \rfloor}\right)$$

Candidates to be maximizing instances: *signed cyclic polyhedral cones*

Theorem (Allamigeon et al. (submitted to JCTA)

- *the bound $U(p+d+1,d)$ is tight when $d \to +\infty$ and p fixed*
- *when $p \geq 2d$, lower bound in*

$$O((p-2d)2^{d-2})$$



Upper bound on the complexity of our algorithms

- from inequalities to generators:

$$\begin{cases} O(p^2 d(p+d+1)^{d-1}) & \text{if } d \text{ is odd} \\ O(p^2 d(p+d+1)^{d}) & \text{if } d \text{ is even} \end{cases}$$

- from generators to inequalities:

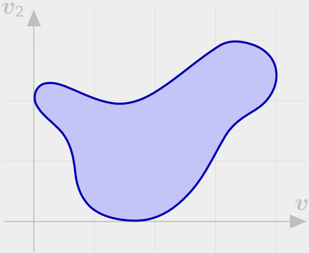
$$\begin{cases} O(pd^2 (p+d)^{d-1}) & \text{if } d \text{ is even} \\ O(pd^2 (p+d)^{d}) & \text{if } d \text{ is odd} \end{cases}$$

Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
3. Tropical polyhedra based numerical domains
   - Inferring max-invariants: the abstract domain MaxPoly
   - Other tropical polyhedra based abstract domains
   - Experiments
4. Conclusion

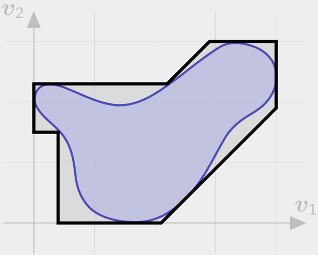
Principle of the abstract domain MaxPoly

Over-approximates subsets of $\mathbb{R}^d$ by means of tropical polyhedra:



Principle of the abstract domain MaxPoly

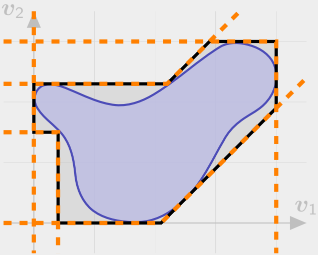
Over-approximates subsets of $\mathbb{R}^d$ by means of tropical polyhedra:



Double representation:

Principle of the abstract domain MaxPoly

Over-approximates subsets of $\mathbb{R}^d$ by means of tropical polyhedra:

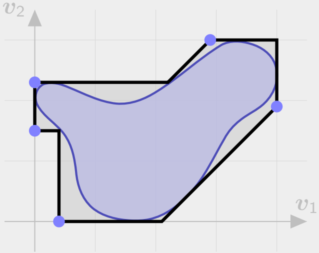


Double representation:

- by inequalities $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{d} \oplus \boldsymbol{d}$

Principle of the abstract domain MaxPoly

Over-approximates subsets of $\mathbb{R}^d$ by means of tropical polyhedra:

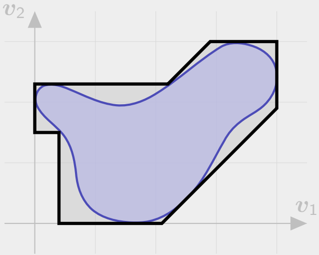


Double representation:

- by inequalities $Ax \oplus c \leq Bd \oplus d$
- by generators (P, R)

Principle of the abstract domain MaxPoly

Over-approximates subsets of $\mathbb{R}^d$ by means of tropical polyhedra:



Double representation:

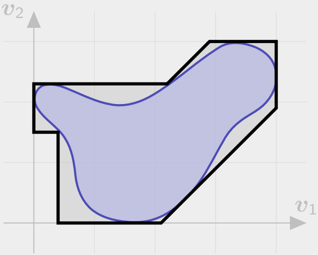
- by inequalities $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{d} \oplus \boldsymbol{d}$
- by generators (P, R)

