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

→ 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

⇒ 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

⇒ 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

⇒ 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
  
  \( \implies \text{can not miss any bug} \)

• if not precise enough, it is not able to show the absence of bugs
  
  \( \implies \text{false alarm} \)
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: \textit{index of the first '0' character}

length $= 7$
Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $v_1, \ldots, v_d$. For instance:
Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $v_1, \ldots, v_d$. For instance:
  - intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

- determines a class of numerical invariants over variables $v_1, \ldots, v_d$. For instance:
  - intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
  - zones $v_i - v_j \geq a$ [Miné, 2001]
Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain

• determines a class of numerical invariants over variables $v_1, \ldots, v_d$. For instance:
  • intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
  • zones $v_i - v_j \geq a$ [Miné, 2001]
  • convex polyhedra $a_1 v_1 + \cdots + a_d v_d \leq b$ [Cousot and Halbwachs, 1978]
Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain
- determines a class of numerical invariants over variables $v_1, \ldots, v_d$. For instance:
  - intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
  - zones $v_i - v_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1v_1 + \cdots + a_dv_d \leq b$ [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 \( v_1, \ldots, v_d \). For instance:
  - intervals \( a \leq v_i \leq b \) [Cousot and Cousot, 1977]
  - zones \( v_i - v_j \geq a \) [Miné, 2001]
  - convex polyhedra \( a_1 v_1 + \cdots + a_d v_d \leq b \) [Cousot and Halbwachs, 1978]

- provides a set of abstract primitives allowing to automatically compute sound properties

Different levels of precision:

\[ v_2 \quad \rightarrow \quad v_1 \]
Determining numerical invariants by abstract interpretation

Central notion: numerical abstract domain
- determines a class of numerical invariants over variables \( v_1, \ldots, v_d \). For instance:
  - intervals \( a \leq v_i \leq b \) [Cousot and Cousot, 1977]
  - zones \( v_i - v_j \geq a \) [Miné, 2001]
  - convex polyhedra \( a_1v_1 + \cdots + a_dv_d \leq b \) [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 $v_1, \ldots, v_d$. For instance:
  - intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
  - zones $v_i - v_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1 v_1 + \cdots + a_d v_d \leq b$ [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 $v_1, \ldots, v_d$. For instance:
  - intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
  - zones $v_i - v_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1 v_1 + \cdots + a_d v_d \leq b$ [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 $v_1, \ldots, v_d$. For instance:
  - intervals $a \leq v_i \leq b$ [Cousot and Cousot, 1977]
  - zones $v_i - v_j \geq a$ [Miné, 2001]
  - convex polyhedra $a_1 v_1 + \cdots + a_d v_d \leq b$ [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

```
1: assume (n ≥ 1);
2: s := malloc(n);
3: i := 0;
4: while i ≤ n − 2 do
5:   s[i] := read();
6:   i := i + 1;
7: done;
8: s[i] := \0;
9: upper := malloc(n);
10: memcpy(upper, s, n);
11: i := 0;
12: while upper[i] ≠ \0 do
13:   c := upper[i];
14:   if (c ≥ 97) ∧ (c ≤ 122) then
15:     upper[i] := c − 32;
16: end;
17: i := i + 1;
18: 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

1: assume \( n \geq 1 \);
2: \( s := \text{malloc}(n) \);
3: \( i := 0 \);
4: while \( i \leq n - 2 \) do
5: \( s[i] := \text{read}() \);
6: \( i := i + 1 \);
7: done;
8: \( s[i] := \backslash 0 \);
9: \( \text{upper} := \text{malloc}(n) \);
10: \( \text{memcpy}(	ext{upper}, s, n) \);
11: \( i := 0 \);
12: while \( \text{upper}[i] \neq \backslash 0 \) do
13: \( c := \text{upper}[i] \);
14: if \( (c \geq 97) \wedge (c \leq 122) \) then
15: \( \text{upper}[i] := c - 32 \);
16: end;
17: \( i := i + 1 \);
18: done;

Typical memory manipulating program:
- reads a string \( s \) from standard input
- copies it in \( \text{upper} \) and capitalizes it

Convex abstract domains: raise a false alarm

iterates up to the first \( \backslash 0 \)
The need of non-convex abstract domains

Typical memory manipulating program:
- reads a string \( s \) from standard input
- copies it in \( \text{upper} \) and capitalizes it

Convex abstract domains: raise a false alarm

Iterates up to the first \( \backslash 0 \)
The need of non-convex abstract domains (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: \( \text{dst}[i] := \text{src}[i] \);
4: done;
The need of non-convex abstract domains (2)

