Partitioning abstractions

MPRI — Cours 2.6 "Interprétation abstraite : application à la vérification et à l'analyse statique"

Xavier Rival

INRIA, ENS, CNRS

Dec, 13th. 2013

Towards disjunctive abstractions

- **disjunctions** are **often needed**...
- ... but **potentially costly**

In this lecture, we will discuss:

- **precision issues** that motivate the use of abstract domains able to **express disjunctions**
- **several ways** to **express disjunctions** using **abstract domain combiners**
  - disjunctive completion
  - cardinal power
  - state partitioning
  - trace partitioning

Domain combinators (or combiners)

General combination of abstract domains

- takes one or more abstract domains as **inputs**
- produces a **new abstract domain**

Input and output abstract domains are **characterized by an "interface"**: concrete domain, abstraction relation, abstract elements and operators

Advantages:

- **general definition**, formalized and proved once
- can be **implemented** in a separate way, e.g., in ML:
  - abstract domain: **module**
  - abstract domain combinator: **functor**

Example: product abstraction

For this example,

- we assume the concrete domain is $(\mathcal{P}(\mathbb{M}), \subseteq)$ where $\mathbb{M} = \mathbb{X} \to \mathbb{V}$
- we require an abstract domain $\mathbb{D}^\sharp$ to provide
  - a concretization function $\gamma : \mathbb{D}^\sharp \to \mathcal{P}(\mathbb{M})$
  - an element $\perp$ with empty concretization $\gamma(\perp) = \emptyset$

Product combinator

Given abstract domains $(\mathbb{D}^\sharp_0, \gamma_0, \perp_0)$ and $(\mathbb{D}^\sharp_1, \gamma_1, \perp_1)$, the **product abstraction** is $(\mathbb{D}^\sharp_\times, \gamma_\times, \perp_\times)$ where:

- $\mathbb{D}^\sharp_\times = \mathbb{D}^\sharp_0 \times \mathbb{D}^\sharp_1$
- $\gamma_\times(x^\sharp_0, x^\sharp_1) = \gamma_0(x^\sharp_0) \cap \gamma_1(x^\sharp_1)$
- $\perp_\times = (\perp_0, \perp_1)$

Example: product abstraction, coalescent product

The product abstraction **needs a reduction**:

$$\forall x_0^\sharp \in \mathbb{D}_0^\sharp, x_1^\sharp \in \mathbb{D}_1^\sharp,\ \gamma_\times(\perp_0, x_1^\sharp) = \gamma_\times(x_0^\sharp, \perp_1) = \emptyset = \gamma_\times(\perp_\times)$$

Coalescent product

Given abstract domains $(\mathbb{D}_0^\sharp, \gamma_0, \perp_0)$ and $(\mathbb{D}_1^\sharp, \gamma_1, \perp_1)$, the **coalescent product abstraction** is $(\mathbb{D}_\times^\sharp, \gamma_\times, \perp_\times)$ where:

- $\mathbb{D}_\times^\sharp = \{\perp_\times\} \uplus \{(x_0^\sharp, x_1^\sharp) \in \mathbb{D}_0^\sharp \times \mathbb{D}_1^\sharp \mid x_0^\sharp \neq \perp_0 \wedge x_1^\sharp \neq \perp_1\}$
- $\gamma_\times(\perp_\times) = \emptyset$, $\gamma_\times(x_0^\sharp, x_1^\sharp) = \gamma_0(x_0^\sharp) \cap \gamma_1(x_1^\sharp)$

In many cases, this is **not enough to achieve reduction**:

- let $\mathbb{D}_0^\sharp$ be the interval abstraction, $\mathbb{D}_1^\sharp$ be the congruences abstraction
- $\gamma_\times(\{\mathtt{x} \in [3,4]\}, \{\mathtt{x} \equiv 0 \mod 5\}) = \emptyset$

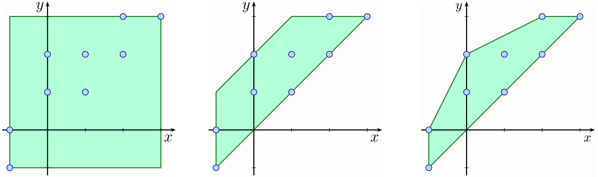
- how to define abstract domain combiners to **add disjunctions** ?