Expressivity: conjunction of max-invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$

$$\max(\alpha_0, \alpha_1 + \boldsymbol{v}_1, \dots, \alpha_d + \boldsymbol{v}_d) \leq \max(\beta_0, \beta_1 + \boldsymbol{v}_1, \dots, \beta_d + \boldsymbol{v}_d)$$

Principle of the abstract domain MaxPoly

Over-approximates subsets of $\mathbb{R}^d$ by means of tropical polyhedra:



Double representation:

- by inequalities $A\boldsymbol{x} \oplus \boldsymbol{c} \leq B\boldsymbol{d} \oplus \boldsymbol{d}$
- by generators (P, R)

Expressivity: conjunction of max-invariants over variables $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$

$$\max(\alpha_0, \alpha_1 + \boldsymbol{v}_1, \dots, \alpha_d + \boldsymbol{v}_d) \leq \max(\beta_0, \beta_1 + \boldsymbol{v}_1, \dots, \beta_d + \boldsymbol{v}_d)$$

$\Longrightarrow$ connected disjunctions of zone invariants $\boldsymbol{v}_i - \boldsymbol{v}_j \geq \kappa$:

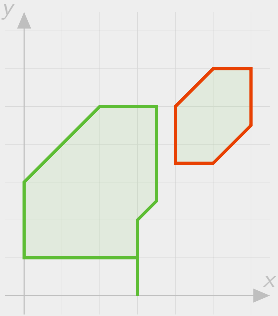
$$\bigvee_{\substack{1 \leq i \leq d \\ \beta_i \neq -\infty}} \left[\left(\bigwedge_{1 \leq j \leq d} \alpha_j - \beta_i \leq \boldsymbol{v}_i - \boldsymbol{v}_j \right) \wedge (\alpha_0 - \beta_i \leq \boldsymbol{v}_i) \right] \vee \left[\bigwedge_{\substack{1 \leq i \leq d \\ \alpha_i \neq -\infty}} \boldsymbol{v}_i \leq \beta_0 - \alpha_i \right]$$

Some abstract primitives

Abstract primitives generally use one of the representations:
$\Longrightarrow$ IneqToGen and GenToIneq are critical

- Abstract union: given two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, and (P, R) and (Q, S) their generating representations,

$$\mathcal{P} \sqcup \mathcal{Q} \overset{def}{=} \text{polyhedron generated by } (P \cup Q, R \cup S)$$

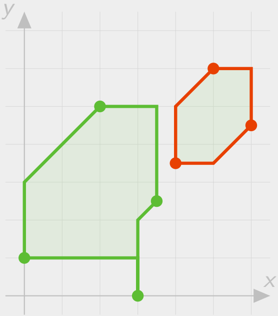


Some abstract primitives

Abstract primitives generally use one of the representations:
$\Longrightarrow$ IneqToGen and GenToIneq are critical

- Abstract union: given two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, and (P, R) and (Q, S) their generating representations,

$$\mathcal{P} \sqcup \mathcal{Q} \overset{def}{=} \text{polyhedron generated by } (P \cup Q, R \cup S)$$

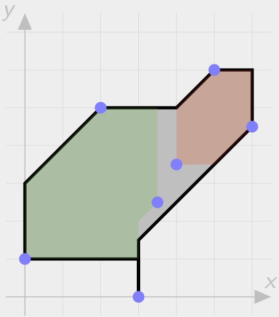


Some abstract primitives

Abstract primitives generally use one of the representations:
$\Longrightarrow$ IneqToGen and GenToIneq are critical

- Abstract union: given two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, and (P, R) and (Q, S) their generating representations,

$$\mathcal{P} \sqcup \mathcal{Q} \overset{def}{=} \text{polyhedron generated by } (P \cup Q, R \cup S)$$

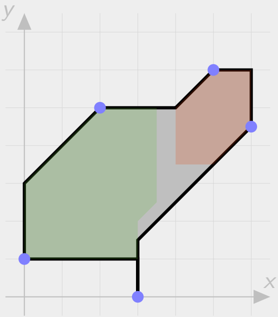


Some abstract primitives

Abstract primitives generally use one of the representations:
$\Longrightarrow$ IneqToGen and GenToIneq are critical