```c
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 n > len_src,
```

![Diagram showing memory copying](image)
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 n > len_src,
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 \( n > \text{len}_\text{src} \), \( \text{len}_\text{dst} = \text{len}_\text{src} \)

\[
\text{memcpy}(\text{dst}, \text{src}, 9)
\]

\[
\begin{align*}
\text{src} & \quad \text{Example}\quad \text{le}\quad \text{nul}\quad 0 \quad ?

dst & \quad \text{Example}\quad \text{le}\quad \text{nul}\quad 0 \quad ?
\end{align*}
\]

\( \text{len}_\text{src} = 7 \)

\( \text{len}_\text{dst} = 7 \)
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 n > len_src, len_dst = len_src
- if n ≤ len_src,
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: \( \text{dst}[i] := \text{src}[i] \);
4: done;

- if \( n > \text{len}_{\text{src}} \), \( \text{len}_{\text{dst}} = \text{len}_{\text{src}} \)
- if \( n \leq \text{len}_{\text{src}} \),

\[ \text{memcpy}(\text{dst}, \text{src}, 5) \]

\( \text{src} \) → Example 0??

\( \text{len}_{\text{src}} = 7 \)

\( \text{dst} \) → Example ???

\( \text{len}_{\text{dst}} \geq 5 \)
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 \( n > \text{len}_\text{src} \), \( \text{len}_\text{dst} = \text{len}_\text{src} \)
- if \( n \leq \text{len}_\text{src} \), \( \text{len}_\text{dst} \geq n \)
The need of non-convex abstract domains (3)

Disjunction of two cases:

- if $n > \text{len}_{\text{src}}$, $\text{len}_{\text{dst}} = \text{len}_{\text{src}}$
- if $n \leq \text{len}_{\text{src}}$, $\text{len}_{\text{dst}} \geq n$

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]

$\rightarrow$ not satisfactory
The need of non-convex abstract domains (3)

Disjunction of two cases:

- if \( n > \text{len}_\text{src} \), \( \text{len}_\text{dst} = \text{len}_\text{src} \)
- if \( n \leq \text{len}_\text{src} \), \( \text{len}_\text{dst} \geq n \)

\[ \iff \min(\text{len}_{\text{src}}, n) = \min(\text{len}_{\text{dst}}, n) \]
The need of non-convex abstract domains (3)

Disjunction of two cases:

- if $n > \text{len}_\text{src}$, $\text{len}_\text{dst} = \text{len}_\text{src}$
- if $n \leq \text{len}_\text{src}$, $\text{len}_\text{dst} \geq n$

\[
\iff \quad \min(\text{len}_\text{src}, n) = \min(\text{len}_\text{dst}, n)
\iff \quad \max(-\text{len}_\text{src}, -n) = \max(-\text{len}_\text{dst}, -n)
\]
The need of non-convex abstract domains (3)

Disjunction of two cases:

- if $n > \text{len}_\text{src}$, $\text{len}_\text{dst} = \text{len}_\text{src}$
- if $n \leq \text{len}_\text{src}$, $\text{len}_\text{dst} \geq n$

\[ \iff \text{min}(\text{len}_{\text{src}}, n) = \text{min}(\text{len}_{\text{dst}}, n) \]
\[ \iff \text{max}(-\text{len}_{\text{src}}, -n) = \text{max}(-\text{len}_{\text{dst}}, -n) \]

a linear equality ... in tropical algebra
Tropical algebra

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

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

$$
\begin{align*}
1 \oplus 1 &= \quad \text{(no inverse!)} \\
1 \otimes 1 &= 2 \\
3 \oplus (-3) &= \quad \text{(same as \oplus)} \\
3 \otimes (-3) &= \quad \text{(same as \otimes)}
\end{align*}
$$
Tropical algebra

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

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

\[
\begin{align*}
1 \oplus 1 &= \max(1, 1) = 1 \\
1 \otimes 1 &= 1 + 1 = 2 \\
3 \oplus (-3) &= \\
3 \otimes (-3) &=
\end{align*}
\]
Tropical algebra

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

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

\[
1 \oplus 1 = \max(1, 1) = 1 \\
1 \otimes 1 = 1 + 1 = 2 \\
3 \oplus (-3) = 3 \\
3 \otimes (-3) =
\]
Tropical algebra

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

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

\[
\begin{align*}
1 \oplus 1 &= \max(1, 1) = 1 \\
1 \otimes 1 &= 1 + 1 = 2 \\
3 \oplus (-3) &= 3 \\
3 \otimes (-3) &= 0
\end{align*}
\]
Tropical algebra

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

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

\[
\begin{align*}
1 \oplus 1 &= \max(1, 1) = 1 \\
1 \otimes 1 &= 1 + 1 = 2 \\
3 \oplus (-3) &= 3 \\
3 \otimes (-3) &= 0
\end{align*}
\]
Tropical algebra

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

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

The addition has no inverse! $\implies$ semi-ring

\[
\begin{align*}
1 \oplus 1 &= \max(1, 1) = 1 \\
1 \otimes 1 &= 1 + 1 = 2 \\
3 \oplus (-3) &= 3 \\
3 \otimes (-3) &= 0
\end{align*}
\]
Tropical polyhedra

- **Tropical affine inequality** =

  \[ \alpha_0 + (\alpha_1 \times x_1) + \cdots + (\alpha_d \times x_d) \leq \beta_0 + (\beta_1 \times x_1) + \cdots + (\beta_d \times x_d) \]
Tropical polyhedra

- *Tropical affine inequality* =

\[ \alpha_0 \oplus (\alpha_1 \otimes x_1) \oplus \ldots \oplus (\alpha_d \otimes x_d) \leq \beta_0 \oplus (\beta_1 \otimes x_1) \oplus \ldots \oplus (\beta_d \otimes x_d) \]
Tropical polyhedra

- *Tropical affine inequality* =

\[
\max(\alpha_0, \alpha_1 + x_1, \ldots, \alpha_d + x_d) \leq \max(\beta_0, \beta_1 + x_1, \ldots, \beta_d x_d)
\]
Tropical polyhedra

- **Tropical affine inequality** =
  \[ \alpha_0 \oplus (\alpha_1 \otimes x_1) \oplus \ldots \oplus (\alpha_d \otimes x_d) \leq \beta_0 \oplus (\beta_1 \otimes x_1) \oplus \ldots \oplus (\beta_d \otimes x_d) \]

- **Tropical polyhedra** = system of tropical affine inequalities
  \[ \max(-\text{len}_{\text{src}},-n) = \max(-\text{len}_{\text{dst}},-n) \]
  \[ \iff \begin{cases} (-\text{len}_{\text{src}}) \oplus (-n) \leq (-\text{len}_{\text{dst}}) \oplus (-n) \\ (-\text{len}_{\text{dst}}) \oplus (-n) \leq (-\text{len}_{\text{src}}) \oplus (-n) \end{cases} \]
Tropical polyhedra

- **Tropical affine inequality**
  \[
  \alpha_0 \oplus (\alpha_1 \otimes x_1) \oplus \ldots \oplus (\alpha_d \otimes x_d) \leq \beta_0 \oplus (\beta_1 \otimes x_1) \oplus \ldots \oplus (\beta_d \otimes x_d)
  \]

- **Tropical polyhedra** = system of tropical affine inequalities
  \[
  \max(-\text{len}_{\text{src}}, -n) = \max(-\text{len}_{\text{dst}}, -n)
  \]
  \[
  \iff \begin{cases}
  (-\text{len}_{\text{src}}) \oplus (-n) \leq (-\text{len}_{\text{dst}}) \oplus (-n) \\
  (-\text{len}_{\text{dst}}) \oplus (-n) \leq (-\text{len}_{\text{src}}) \oplus (-n)
  \end{cases}
  \]

**Idea**: build a numerical domain based on tropical polyhedra
Tropical polyhedra (2)

Very studied in the literature:

- 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]
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]

Algorithmics of tropical polyhedra: little studied
Tropical polyhedra (2)

Very studied in the literature:

- 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]

Algorithmics of tropical polyhedra: **little studied**

- by inequalities
- by vertices/rays
Tropical polyhedra (2)

Very studied in the literature:

- 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]

Algorithmics of tropical polyhedra: little studied
Tropical polyhedra (2)

Very studied in the literature:

- 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]

Algorithmics of tropical polyhedra: **little studied**

- 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 \( x \in \mathbb{R}^d_{\max} \) of

\[
Ax \oplus c \leq Bx \oplus d
\]
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 \( x \in \mathbb{R}^d_{\max} \) of

\[
Ax \oplus c \leq Bx \oplus d
\]

\[
x \leq \max(y, 0)
\]
\[
0 \leq x + 2
\]
\[
x \leq 2
\]
\[
0 \leq \max(x, y - 1)
\]
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 $x \in \mathbb{R}^d_{\max}$ of

$$Ax \oplus c \leq Bx \oplus d$$

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

$0 \leq x + 2$

$x \leq 2$

$0 \leq \max(x, y - 1)$
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 $\mathbb{R}_\text{max}^d$ is the set of the solutions $x \in \mathbb{R}_\text{max}^d$ of

$$ Ax \oplus c \leq Bx \oplus d $$

$$ x \leq \max(y, 0) $$
$$ 0 \leq x + 2 $$
$$ x \leq 2 $$
$$ 0 \leq \max(x, y - 1) $$
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 $x \in \mathbb{R}^d_{\max}$ of

$$Ax \oplus c \leq Bx \oplus d$$

- $x \leq \max(y, 0)$
- $0 \leq x + 2$
- $x \leq 2$
- $0 \leq \max(x, y - 1)$
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 $x \in \mathbb{R}^d_{\max}$ of

$$Ax \oplus c \leq Bx \oplus d$$

- $x \leq \max(y, 0)$
- $0 \leq x + 2$
- $x \leq 2$
- $0 \leq \max(x, y - 1)$
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 $\mathbb{R}_\text{max}^d$ is the set of the solutions $x \in \mathbb{R}_\text{max}^d$ of

$$Ax \oplus c \leq Bx \oplus d$$

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

$$\begin{pmatrix}
0 & 1 \\
2 & 0 \\
0 & 0 \\
1 & -1
\end{pmatrix} \begin{pmatrix}
x \\
y
\end{pmatrix} \oplus \begin{pmatrix}
1 \\
0 \\
2 \\
0
\end{pmatrix}$$
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 $\mathbb{R}^d_{\text{max}}$ is the set of the solutions $x \in \mathbb{R}^d_{\text{max}}$ of

$$Ax \oplus c \leq Bx \oplus d$$

Two-sided inequalities

