TreeKs: a Functor to Make Abstract Numerical Domains Scalable

Mehdi Bouaziz

École normale supérieure, Paris

Fourth International Workshop on Numerical and Symbolic Abstract Domains

September 10, 2012 – Deauville, France
Motivation

Numerical static analysis:

- automatic and static discovery of properties on the numerical variables of a program

Applications:

- static verification of programs
- invariant discovery
- program optimization
Abstract interpretation [Cousot Cousot 77] defines a formal framework of sound approximations of semantics.

A numerical abstract domain is:

- a set \( \mathcal{D}_V \) of computer-representable abstract values,
- a concretisation \( [ . ] : \mathcal{D}_V \rightarrow \mathcal{P}(V \mapsto \mathbb{Q}) \),
- a comparison algorithm \( \sqsubseteq \mathcal{D}_V \) of abstract values,
- effective algorithms to compute sound abstractions of the operations: intersection \( \cap^{\mathcal{D}_V} \), union \( \sqcup^{\mathcal{D}_V} \), projection \( \exists^{\mathcal{D}_V} \), . . .
- a widening \( \nabla^{\mathcal{D}_V} \) to ensure termination, if needed.
Numerical abstract domains: basics

Intervals [Cousot Cousot 76]

\[ \bigwedge_i a_i \leq X_i \leq b_i \]
Non-relational
Linear cost

Polyhedra [Cousot Halbwachs 78]

\[ \bigwedge_j \sum_i a_{ij} X_i \leq b_j \]
Relational and very precise
Worst-case exponential cost
Weakly relational numerical abstract domains

Zones [Miné 01]

\[ \bigwedge_{i,j} X_i - X_j \leq c_{ij} \]

Weakly relational
Cubic cost

Octagons [Miné 01]

\[ \bigwedge_{i,j} X_i \pm X_j \leq c_{ij} \]
Cubic cost

Logahedra [Howe King 09]

\[ \bigwedge_{i,j} 2^{a_i} X_i \pm 2^{b_j} X_j \leq c_{ij} \]
Cubic cost

TVPI [Simon King Howe 02]

\[ \bigwedge_{i,j} a_i X_i + b_j X_j \leq c_{ij} \]
Quasi-cubic cost

Octahedra [Clarísó Cortadella 07]

\[ \bigwedge \sum_i X_i \leq c \]
Worst-case exponential cost
Our contribution: TreeKs

- a domain functor
- applied to linear inequality domains
- with a configurable cost/precision tradeoff
Our contribution: TreeKs

- a domain functor
- applied to linear inequality domains
- with a configurable cost/precision tradeoff

Outline:
- the completion operation
- scaling up with packs
- application to the domain of zones
Completion: a key operation

- Common point of the weakly relational domains
- Goal: making explicit the implicit relations
- Done by constraint combination/propagation
- Needed for the other operations (⊔, ⋂, ⊆, ...)
- Dominates the cost of the domain
Closure operation: example

Domain of zones \((\bigwedge_{ij} X_i - X_j \leq b_{ij})\)

\[ V = \{x, y, z\} \]
Closure operation: example

Domain of zones \((\bigwedge_{i,j} X_i - X_j \leq b_{ij})\)

\[ \mathcal{V} = \{x, y, z\} \]

\[-x \leq -1\]
Closure operation: example

Domain of zones \((\bigwedge_{ij} X_i - X_j \leq b_{ij})\)
\[\mathcal{V} = \{x, y, z\}\]

\[-x \leq -1\]
\[x - y \leq 0\]
Closure operation: example

Domain of zones \((\wedge_{ij} X_i - X_j \leq b_{ij})\)
\[ V = \{x, y, z\} \]

\[-x \leq -1\]
\[x - y \leq 0\]
\[y - z \leq -2\]
Closure operation: example

Domain of zones \((\bigwedge_{ij} X_i - X_j \leq b_{ij})\)

\[ V = \{x, y, z\} \]

\[-x \leq -1\]
\[x - y \leq 0\]
\[y - z \leq -2\]

\[-y \leq -1\]
Closure operation: example

Domain of zones \((\bigwedge_{ij} X_i - X_j \leq b_{ij})\)

\(\mathcal{V} = \{x, y, z\}\)

\(-x \leq -1\)
\(x - y \leq 0\)
\(y - z \leq -2\)

\(-y \leq -1\)
\(-z \leq -3\)
Closure operation: example

Domain of zones $(\bigwedge_{ij} X_i - X_j \leq b_{ij})$

$V = \{x, y, z\}$

$-x \leq -1$

$x - y \leq 0$

$y - z \leq -2$

$-y \leq -1$

$-z \leq -3$

$x - z \leq -2$
Closure operation: example

Domain of zones \( \bigwedge_{ij} X_i - X_j \leq b_{ij} \)

\[ V = \{ x, y, z \} \]

\( -x \leq -1 \)
\( x - y \leq 0 \)
\( y - z \leq -2 \)

\( -y \leq -1 \)
\( -z \leq -3 \)
\( x - z \leq -2 \)