- Abstract union: given two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, and (P, R) and (Q, S) their generating representations,

$$\mathcal{P} \sqcup \mathcal{Q} \overset{def}{=} \text{polyhedron generated by } (P \cup Q, R \cup S)$$



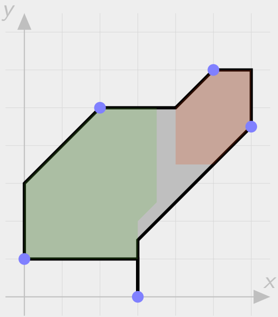
Non-extreme generators can be eliminated in polynomial time

Some abstract primitives

Abstract primitives generally use one of the representations:
$\Longrightarrow$ IneqToGen and GenToIneq are critical

- Abstract union: given two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, and (P, R) and (Q, S) their generating representations,

$$\mathcal{P} \sqcup \mathcal{Q} \overset{def}{=} \text{polyhedron generated by } (P \cup Q, R \cup S)$$



- sound: $\mathcal{P} \cup \mathcal{Q} \subset \mathcal{P} \sqcup \mathcal{Q}$
- as precise as possible: for all $\mathcal{P}, \mathcal{Q} \subset \mathcal{R}$,

$$\mathcal{P} \sqcup \mathcal{Q} \subset \mathcal{R}$$

Non-extreme generators can be eliminated in polynomial time

Some abstract primitives (2)

- abstract intersection, assignments, . . . all sound and exact

Some abstract primitives (2)

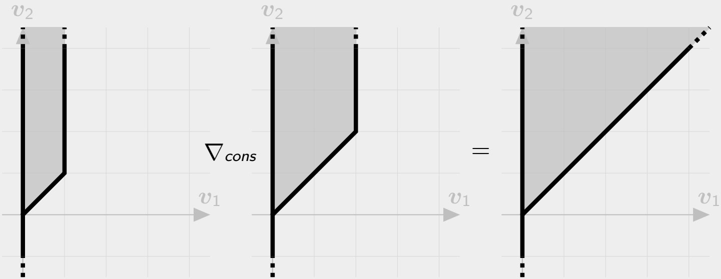
- abstract intersection, assignments, . . . all sound and exact
- widening operators to enforce convergence:
  if $\mathcal{P}_0 \subset \cdots \subset \mathcal{P}_n \subset \cdots$, the sequence defined by

$$\begin{cases} \mathcal{Q}_0 \overset{def}{=} \mathcal{P}_0 \\ \mathcal{Q}_{n+1} \overset{def}{=} \mathcal{Q}_n \nabla \mathcal{P}_{n+1} \end{cases}$$

  eventually stabilizes.

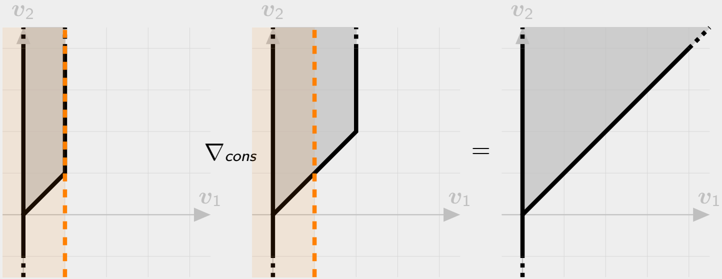
Some abstract primitives (2)

- abstract intersection, assignments, . . . all sound and exact
- widening operators to enforce convergence:
    - ∇_{cons}: eliminate non-stable inequalities



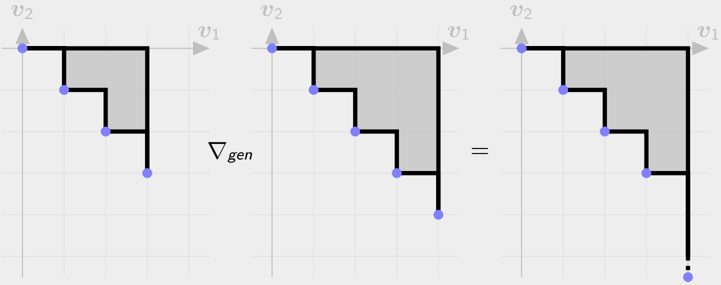
Some abstract primitives (2)