```
\begin{align*}
\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \end{pmatrix}
\end{align*}
```
Tropical halfspaces

Tropical halfspace = set of the solutions \( x \in \mathbb{R}^d_{\text{max}} \) 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 $p_i \in P$,
- and of rays $r_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j$$

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

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 \( p_i \in P \),
- and of rays \( r_j \in R \),

of the form:

\[
\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j
\]

where \( \bigoplus_{i=1}^{p} \lambda_i = 1 \).

no positivity constraints on the \( \lambda_i \) and \( \mu_j \):

\[
\forall x \in \mathbb{R}_{\text{max}}. \ x \geq 0 \ (= -\infty)
\]
Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:
- points \( p_i \in P \),
- and of rays \( r_j \in R \),
of the form:

\[
\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j
\]

where \( \bigoplus_{i=1}^{p} \lambda_i = 1 \).

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 $p_i \in P$,
- and of rays $r_j \in R$,
of the form:

$$\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j$$

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

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 \( p_i \in P \),
- and of rays \( r_j \in R \),
of the form:

\[
\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j
\]

where \( \bigoplus_{i=1}^{p} \lambda_i = 1 \).

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

A tropical polyhedron is formed by the combinations of finitely many:
- points \( p_i \in P \),
- and of rays \( r_j \in R \),

of the form:

\[
\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j
\]

where \( \bigoplus_{i=1}^{p} \lambda_i = 1 \).

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

**Tropical polyhedra = convex hull of generators**
Tropical polyhedra = convex hull of generators

Definition (Generating representation)

A tropical polyhedron is formed by the combinations of finitely many:
- points $p_i \in P$,
- and of rays $r_j \in R$,

of the form:

$$\bigoplus_{i=1}^{p} \lambda_i p_i \oplus \bigoplus_{j=1}^{q} \mu_j r_j$$

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

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:

\[ Ax \oplus c \leq Bx \oplus d \]

\((P, R)\)
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

= \textbf{incremental} method computing generators from inequalities

\begin{align*}
x &\leq y \oplus 1 \\
1 &\leq 2x \\
x &\leq 2 \\
1 &\leq x \oplus (-1)y
\end{align*}
Tropical double description method

= **incremental** method computing generators from inequalities

\[ Q : \begin{align*}
    x & \leq y \oplus 1 \\
    1 & \leq 2x \\
    x & \leq 2 \\
    1 & \leq x \oplus (-1)y
\end{align*} \]
Tropical double description method

= **incremental** method computing generators from inequalities

\[
Q : \begin{cases} 
  x \leq y \oplus 1 \\
  1 \leq 2x \\
  x \leq 2 
\end{cases}
\]

\[
\mathcal{H} : 1 \leq x \oplus (-1)y
\]
Tropical double description method

= incremental method computing generators from inequalities

\[ \begin{align*}
Q : & \quad \begin{cases} 
  x \leq y \oplus 1 \\
  1 \leq 2x \\
  x \leq 2 
\end{cases} \\
H : & \quad 1 \leq x \oplus (-1)y 
\end{align*} \]

Generators of \( Q \): \( r^0, p^1, p^2, p^3 \)
Tropical double description method

= incremental method computing generators from inequalities

\[Q:\begin{cases}x \leq y \oplus 1 \\1 \leq 2x \\x \leq 2\end{cases}\]

\[\mathcal{H} : 1 \leq x \oplus (-1)y\]

Generators of \(Q\): \(r^0, p^1, p^2, p^3\)
Tropical double description method

= incremental method computing generators from inequalities

\[ Q : \begin{align*}
  x &\leq y \oplus 1 \\
  1 &\leq 2x \\
  x &\leq 2
\end{align*} \]

\[ \mathcal{H} : \ 1 \leq x \oplus (-1)y \]

Generators of \( Q \): \( r^0, p^1, p^2, p^3 \)

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

- generators of \( Q \) in \( \mathcal{H} \)
Tropical double description method

= incremental method computing generators from inequalities

\[ Q : \begin{cases} 
  x \leq y \oplus 1 \\
  1 \leq 2x \\
  x \leq 2 
\end{cases} \]

\[ \mathcal{H} : 1 \leq x \oplus (-1)y \]

Generators of \( Q \): \( r^0, p^1, p^2, p^3 \)

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

- generators of \( Q \) in \( \mathcal{H} \)
- combinations of green and red generators of \( Q \) lying on the boundary of \( \mathcal{H} \)
Tropical double description method

= **incremental** method computing generators from inequalities

\[ Q : \begin{align*}
  x &\leq y \oplus 1 \\
  1 &\leq 2x \\
  x &\leq 2
\end{align*} \]

\[ \mathcal{H} : \ 1 \leq x \oplus (-1)y \]

Generators of \( Q \): \( r^0, p^1, p^2, p^3 \)

Intersection of \( Q \) and \( \mathcal{H} \)

generated by:

- generators of \( Q \) in \( \mathcal{H} \)
- combinations of green and red generators of \( Q \) lying on the boundary of \( \mathcal{H} \)
Tropical double description method

= incremental method computing generators from inequalities

\[ Q : \begin{cases} 
  x \leq y \oplus 1 \\
  1 \leq 2x \\
  x \leq 2 
\end{cases} \]

\[ H : \quad 1 \leq x \oplus (-1)y \]

Generators of \( Q \): \( r^0, p^1, p^2, p^3 \)

Intersection of \( Q \) and \( H \) generated by:

- generators of \( Q \) in \( H \)
- combinations of green and red generators of \( Q \) lying on the boundary of \( H \)
Tropical double description method

= **incremental** method computing generators from inequalities

\[
\begin{align*}
Q : & \quad x \leq y \oplus 1 \\
& \quad 1 \leq 2x \\
& \quad x \leq 2 \\
H : & \quad 1 \leq x \oplus (-1)y
\end{align*}
\]

Generators of \( Q \): \( r^0, p^1, p^2, p^3 \)

Intersection of \( Q \) and \( H \)
generated by:

- generators of \( Q \) in \( H \)
- combinations of green and red generators of \( Q \) lying on the boundary of \( H \)