Outline

1. Introduction
2. Imprecisions in convex abstractions
3. Disjunctive completion
4. Cardinal power and partitioning abstractions
5. State partitioning
6. Trace partitioning
7. Conclusion

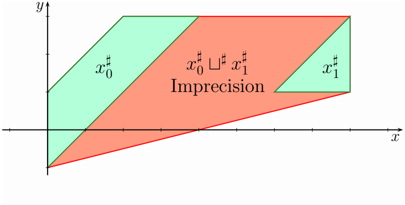
Convex abstractions

Many numerical abstractions describe **convex sets of points**



interval domain   octagon domain   polyedra domain

Imprecisions inherent in the **convexity:**



Such imprecisions may impact analysis results

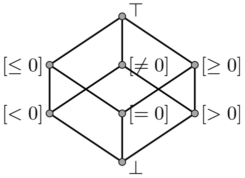
Non convex abstractions

We consider abstractions of $\mathbb{D} = \mathcal{P}(\mathbb{Z})$

Congruences:

- $\mathbb{D}^\sharp = \mathbb{Z} \times \mathbb{N}$
- $\gamma(n, k) = \{n + k \cdot p \mid p \in \mathbb{Z}\}$
- $-1, 1 \in \gamma(1, 2)$
  but $0 \notin \gamma(1, 2)$

Signs:



- $0 \notin \gamma([\neq 0])$

Example 1: verification problem

```
bool b0, b1;
int x, y;          (uninitialized)
b0 = x ≥ 0;
b1 = x ≤ 0;
if(b0 && b1){
      y = 0;
} else {
①     y = 100/x;
}
```

- if $\neg b_0$, then $x < 0$
- if $\neg b_1$, then $x > 0$
- if either b_0 or b_1 is false, then $x \neq 0$
- thus, if point ① is reached the division is safe

How to verify the division operation ?

- Non relational abstraction (e.g., intervals), at point ①:

$$\left\{ \begin{array}{lcl} b_0 & = & \mathtt{FALSE} \\ b_1 & = & \mathtt{FALSE} \\ x & : & \top \end{array} \right.$$

- Signs, congruences do not help:
  in the concrete, x may take any value but 0

Example 1: Hoare style program proof

bool $\mathrm{b_0}$, $\mathrm{b_1}$;
int x, y; (uninitialized)
$\mathrm{b_0} = \mathrm{x} \geq 0;$
$(\mathrm{b_0} \wedge \mathrm{x} \geq 0) \vee (\neg \mathrm{b_0} \wedge \mathrm{x} < 0)$
$\mathrm{b_1} = \mathrm{x} \leq 0;$
$(\mathrm{b_0} \wedge \mathrm{b_1} \wedge \mathrm{x} = 0) \vee (\mathrm{b_0} \wedge \neg \mathrm{b_1} \wedge \mathrm{x} > 0) \vee (\neg \mathrm{b_0} \wedge \mathrm{b_1} \wedge \mathrm{x} < 0)$
if($\mathrm{b_0}$ && $\mathrm{b_1}$){
$(\mathrm{b_0} \wedge \mathrm{b_1} \wedge \mathrm{x} = 0)$
$\mathrm{y} = 0;$
$(\mathrm{b_0} \wedge \mathrm{b_1} \wedge \mathrm{x} = 0 \wedge \mathrm{y} = 0)$
} **else** {
$(\mathrm{b_0} \wedge \neg \mathrm{b_1} \wedge \mathrm{x} > 0) \vee (\neg \mathrm{b_0} \wedge \mathrm{b_1} \wedge \mathrm{x} < 0)$
$\mathrm{y} = 100/\mathrm{x};$
$(\mathrm{b_0} \wedge \neg \mathrm{b_1} \wedge \mathrm{x} > 0) \vee (\neg \mathrm{b_0} \wedge \mathrm{b_1} \wedge \mathrm{x} < 0)$
}

We need to **add disjunctions** to our abstract domain

Example 2: verification problem

```
    int x ∈ ℤ;
    int s;
    int y;
    if(x ≥ 0){
        s = 1;
    } else {
        s = −1;
    }
①   y = x/s;
②   assert(y ≥ 0);
```

- s is either 1 or -1
- thus, the division at ① should not fail
- moreover s has the same sign as x
- thus, the value stored in y should always be positive at ②