- abstract intersection, assignments, . . . all sound and exact
- widening operators to enforce convergence:
  - ∇_{cons}: eliminate non-stable inequalities



Some abstract primitives (2)

- abstract intersection, assignments, . . . all sound and exact
- widening operators to enforce convergence:
    - ∇_{cons}: eliminate non-stable inequalities
    - ∇_{gen}: extrapolation of generators (using projection)



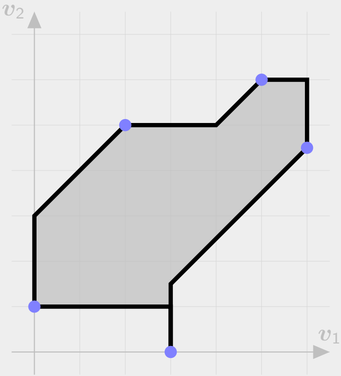
Comparison with the abstract domain of zones

- zones are tropical polyhedra with at most $(d+1)$ generators
- MaxPoly is strictly more precise that the domain of zones

Comparison with the abstract domain of zones

- zones are tropical polyhedra with at most $(d+1)$ generators
- MaxPoly is strictly more precise that the domain of zones

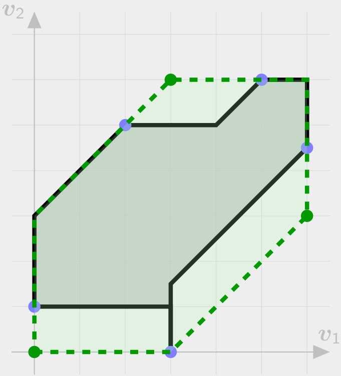
toZone($\mathcal{P}$) $=$ extract the smallest zone abstract element containing $\mathcal{P}$



Comparison with the abstract domain of zones

- zones are tropical polyhedra with at most $(d+1)$ generators
- MaxPoly is strictly more precise that the domain of zones

toZone($\mathcal{P}$) = extract the smallest zone abstract element containing $\mathcal{P}$



Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
3. Tropical polyhedra based numerical domains
   - Inferring max-invariants: the abstract domain MaxPoly
   - Other tropical polyhedra based abstract domains
   - Experiments
4. Conclusion

Other tropical polyhedra based abstract domains

- MinPoly: infer min-invariants

$$\min(\alpha_0, \alpha_1 + \boldsymbol{v}_1, \dots, \alpha_d + \boldsymbol{v}_d) \leq \min(\beta_0, \beta_1 + \boldsymbol{v}_1, \dots, \beta_d + \boldsymbol{v}_d)$$

using MaxPoly on special variables $\boldsymbol{w}_i =$ "$- \boldsymbol{v}_i$".

Other tropical polyhedra based abstract domains

- MinPoly: infer min-invariants

$$\max(-\alpha_0, -\alpha_1 + \boldsymbol{w}_1, \dots, -\alpha_d + \boldsymbol{w}_d) \geq \max(-\beta_0, -\beta_1 + \boldsymbol{w}_1, \dots, -\beta_d + \boldsymbol{w}_d)$$

using MaxPoly on special variables $\boldsymbol{w}_i =$ "$-\boldsymbol{v}_i$".

Other tropical polyhedra based abstract domains

- MinPoly: infer min-invariants

$$\min(\alpha_0, \alpha_1 + \boldsymbol{v}_1, \dots, \alpha_d + \boldsymbol{v}_d) \leq \min(\beta_0, \beta_1 + \boldsymbol{v}_1, \dots, \beta_d + \boldsymbol{v}_d)$$

  using MaxPoly on special variables $\boldsymbol{w}_i =$ “$-\boldsymbol{v}_i$”.