Done!
Domain of zones: representation

We represent a set of difference constraints between two variables \((X_i - X_j \leq m_{ji})\) by a potential graph or by a DBM (Difference Bound Matrix).

\[
\begin{array}{ccccc}
0 & x & y & z \\
0 & 0 & +\infty & +\infty & +\infty \\
x & -1 & 0 & +\infty & +\infty \\
y & +\infty & 0 & 0 & +\infty \\
z & +\infty & +\infty & -2 & 0 \\
\end{array}
\]

\[
0 - x \leq -1 \\
x - y \leq 0 \\
y - z \leq -2
\]
Domain of zones: representation

We represent a set of difference constraints between two variables \((X_i - X_j \leq m_{ji})\) by a potential graph or by a DBM (Difference Bound Matrix).

\begin{align*}
0 - x &\leq -1 \\
x - y &\leq 0 \\
y - z &\leq -2
\end{align*}

\begin{align*}
0 - y &\leq -1 \\
0 - z &\leq -3 \\
x - z &\leq -2
\end{align*}
In the domain of zones, the completion operation is a shortest-path closure.

---

**Floyd-Warshall algorithm** $O(n^3)$

```plaintext
for k ← 1 to N do
    for i ← 1 to N do
        for j ← 1 to N do
            $m_{ij} ← \min(m_{ij}, m_{ik} + m_{kj})$
```

At the end:

\[
\begin{align*}
\forall i, j, k, & m_{ij} \leq m_{ik} + m_{kj} \\
\exists i, & m_{ii} < 0
\end{align*}
\]

if satisfiable

if unsatisfiable
Domain of zones: operators

After completion, operators are pointwise.

Join (best approximation of union):

$$(m \sqcup n)_{ij} = \max(m_{ij}, n_{ij})$$

Forget operator (projection):

$$\exists_{X_k} m_{ij} = \begin{cases} m_{ij} & \text{if } i \neq k \text{ and } j \neq k \\ 0 & \text{if } i = j = k \\ +\infty & \text{otherwise} \end{cases}$$
How to scale up: variable packing

Principle:

- split variables into packs
- use a DBM per pack
How to scale up: variable packing

Principle:
- split variables into packs
- use a DBM per pack
How to scale up: variable packing

Principle:
- split variables into packs
- use a DBM per pack
How to scale up: variable packing

**Principle:**
- split variables into packs
- use a DBM per pack

**Cost:** linear for bounded-size packs
How to scale up: variable packing

Principle:
- split variables into packs
- use a DBM per pack

Cost: linear for bounded-size packs
Information loss: no communication between packs!
How to scale up: variable packing

Principle:
- split variables into packs
- use a DBM per pack

Cost: linear for bounded-size packs
Information loss: no communication between packs!
Solution: intervals constraints sharing
How to scale up: variable packing

Principle:
- split variables into packs
- use a DBM per pack

Cost: linear for bounded-size packs
Information loss: no communication between packs!
Solution: intervals constraints sharing
Not good enough!
How to scale up: variable packing

**Principle:**
- split variables into packs
- use a DBM per pack

\[ P_1 = \{ t, x, y \} \]
\[ t \leq y \]
\[ y \leq x \]

\[ P_2 = \{ t, x, z \} \]
\[ x \leq z \]
\[ z \leq t \]

Cost: linear for bounded-size packs
Information loss: no communication between packs!
Solution: intervals constraints sharing
Not good enough!
How to scale up: variable packing

Principle:
▶ split variables into packs
▶ use a DBM per pack

$P_1 = \{t, x, y\}$
$P_2 = \{t, x, z\}$

$\begin{align*}
t &\leq y \\
y &\leq x \\
t &\leq x
\end{align*}$

$\begin{align*}
x &\leq z \\
z &\leq t \\
x &\leq t
\end{align*}$

Cost: linear for bounded-size packs
Information loss: no communication between packs!
Solution: intervals constraints sharing
Not good enough!
How to scale up: variable packing

**Principle:**
- split variables into packs
- use a DBM per pack

\[
\begin{align*}
P_1 &= \{t, x, y\} \\
\text{and} \quad P_2 &= \{t, x, z\}
\end{align*}
\]

\[
\begin{align*}
t &\leq y \\
y &\leq x \\
t &\leq x \\
\text{and} \quad x &\leq z \\
z &\leq t \\
x &\leq t
\end{align*}
\]

**Cost:** linear for bounded-size packs

**Information loss:** no communication between packs!

**Solution:** intervals constraints sharing

Not good enough!
How to scale up: variable packing

**Principle:**
- split variables into packs
- use a DBM per pack

\[
P_1 = \{t, x, y\} \quad \text{and} \quad P_2 = \{t, x, z\}
\]
\[
t \leq y \quad \quad \quad \quad x \leq z
\]
\[
y \leq x \quad \quad \quad \quad z \leq t
\]
\[
t \leq x \quad \quad \quad \quad x \leq t
\]
\[
x = t
\]