Theorem (Elementary step of the DDM, Allamigeon et al. (STACS’10))

Consider:
- a tropical polyhedron \( Q \) of generating representation \( (\{p^i\}, \{r^j\}) \)
- a tropical halfspace \( H \) defined by \( ax \oplus c \leq bx \oplus d \)

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

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

$$Q = \left\{ p^i \mid ap^i \oplus c \leq bp^i \oplus d \right\}$$

$$\cup \left\{ \lambda p^i \oplus \mu p^i \mid ap^i \oplus c \leq bp^i \oplus d \text{ and } ap^i \oplus c > bp^i \oplus d \right\}$$

$$\lambda = \kappa^{-1}(ap^i \oplus c), \mu = \kappa^{-1}(bp^i \oplus d), \kappa = ap^i \oplus c \oplus bp^i \oplus d$$

$$\cup \left\{ p^i \oplus \alpha r^j \mid ap^i \oplus c \leq bp^i \oplus d \text{ and } ar^j > br^i, \alpha = (ar^j)^{-1}(bp^i \oplus d) \right\}$$

$$\cup \left\{ \beta r^i \oplus p^j \mid ar^i < br^i \text{ and } ap^j \oplus c > bp^j \oplus d, \beta = (br^i)^{-1}(ap^i \oplus c) \right\}$$

$$R = \left\{ r^i \mid ar^i \leq br^i \right\} \cup \left\{ (ar^j)r^i \oplus (br^i)r^j \mid ar^i \leq br^i \text{ and } ar^i > br^i \right\}$$
Tropical double description method (2): homogenized version

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

Consider:
- a tropical cone $C$ of generating representation $G = (g^i)_i$
- a tropical linear halfspace $H$ defined by $ax \leq bx$

Then $C \cap H$ is generated by:

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

Static analysis of memory manipulations by abstract interpretation — Xavier Allamigeon — 25/57
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 $Ax \oplus c \leq Bx \oplus d$ \quad \Rightarrow \quad \text{Reachability problem is a directed hypergraph}
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:

\[ \{ v, w \} \longrightarrow \{ x, y \} \]
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:

![Directed hypergraph diagram](image-url)
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:

![Directed hypergraph diagram]

Static analysis of memory manipulations by abstract interpretation — Xavier Allamigeon — 27/57
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:

![Directed hypergraph diagram]
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 (2)

Let $\mathcal{P}$ given by

$$
\begin{align*}
A_1x \oplus c_1 & \leq B_1x \oplus d_1 \\
\vdots & \\
A_p x \oplus c_p & \leq B_p x \oplus d_p
\end{align*}
$$

Is $p \in \mathcal{P}$ extreme?
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by
\[
\begin{align*}
A_1x \oplus c_1 &\leq B_1x \oplus d_1 \\
\vdots &
\end{align*}
\]
\[
A_px \oplus c_p \leq B_px \oplus d_p
\]
Is $p \in \mathcal{P}$ extreme?

Definition

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

for each $k$ such that $A_kp \oplus c_k = B_kp \oplus d_k$

\[
\text{arg max}(B_kp \oplus d_k)
\]

\[
\text{arg max}(A_kp \oplus c_k)
\]
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by

\[
\begin{align*}
A_1 x \oplus c_1 & \leq B_1 x \oplus d_1 \\
& \vdots \\
A_p x \oplus c_p & \leq B_p x \oplus d_p
\end{align*}
\]

Is $p \in \mathcal{P}$ extreme?

**Definition**

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

\[
\text{arg max}(B_k p \oplus d_k)
\]

for each $k$ such that

\[
A_k p \oplus c_k = B_k p \oplus d_k
\]

\[
\text{arg max}(A_k p \oplus c_k)
\]

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

$p \in \mathcal{P}$ is extreme $\iff$ $\mathcal{H}(p)$ admits a sink
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by

$$
\begin{align*}
A_1 x \oplus c_1 &\leq B_1 x \oplus d_1 \\
\vdots \\
A_p x \oplus c_p &\leq B_p x \oplus d_p 
\end{align*}
$$

Is $p \in \mathcal{P}$ extreme?

Definition

The tangent directed hypergraph $\mathcal{H}(p)$ at the point $p$ is formed by the hyperedges for each $k$ such that $A_k p \oplus c_k = B_k p \oplus d_k$

$$
\arg \max(B_k p \oplus d_k)
$$

$$
\arg \max(A_k p \oplus c_k)
$$

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

$p \in \mathcal{P}$ is extreme $\iff$ $\mathcal{H}(p)$ admits a sink reachable from all nodes
Combinatorial characterization of extreme points (2)

Let $\mathcal{P}$ given by

$$
\begin{align*}
A_1 x \oplus c_1 & \leq B_1 x \oplus d_1 \\
& \vdots \\
A_p x \oplus c_p & \leq B_p x \oplus d_p
\end{align*}
$$

Is $p \in \mathcal{P}$ extreme?

Definition

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

for each $k$ such that

$A_k p \oplus c_k = B_k p \oplus d_k$

$$
\arg \max (B_k p \oplus d_k) \\
\arg \max (A_k p \oplus c_k)
$$

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

$p \in \mathcal{P}$ is extreme $\iff \mathcal{H}(p)$ admits a sink reachable from all nodes

Stronger property than in the classical case:

- saturated inequalities
- “maximality” criterion
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[
x \leq \max(y, 0) \\
0 \leq x + 2 \\
x \leq 2 \\
0 \leq \max(x, y - 1)
\]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[-2 = x \leq \max(y, 0) = 1\]
\[0 \leq x + 2\]
\[x \leq 2\]
\[0 \leq \max(x, y - 1)\]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[-2 = x < \max(y, 0) = 1\]
\[ 0 \leq x + 2 \]
\[ x \leq 2 \]
\[ 0 \leq \max(x, y - 1) \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[-2 = x < \max(y, 0) = 1 \]
\[ 0 \leq x + 2 = 0 \]
\[ x \leq 2 \]
\[ 0 \leq \max(x, y - 1) \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[ -2 = x < \max(y, 0) = 1 \]
\[ 0 = x + 2 = 0 \]
\[ x \leq 2 \]
\[ 0 \leq \max(x, y - 1) \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[ -2 = x < \max(y, 0) = 1 \]
\[ 0 = x + 2 = 0 \]
\[ x \leq 2 \]
\[ 0 \leq \max(x, y - 1) \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[ -2 = x < \max(y, 0) = 1 \]
\[ 0 = x + 2 = 0 \]
\[ -2 = x \leq 2 \]
\[ 0 \leq \max(x, y - 1) \]
Combinatorial characterization of extreme points (let’s practice!)

\( p = (-2, 1) \)