- MinMaxPoly: infer a superclass of min- and max-invariants

$$\begin{aligned}&\max(\alpha_0, \alpha_1 + \boldsymbol{v}_1, \dots, \alpha_d + \boldsymbol{v}_d, \alpha_{d+1} - \boldsymbol{v}_1, \dots, \alpha_{2d} - \boldsymbol{v}_d)\\ &\qquad \leq \max(\beta_0, \beta_1 + \boldsymbol{v}_1, \dots, \beta_d + \boldsymbol{v}_d, \beta_{d+1} - \boldsymbol{v}_1, \dots, \beta_{2d} - \boldsymbol{v}_d)\end{aligned}$$

  using MaxPoly on the $\boldsymbol{v}_i$ and $\boldsymbol{w}_i =$ “$-\boldsymbol{v}_i$”.

Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
3. Tropical polyhedra based numerical domains
   - Inferring max-invariants: the abstract domain MaxPoly
   - Other tropical polyhedra based abstract domains
   - Experiments
4. Conclusion

Memory manipulating programs

- widespread library functions
  - `memcpy`(dst, src, n)

```
$i := 0;$
while $i \leq n - 1$ do
    $dst[i] := src[i];$
    $i := i + 1;$
done;
```

Memory manipulating programs

- widespread library functions
  - `memcpy`(dst, src, n)

```
$i := 0;$
while $i \leq n - 1$ do
    $dst[i] := src[i];$
    $i := i + 1;$
done;
```

  - `strncpy`(dst, src, n)

> *The* `strncpy` *function copies not more than n characters (characters that follow a null character are not copied) from the array src to the array dst.*
>
> *...*
>
> *If the array src stores a string that is shorter than n characters, null characters are appended to the copy in the array dst, until n characters in all are written.*

Memory manipulating programs

- widespread library functions
  - `memcpy`(dst, src, n)

```
$i := 0$;
while $i \leq n - 1$ do
    $dst[i] := src[i]$;
    $i := i + 1$;
done;
```

  - `strncpy`(dst, src, n)

> *The* `strncpy` *function copies not more than n characters (characters that follow a null character are not copied) from the array src to the array dst.*
>
> . . .
>
> *If the array src stores a string that is shorter than n characters, null characters are appended to the copy in the array dst, until n characters in all are written.*

$$\min(len_{dst}, n) = \min(len_{src}, n)$$

Memory manipulating programs (2)

- programs embedding memory manipulation primitives

```
assume $(\mathtt{n} \geq 1)$;
$\mathtt{s} := \text{malloc}(\mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{i} \leq \mathtt{n} - 2$ do
    $\mathtt{s}[\mathtt{i}] := \text{read}()$;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
$\mathtt{s}[\mathtt{i}] := \backslash 0$;
$\mathtt{upper} := \text{malloc}(\mathtt{n})$;
$\mathtt{memcpy}(\mathtt{upper}, \mathtt{s}, \mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{upper}[\mathtt{i}] \neq \backslash 0$ do
    $\mathtt{c} := \mathtt{upper}[\mathtt{i}]$;
    if $(\mathtt{c} \geq 97) \wedge (\mathtt{c} \leq 122)$ then
        $\mathtt{upper}[\mathtt{i}] := \mathtt{c} - 32$;
    end;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
```

iterates up to the first \0

Memory manipulating programs (2)

- programs embedding memory manipulation primitives

```
assume $(\mathtt{n} \geq 1)$;
$\mathtt{s} := \mathrm{malloc}(\mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{i} \leq \mathtt{n} - 2$ do
    $\mathtt{s}[\mathtt{i}] := \mathrm{read}()$;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
$\mathtt{s}[\mathtt{i}] := \backslash 0$;
$\mathtt{upper} := \mathrm{malloc}(\mathtt{n})$;
$\mathtt{memcpy}(\mathtt{upper}, \mathtt{s}, \mathtt{n})$;
$\mathtt{i} := 0$;
while $\mathtt{upper}[\mathtt{i}] \neq \backslash 0$ do
    $\mathtt{c} := \mathtt{upper}[\mathtt{i}]$;
    if $(\mathtt{c} \geq 97) \wedge (\mathtt{c} \leq 122)$ then
        $\mathtt{upper}[\mathtt{i}] := \mathtt{c} - 32$;
    end;
    $\mathtt{i} := \mathtt{i} + 1$;
done;
```

iterates up to the first \0

- convex polyhedra: raise a false alarm

$$\mathit{len}_{\mathtt{upper}} \leq \mathit{sz}_{\mathtt{upper}}$$

Memory manipulating programs (2)