**Cost:** linear for bounded-size packs

**Information loss:** no communication between packs!

**Solution:** intervals constraints sharing

Not good enough!
An idea: a subgraph

Goal: share relational constraints
An idea: a subgraph

Goal: share relational constraints
An idea: a subgraph

Goal: share relational constraints
An idea: a subgraph

Goal: share relational constraints

Issues: we need to keep
An idea: a subgraph

**Goal:** share relational constraints

**Issues:** we need to keep
- a good expressiveness
- a structure with packs
- precise and efficient algorithms
TreeKs: a certain subgraph

Shape:

- a tree of complete graphs (packs)
- sharing frontiers
TreeKs: a certain subgraph

Shape:

- a tree of complete graphs (packs)
- sharing frontiers
TreeKs: a certain subgraph

Shape:
- a tree of complete graphs (packs)
- sharing frontiers

Abstract value: tuple of DBMs
TreeKs: a certain subgraph

Shape:
- a tree of complete graphs (packs)
- sharing frontiers

Parameters:
- $N$: number of variables
- $m$: number of packs
- $p$: size of a pack
- $f$: size of a frontier
- $d$: diameter of the graph
TreeKs: abstract operators

On complete values, all operations can be done pointwisely:

▶ inclusion test
▶ intersection
▶ union

but constraint extraction and addition...
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

**foreach** pack *from the leaves to the root* **do**
  - Apply completion on this pack in the domain of zones
  - Pass the new constraints to its father

**foreach** pack *from the root to the leaves* **do**
  - Apply completion on this pack in the domain of zones
  - Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

**foreach** pack from the leaves to the root **do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its father

**foreach** pack from the root to the leaves **do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its children
Completion algorithm in TreeKs $O(mp^3)$

**foreach pack from the leaves to the root do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its father

**foreach pack from the root to the leaves do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

```plaintext
foreach pack from the leaves to the root do
    Apply completion on this pack in the domain of zones
    Pass the new constraints to its father

foreach pack from the root to the leaves do
    Apply completion on this pack in the domain of zones
    Pass the new constraints to its children
```

---

[Diagram showing completion process in TreeKs]
Completion algorithm in TreeKs $O(mp^3)$

**foreach** pack from the leaves to the root **do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its father

**foreach** pack from the root to the leaves **do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

**foreach pack from the leaves to the root do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its father

**foreach pack from the root to the leaves do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

`foreach pack from the leaves to the root do`
  Apply completion on this pack in the domain of zones
  Pass the new constraints to its father

`foreach pack from the root to the leaves do`
  Apply completion on this pack in the domain of zones
  Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

**foreach pack from the leaves to the root do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its father

**foreach pack from the root to the leaves do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs \(O(mp^3)\)

\[
\textbf{foreach pack from the leaves to the root do}
\]
\hspace{1cm} Apply completion on this pack in the domain of zones
\hspace{1cm} Pass the new constraints to its father

\[
\textbf{foreach pack from the root to the leaves do}
\]
\hspace{1cm} Apply completion on this pack in the domain of zones
\hspace{1cm} Pass the new constraints to its children
Completion algorithm in TreeKs $O(mp^3)$

**foreach pack from the leaves to the root do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its father

**foreach pack from the root to the leaves do**
- Apply completion on this pack in the domain of zones
- Pass the new constraints to its children
Completion algorithm

Completion algorithm in TreeKs $O(mp^3)$

\begin{itemize}
\item \textbf{foreach} pack from the leaves to the root \textbf{do}
  \begin{itemize}
  \item Apply completion on this pack in the domain of zones
  \item Pass the new constraints to its father
  \end{itemize}
\item \textbf{foreach} pack from the root to the leaves \textbf{do}
  \begin{itemize}
  \item Apply completion on this pack in the domain of zones
  \item Pass the new constraints to its children
  \end{itemize}
\end{itemize}
Constraint extraction

Goal: to bound $X_u - X_v$

Simple case: $X_u$ and $X_v$ are in the same pack
Constraint extraction

Goal: to bound $X_u - X_v$

Complex case: $X_u$ and $X_v$ are in different packs
Constraint extraction

**Goal**: to bound $X_u - X_v$

Complex case: $X_u$ and $X_v$ are in different packs

Only constraints in the path between $X_v$ and $X_u$ need to be considered
Adding constraints

**Goal**: to add the constraint $X_u - X_v \leq c$

**Simple case**: $X_u$ and $X_v$ are in the same pack
Adding constraints

**Goal**: to add the constraint $X_u - X_v \leq c$

**Complex case**: $X_u$ and $X_v$ are in different packs
Adding constraints

**Goal:** to add the constraint $X_u - X_v \leq c$

Complex case: $X_u$ and $X_v$ are in different packs

Only constraints in the path between $X_v$ and $X_u$ have to be updated
Summary

We proposed a new numerical abstract domain as a functor that:

▶ can be applied to many numerical abstract domains (zones, octagons, logahedra, TVPI, octahedra, polyhedra, ...)
▶ can be applied to other linear inequality domains to come
▶ with linear cost completion when pack size is bounded
▶ simple, precise, and efficient algorithms

Future work:

▶ application to other convex domains and non-convex domains
▶ development of packs generation strategies
▶ implementations are welcome!