\[-2 = x < \max(y, 0) = 1\]
\[0 = x + 2 = 0\]
\[-2 = x < 2\]
\[0 \leq \max(x, y - 1)\]
Combinatorial characterization of extreme points (let’s practice!)  

\[ p = (-2, 1) \]

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

\[ 0 = x + 2 = 0 \]

\[-2 = x < 2 \]

\[ 0 \leq \max(x, y - 1) = 0 \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[ -2 = x < \max(y, 0) = 1 \]
\[ 0 = x + 2 = 0 \]
\[ -2 = x < 2 \]
\[ 0 = \max(x, y - 1) = 0 \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[ -2 = x < \max(y, 0) = 1 \]
\[ 0 = x + 2 = 0 \]
\[ -2 = x < 2 \]
\[ 0 = \max(x, y - 1) = 0 \]
Combinatorial characterization of extreme points (let’s practice!)

\[ p = (-2, 1) \]

\[ -2 = x < \max(y, 0) = 1 \]
\[ 0 = x + 2 = 0 \]
\[ -2 = x < 2 \]
\[ 0 = \max(x, y - 1) = 0 \]

\[ \mathcal{H}(p) \text{ has a sink: } z \]
Efficient evaluation of the extremality criterion

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest } \text{Scc} \]

(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

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest SCC} \]

(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.
Efficient evaluation of the extremality criterion

\[ \mathcal{H} \text{ admits a sink} \iff \mathcal{H} \text{ has a greatest SCC} \]

(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

\[ \mathcal{H} \text{ admits a sink} \iff \mathcal{H} \text{ has a greatest SCC} \]

(order induced by the reachability relation: \( C_1 \prec C_2 \text{ iff } C_1 \text{ reaches } C_2 \))

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

\[ \mathcal{H} \text{ admits a sink} \iff \mathcal{H} \text{ has a greatest SCC} \]

(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 $\text{Scc}$
(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 $\text{Sccs}$ can be determined in almost linear time.*

Sequence of operations of two kinds:
- merging some nodes in the directed hypergraph
- discovering the maximal $\text{Sccs}$ in the underlying directed graph (Tarjan)
Efficient evaluation of the extremality criterion

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest SCC} \]
(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

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest } \text{Scc} \]
(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

\[ \mathcal{H} \text{ admits a sink} \iff \mathcal{H} \text{ has a greatest SCC} \]
(order induced by the reachability relation: \( C_1 \preceq C_2 \) 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

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest SCC} \]

(order induced by the reachability relation: \( C_1 \prec C_2 \text{ iff } C_1 \text{ reaches } C_2 \))

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

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest SCC} \]

(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

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

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

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest SCC} \]

(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

\[ \mathcal{H} \text{ admits a sink } \iff \mathcal{H} \text{ has a greatest SCC} \]

(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)

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

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

Static analysis of memory manipulations by abstract interpretation — Xavier Allamigeon — 31/57
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

\textbf{INEQToGen} = combination of

- tropical double description method
- elimination of non-extreme elements by hypergraph-based characterization
From inequalities to generators

\textbf{INEQToGen} = combination of
- tropical double description method
- elimination of non-extreme elements by hypergraph-based characterization

\textbf{Proposition}

The time complexity of \textbf{INEQToGen} is

\[ O(p^2 \ d \ G_{\text{max}}^2) \]

where:
- \( d = \text{dimension} \)
- \( p = \text{nb of constraints in } Ax \oplus c \leq Bx \oplus d \)
- \( G_{\text{max}} = \text{maximal number of extreme generators in the intermediate polyhedra} \)
From inequalities to generators

\textbf{INEQToGen} = combination of

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

**Proposition**

The time complexity of \textbf{INEQToGen} is

\[
O(p^2 \ d \ G_{\text{max}}^2)
\]

where:

- \( d = \text{dimension} \)
- \( p = \text{nb of constraints in } Ax \oplus c \leq Bx \oplus d \)
- \( G_{\text{max}} = \text{maximal number of extreme generators in the intermediate polyhedra leading term, exponential in } d \)
Comparison to existing works

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

Notations

- $d =$ dimension
- $p =$ nb of constraints in $Ax \oplus c \leq Bx \oplus d$
- $G_{max} =$ maximal number of extreme generators in the intermediate polyhedra leading term, exponential in $d$
Comparison to existing works

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

Tropical world:

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

Notations

- \( d \) = dimension
- \( p \) = nb of constraints in \( Ax \oplus c \leq Bx \oplus d \)
- \( G_{\text{max}} \) = maximal number of extreme generators in the intermediate polyhedra leading term, exponential in \( d \)
Comparison to existing works

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

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

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

Notations

- \( d = \) dimension
- \( p = \) nb of constraints in \( Ax \oplus c \leq Bx \oplus d \)
- \( G_{\text{max}} = \) maximal number of extreme generators in the intermediate polyhedra leading term, exponential in \( d \)
Comparison to existing works

Time complexity of \( \text{INEQToGEN} = O(p^2 \, d \, G_{\text{max}}^2) \)