- programs embedding memory manipulation primitives

```
assume ($n \geq 1$);
$s := \mathrm{malloc}(n)$;
$i := 0$;
while $i \leq n - 2$ do
    $s[i] := \mathrm{read}()$;
    $i := i + 1$;
done;
$s[i] := \backslash 0$;
$upper := \mathrm{malloc}(n)$;
$\mathtt{memcpy}(upper, s, n)$;
$i := 0$;
while $upper[i] \neq \backslash 0$ do
    $c := upper[i]$;
    if $(c \geq 97) \wedge (c \leq 122)$ then
        $upper[i] := c - 32$;
    end;
    $i := i + 1$;
done;
```

iterates up to the first \0

- convex polyhedra: raise a false alarm

$$\mathit{len}_{\mathtt{upper}} \leq \mathit{sz}_{\mathtt{upper}}$$

- MinPoly: no buffer overflow

$$\mathit{len}_{\mathtt{upper}} < \mathit{sz}_{\mathtt{upper}}$$

Disjunctive invariants

- class of programs derived from predicate abstraction

```
1 :      $i := p_1$;
2 :      while $i \leq p_2 - 1$ do
3 :          $i := i + 1$;
4 :      done;
5 :      while $i \leq p_3 - 1$ do
6 :          $i := i + 1$;
7 :      done;
             ⋮
3n − 1 : while $i \leq p_n - 1$ do
3n − 2 :     $i := i + 1$;
3n − 3 : done;
```

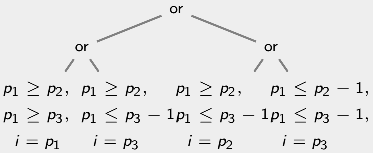
$$\boxed{i = \max(p_1, \ldots, p_n)}$$

tropical polyhedra:

- linear growth of the representation
- scales up to large values of n ($n = 60 \rightarrow 19$ s)

classical disjunctive techniques:

- exponential growth of the representation



or

or or

$p_1 \geq p_2$, $p_1 \geq p_2$, $p_1 \geq p_2$, $p_1 \leq p_2 - 1$,

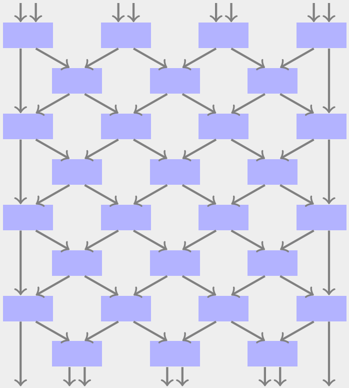
$p_1 \geq p_3$, $p_1 \leq p_3 - 1$ $p_1 \leq p_3 - 1$ $p_1 \leq p_3 - 1$,

$i = p_1$ $i = p_3$ $i = p_2$ $i = p_3$

- not practical for large values of n ($n = 60 \rightarrow 10^5$ terabytes)

Disjunctive invariants (2)

Analysis of sort algorithms:



leftmost elt = min of the initial elements

rightmost elt = max of the initial elements

Analysis for 10 elements:

- 1979.7 s with tropical polyhedra
- not practical with existing disjunctive techniques (2^{45} disjunction)

Benchmarks

Experimentations on a 3 GHz Intel Xeon with 3 Gb RAM

Program	# line	# var.	time (s) (new algo)	time (s) [Allamigeon et al., 2008]
`memcpy`	16	8	0.024	2.87
`strncpy`	20	8	0.024	2.82
incrementing-10	34	12	0.064	27.3
incrementing-11	37	13	0.088	49.64
incrementing-12	40	14	0.108	77.12
incrementing-13	43	15	0.136	130.65
incrementing-14	46	16	0.158	158.28
incrementing-15	49	17	0.210	245.32
incrementing-20	64	22	0.5	1289.29
incrementing-25	79	27	1.0	5258.55
incrementing-30	94	32	1.7	15692.9
incrementing-40	124	42	4.7	1 day
incrementing-45	139	47	7.0	> 2 days
incrementing-60	184	62	19.0	—
oddeven-4	39	9	0.012 + 0.016	0.028 + 79.51
oddeven-5	70	11	0.10 + 0.064	0.47 + —
oddeven-6	86	13	0.52 + 0.57	3.08 + —
oddeven-7	102	15	4.05 + 4.48	59.55 + —
oddeven-8	118	17	21.90 + 31.6	437.17 + —
oddeven-9	214	19	202.2 + 254.38	8240.65 + —
oddeven-10	240	19	1979.7 + 2591.0	81050.27 + —

Contents

1. A better insight into tropical polyhedra

2. Algorithmics of tropical polyhedra

3. Tropical polyhedra based numerical domains

4. Conclusion
   - Contributions of this thesis
   - Perspectives

Contributions of this thesis

Advances in combinatorics and algorithmics of tropical polyhedra in Allamigeon et al. (STACS'10), and Allamigeon et al. (submitted to JCTA)

- two conversion algorithms inequalities ⟷ generators which improve the state of the art by several orders of magnitude
- new combinatorial characterization of extreme elements from inequalities
- almost linear time algorithm to determine the maximal Sccs in directed hypergraphs
- new results on the maximal number of extreme elements in tropical polyhedra

Tropical polyhedra based abstract domains in Allamigeon et al. (SAS'08)

- infer min- and/or max-invariants
- successfully show the correctness of memory manipulating programs
- scale up to highly disjunctive invariants

Contents

1. A better insight into tropical polyhedra
2. Algorithmics of tropical polyhedra
3. Tropical polyhedra based numerical domains
4. Conclusion
   - Contributions of this thesis
   - Perspectives

Perspectives

Algorthmics of tropical polyhedra

- output-sensitive algorithm for inequalities $\longleftrightarrow$ generators
- tropical linear programming, [see Cuninghame-Green and Butkovic, 2003]
  - how to find a point in a tropical polyhedron in polynomial time? NP $\cap$ coNP [see Bezem et al., 2008, Akian et al., 2009]
- faces of tropical polyhedra [see Joswig, 2005, Develin and Yu, 2007]
- tropical upper bound on the nb of extreme elements

Abstract interpretation

- improving precision: mixing tropical and classical linear invariants

$$\max(\alpha_0, \alpha_1 + f_1, \dots, \alpha_p + f_p) \leq \max(\beta_0, \beta_1 + f'_1, \dots, \beta_p + f'_q)$$

  with f_i, f'_j classical linear forms over $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$
- improving scalability: towards subpolyhedral domains
- application to further static analyses

Thanks!

M. Akian, S. Gaubert, and A. Guterman. Tropical polyhedra are equivalent to mean payoff games. preprint, 2009.

X. Allamigeon, S. Gaubert, and É. Goubault. Inferring min and max invariants using max-plus polyhedra. In *SAS'08*, volume 5079 of *LNCS*, pages 189–204. Springer, Valencia, Spain, 2008.

X. Allamigeon, S. Gaubert, and É. Goubault. The tropical double description method. In *STACS'08*. 2010. To appear.

Xavier Allamigeon, Stéphane Gaubert, and Ricardo D. Katz. The Number of Extreme Points of Tropical Polyhedra. arXiv:math/0906.3492, 2009.

R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. *Software Tools for Technology Transfer*, 8(4/5):449–466, 2006.

Marc Bezem, Robert Nieuwenhuis, and Enric Rodríguez-Carbonell. The max-atom problem and its relevance. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008*, volume 5330 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2008.

W. Briec and C. Horvath. $\mathbb{B}$-convexity. *Optimization*, 53:103–127, 2004.

W. Briec, C. Horvath, and A. Rubinov. Separation in $\mathbb{B}$-convexity. *Pacific Journal of Optimization*, 1:13–30, 2005.