Tropical world:
- seminal algorithm due to Butkovič and Hegedüs [1984]: double exponential
- implementation in the Max-plus toolbox of \textsc{Scilab} and \textsc{ScicosLab}, later refined in Allamigeon et al. (SAS’08)
  \[ O(p \, d \, G_{\text{max}}^4) \]
- Elimination of non-extreme elements by residuation [see Vorobyev, 1967,\n  Cuninghame-Green, 1976]

Classical world: Motzkin et al. [1953], Fukuda and Prodon [1996]
  \[ O(p^2 \, G_{\text{max}}^3) \]

Notations
- \( d = \) dimension
- \( p = \) nb of constraints in \( Ax \oplus c \leq Bx \oplus d \)
- \( G_{\text{max}} = \) 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

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

- $T$: 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

\textbf{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 \( \mathbb{R}^d \) defined by \( p \) inequalities is bounded by

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

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

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

Expressivity: conjunction of max-invariants over variables $v_1, \ldots, v_d$

$$\max(\alpha_0, \alpha_1 + v_1, \ldots, \alpha_d + v_d) \leq \max(\beta_0, \beta_1 + v_1, \ldots, \beta_d + v_d)$$
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 $(P, R)$

Expressivity: conjunction of max-invariants over variables $v_1, \ldots, v_d$

$$\max(\alpha_0, \alpha_1 + v_1, \ldots, \alpha_d + v_d) \leq \max(\beta_0, \beta_1 + v_1, \ldots, \beta_d + v_d)$$

$\implies$ connected disjunctions of zone invariants $v_i - v_j \geq \kappa$:

$$\bigvee_{1 \leq i \leq d \atop \beta_i \neq -\infty} \left[ \left( \bigwedge_{1 \leq j \leq d} \alpha_j - \beta_i \leq v_i - v_j \right) \land (\alpha_0 - \beta_i \leq v_i) \right] \lor \left[ \bigwedge_{1 \leq i \leq d \atop \alpha_i \neq -\infty} v_i \leq \beta_0 - \alpha_i \right]$$
Some abstract primitives

Abstract primitives generally use one of the representations:

\[ \implies \text{INEQToGEN and GENToINEQ are critical} \]

- **Abstract union**: given two polyhedra \( P \) and \( Q \), and \( (P, R) \) and \( (Q, S) \) their generating representations,

\[
P \cup Q \overset{\text{def}}{=} \text{polyhedron generated by } (P \cup Q, R \cup S)
\]
Some abstract primitives

Abstract primitives generally use one of the representations:

⇒ \textsc{IneqToGen} and \textsc{GenToIneq} are critical

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

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

Abstract primitives generally use one of the representations:

\[ \text{INEQToGen and GENToINEQ are critical} \]

- **Abstract union**: given two polyhedra \( P \) and \( Q \), and \((P, R)\) and \((Q, S)\) their generating representations,

\[ P \sqcup Q \overset{\text{def}}{=} \text{polyhedron generated by } (P \cup Q, R \cup S) \]
Some abstract primitives

Abstract primitives generally use one of the representations:

\[ \text{INEQToGen} \text{ and GENToINEQ are critical} \]

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

\[ P \cup Q \overset{\text{def}}{=} \text{polyhedron generated by } (P \cup Q, R \cup S) \]

Non-extreme generators can be eliminated in polynomial time
Some abstract primitives

Abstract primitives generally use one of the representations:

$\implies$ IneqToGen and GenToIneq are critical

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

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

- 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 $P_0 \subset \cdots \subset P_n \subset \cdots$, the sequence defined by
  \[
  \begin{align*}
  Q_0 & \overset{\text{def}}{=} P_0 \\
  Q_{n+1} & \overset{\text{def}}{=} Q_n \lor P_{n+1}
  \end{align*}
  \]
eventually stabilizes.
Some abstract primitives (2)

- abstract intersection, assignments, … all sound and exact
- widening operators to enforce convergence:
  - $\nabla_{cons}$: eliminate non-stable inequalities
Some abstract primitives (2)

- abstract intersection, assignments, ... all sound and exact
- widening operators to enforce convergence:
  - $\nabla_{cons}$: eliminate non-stable inequalities
Some abstract primitives (2)

- abstract intersection, assignments, ... all sound and exact
- widening operators to enforce convergence:
  - $\nabla_{cons}$: eliminate non-stable inequalities
  - $\nabla_{gen}$: 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 than 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 than the domain of zones

\[\text{toZone}(\mathcal{P}) = \text{extract the smallest zone abstract element containing } \mathcal{P}\]
Comparison with the abstract domain of zones

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

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

\[
\min(\alpha_0, \alpha_1 + v_1, \ldots, \alpha_d + v_d) \leq \min(\beta_0, \beta_1 + v_1, \ldots, \beta_d + v_d)
\]

using MaxPoly on special variables \(w_i = "-v_i"\).
Other tropical polyhedra based abstract domains

- MinPoly: infer min-invariants

\[
\max(-\alpha_0, -\alpha_1 + w_1, \ldots, -\alpha_d + w_d) \geq \max(-\beta_0, -\beta_1 + w_1, \ldots, -\beta_d + w_d)
\]

using MaxPoly on special variables \( w_i = "-v_i" \).
Other tropical polyhedra based abstract domains

- **MinPoly**: infer min-invariants

  \[
  \min(\alpha_0, \alpha_1 + v_1, \ldots, \alpha_d + v_d) \leq \min(\beta_0, \beta_1 + v_1, \ldots, \beta_d + v_d)
  \]

  using MaxPoly on special variables \(w_i = "-v_i"\).

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

  \[
  \max(\alpha_0, \alpha_1 + v_1, \ldots, \alpha_d + v_d, \alpha_{d+1} - v_1, \ldots, \alpha_{2d} - v_d)
  \leq \max(\beta_0, \beta_1 + v_1, \ldots, \beta_d + v_d, \beta_{d+1} - v_1, \ldots, \beta_{2d} - v_d)
  \]

  using MaxPoly on the \(v_i\) and \(w_i = "-v_i"\).
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)`