P. Butkovič and G. Hegedüs. An elimination method for finding all solutions of the system of linear equations over an extremal algebra. *Ekonomicko-matematicky Obzor*, 20, 1984.

G. Cohen, S. Gaubert, and J.P. Quadrat. Hahn-Banach separation theorem for max-plus semimodules. In J.L. Menaldi, E. Rofman, and A. Sulem, editors, *Optimal Control and Partial Differential Equations*, pages 325–334. IOS Press, 2001.

G. Cohen, S. Gaubert, and J. P. Quadrat. Duality and separation theorem in idempotent semimodules. *Linear Algebra and Appl.*, 379:395–422, 2004. doi: 10.1016/j.laa.2003.08.010.

P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.

P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY.

P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, NY.

R. A. Cuninghame-Green. *Minimax algebra*, volume 166 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1979. ISBN 3-540-09113-0.

R. A. Cuninghame-Green. Projections in minimax algebra. *Mathematical Programming*, 10(1):111–123, December 1976.

R. A. Cuninghame-Green and P. Butkovic. The equation a &#x2297; x = b &#x2297; y over (max, +). *Theor. Comput. Sci.*, 293(1):3–12, 2003. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(02)00228-1.

M. Develin and B. Sturmfels. Tropical convexity. *Doc. Math.*, 9:1–27 (electronic), 2004. ISSN 1431-0635.

M. Develin and J. Yu. Tropical polytopes and cellular resolutions. *Experimental Mathematics*, 16(3):277–292, 2007.

Komei Fukuda and Alain Prodon. Double description method revisited. In *Selected papers from the 8th Franco-Japanese and 4th Franco-Chinese Conference on Combinatorics and Computer Science*, pages 91–111, London, UK, 1996. Springer-Verlag. ISBN 3-540-61576-8.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete Appl. Math.*, 42(2-3):177–201, 1993. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/0166-218X(93)90045-P.

S. Gaubert and R. Katz. The Minkowski theorem for max-plus convex sets. *Linear Algebra and Appl.*, 421:356–369, 2007. URL http://www.arxiv.org/abs/math.GM/0605078.

S. Gaubert and R. Katz. Max-plus convex geometry. In R. A. Schmidt, editor, *RelMiCS/AKA 2006*, volume 4136 of *Lecture Notes in Comput. Sci.*, pages 192–206. Springer, 2006.

S. Gaubert and R. Katz. The tropical analogue of polar cones. *Linear Algebra and Appl.*, 431:608–625, 2009. URL `http://arxiv.org/abs/0805.3688`.

R. Giacobazzi and F. Ranzato. Optimal domains for disjunctive abstract interpretation. *Science of Computer Programming*, 32(1–3):177–210, 1998.

M. Joswig. Tropical halfspaces. In *Combinatorial and computational geometry*, volume 52 of *Math. Sci. Res. Inst. Publ.*, pages 409–431. Cambridge Univ. Press, Cambridge, 2005.

Laurent Mauborgne and Xavier Rival. Trace partitioning in abstract interpretation based static analyzers. In M. Sagiv, editor, *European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 5–20. Springer-Verlag, 2005.

A. Miné. A new numerical abstract domain based on difference-bound matrices. In *PADO II*, volume 2053 of *LNCS*, pages 155–172. Springer-Verlag, May 2001. `http://www.di.ens.fr/~mine/publi/article-mine-padoII.pdf`.

T.S. Motzkin, H. Raiffa, G.L. Thompson, and R.M. Thrall. The double description method. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games*, volume II, pages 51–73, 1953.

V. Nitica and I. Singer. Max-plus convex sets and max-plus semispaces. I. *Optimization*, 56(1–2):171–205, 2007.

Xavier Rival and Laurent Mauborgne. The trace partitioning abstract domain. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29 (5), 2007.

N.N. Vorobyev. Extremal algebra of positive matrices. *Elektron. Informationsverarbeitung und Kybernetik*, 3:39–71, 1967. in Russian.

K. Zimmermann. A general separation theorem in extremal algebras. *Ekonom.-Mat. Obzor*, 13(2):179–201, 1977.