```plaintext
1:   i := 0;
2:   while i ≤ n - 1 do
3:     dst[i] := src[i];
4:     i := i + 1;
5:   done;
```
Memory manipulating programs

- widespread library functions
  - `memcpy(dst, src, n)`
    
    ```
    i := 0;
    while i ≤ 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)`
    
    ```
    1:  i := 0;
    2:  while i ≤ n − 1 do
    3:      dst[i] := src[i];
    4:      i := i + 1;
    5:  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

```plaintext
1: assume (n \geq 1);
2: s := malloc(n);
3: i := 0;
4: while i \leq n - 2 do
5:   s[i] := read();
6:   i := i + 1;
7: done;
8: s[i] := \0;
9: upper := malloc(n);
10: memcpy(upper, s, n);
11: i := 0;
12: while upper[i] \neq \0 do
13:   c := upper[i];
14:   if (c \geq 97) \land (c \leq 122) then
15:     upper[i] := c - 32;  \text{ iterates up to the first } \0
16:   end;
17:   i := i + 1;
18: done;
```
Memory manipulating programs (2)

- programs embedding memory manipulation primitives

```
1: assume (n ≥ 1);
2: s := malloc(n);
3: i := 0;
4: while i ≤ n − 2 do
5:   s[i] := read();
6:   i := i + 1;
7: done;
8: s[i] := \0;
9: upper := malloc(n);
10: memcpy(upper, s, n);
11: i := 0;
12: while upper[i] ≠ \0 do
13:   c := upper[i];
14:   if (c ≥ 97) ∧ (c ≤ 122) then
15:     upper[i] := c − 32;
16:   end;
17:   i := i + 1;
18: done;
```

- convex polyhedra: raise a false alarm

\( \text{len}_{\text{upper}} \leq \text{sz}_{\text{upper}} \)
Memory manipulating programs (2)

- programs embedding memory manipulation primitives

1: assume \( n \geq 1 \);
2: \( s := \text{malloc}(n) \);
3: \( i := 0 \);
4: while \( i \leq n - 2 \) do
5: \( s[i] := \text{read}() \);
6: \( i := i + 1 \);
7: done;
8: \( s[i] := \backslash 0 \);
9: \( \text{upper} := \text{malloc}(n) \);
10: \( \text{memcpy}(\text{upper}, s, n) \);
11: \( i := 0 \);
12: while \( \text{upper}[i] \neq \backslash 0 \) do
13: \( c := \text{upper}[i] \);
14: if (\( c \geq 97 \) \&\& \( c \leq 122 \)) then
15: \( \text{upper}[i] := c - 32 \);
16: end;
17: \( i := i + 1 \);
18: done;

- convex polyhedra: raise a false alarm
  \( \text{len}_{\text{upper}} \leq \text{sz}_{\text{upper}} \)

- MinPoly: no buffer overflow
  \( \text{len}_{\text{upper}} < \text{sz}_{\text{upper}} \)

iterates up to the first \( \backslash 0 \)
Disjunctive invariants

- class of programs derived from predicate abstraction

```plaintext
1:  i := p₁;
2:  while i ≤ p₂ − 1 do
3:      i := i + 1;
4:  done;
5:  while i ≤ p₃ − 1 do
6:      i := i + 1;
7:  done;
```

\[ i = \max(p₁, \ldots, pₙ) \]

- tropical polyhedra:
  - linear growth of the representation
  - scales up to large values of \( n \) (\( n = 60 \rightarrow 19 \text{ s} \))

- classical disjunctive techniques:
  - exponential growth of the representation

\[ \begin{align*}
p₁ & \geq p₂, & p₁ & \geq p₃, & p₁ & \geq p₂, & p₁ & \leq p₂ - 1, \nonumber 
p₁ & \geq p₃, & p₁ & \leq p₃ - 1, & p₁ & \leq p₁ - 1, & p₁ & \leq p₃ - 1, 
i & = p₁, & i & = p₃, & i & = p₂, & i & = p₃ \end{align*} \]

- not practical for large values of \( n \) (\( n = 60 \rightarrow 10^5 \text{ terabytes} \))
Disjunctive invariants (2)

Analysis of sort algorithms:

- Analysis for 10 elements:
  - 1979.7 s with tropical polyhedra
  - not practical with existing disjunctive techniques ($2^{45}$ disjunction)

**leftmost elt = min of the initial elements**

**rightmost elt = max of the initial elements**
### Benchmarks

**Experimentations on a 3 GHz Intel Xeon with 3 Gb RAM**

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

Algorithmics of tropical polyhedra

- output-sensitive algorithm for inequalities $\leftrightarrow$ generators
- tropical linear programming, [see Cuninghame-Green and Butkovic, 2003]
  - how to find a point in a tropical polyhedron in polynomial time? $\text{NP} \cap \text{coNP}$ [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

\[
\max(\alpha_0, \alpha_1 + f_1, \ldots, \alpha_p + f_p) \leq \max(\beta_0, \beta_1 + f'_1, \ldots, \beta_p + f'_q)
\]

with $f_i, f'_j$ classical linear forms over $v_1, \ldots, v_d$
- 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.


V. Nitica and I. Singer. Max-plus convex sets and max-plus semispaces. I.

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


K. Zimmermann. A general separation theorem in extremal algebras.