- **How to verify the division operation ?**
- In the concrete, s is **always non null**:
  convex abstractions cannot establish this; congruences can
- Moreover, s has always the **same sign** as x
  expressing this would require a fairly complex numerical abstraction

Example 2: Hoare style program proof

$\textbf{int}\ \mathrm{x} \in \mathbb{Z};$
$\textbf{int}\ \mathrm{s};$
$\textbf{int}\ \mathrm{y};$
$\textbf{if}(\mathrm{x} \geq 0)\{$
$\qquad (\mathrm{x} \geq 0)$
$\quad \mathrm{s} = 1;$
$\qquad (\mathrm{x} \geq 0 \wedge \mathrm{s} = 1)$
$\}\ \textbf{else}\ \{$
$\qquad (\mathrm{x} < 0)$
$\quad \mathrm{s} = -1;$
$\qquad (\mathrm{x} < 0 \wedge \mathrm{s} = -1)$
$\}$
$\qquad (\mathrm{x} \geq 0 \wedge \mathrm{s} = 1) \vee (\mathrm{x} < 0 \wedge \mathrm{s} = -1)$
① $\mathrm{y} = \mathrm{x}/\mathrm{s};$
$\qquad (\mathrm{x} \geq 0 \wedge \mathrm{s} = 1 \wedge \mathrm{y} \geq 0) \vee (\mathrm{x} < 0 \wedge \mathrm{s} = -1 \wedge \mathrm{y} > 0)$
② $\textbf{assert}(\mathrm{y} \geq 0);$

We need to **add disjunctions** to our abstract domain

Outline

1 Introduction

2 Imprecisions in convex abstractions

3 Disjunctive completion

4 Cardinal power and partitioning abstractions

5 State partitioning

6 Trace partitioning

7 Conclusion

Distributive abstract domain

Principle:

1. consider concrete domain $(\mathbb{D}, \sqsubseteq)$, with lower upper bound operator $\sqcup$
2. start with an abstract domain $(\mathbb{D}^\sharp, \sqsubseteq^\sharp)$ with concretization $\gamma : \mathbb{D}^\sharp \to \mathbb{D}$
3. build a domain containing **all the disjunctions** of elements of $\mathbb{D}^\sharp$

Definition: distributive abstract domain

Abstract domain $(\mathbb{D}^\sharp, \sqsubseteq^\sharp)$ with concretization function $\gamma : \mathbb{D}^\sharp \to \mathbb{D}$ is **distributive** (or **complete for disjunction**) if and only if:

$$\forall \mathcal{E} \subseteq \mathbb{D}^\sharp,\ \exists x^\sharp \in \mathbb{D}^\sharp,\ \gamma(x^\sharp) = \bigsqcup_{y^\sharp \in \mathcal{E}} \gamma(y^\sharp)$$

Examples:

- the lattice $\{\bot, < 0, = 0, > 0, \leq 0, \neq 0, \geq 0, \top\}$ is distributive
- the lattice of intervals is not distributive:
  there is no interval with concretization $\gamma([0, 10]) \cup \gamma([12, 20])$

Definition

Definition: disjunctive completion

The disjunctive completion of abstract domain $(\mathbb{D}^\sharp, \sqsubseteq^\sharp)$ with concretization function $\gamma : \mathbb{D}^\sharp \to \mathbb{D}$ is the smallest abstract domain $(\mathbb{D}^\sharp_\vee, \sqsubseteq^\sharp_\vee)$ with concretization function $\gamma_\vee : \mathbb{D}^\sharp_\vee \to \mathbb{D}$ such that:

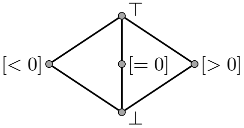
- $\mathbb{D}^\sharp \subseteq \mathbb{D}^\sharp_\vee$
- $\forall x^\sharp \in \mathbb{D}^\sharp,\ \gamma_\vee(x^\sharp) = \gamma(x^\sharp)$
- $(\mathbb{D}^\sharp_\vee, \sqsubseteq^\sharp_\vee)$ with concretization $\gamma_\vee$ is distributive

Building a disjunctive completion domain:

- start with $\mathbb{D}^\sharp_\vee = \mathbb{D}^\sharp$
- for all set $\mathcal{E} \subseteq \mathbb{D}^\sharp$ such that there is no $x^\sharp \in \mathbb{D}^\sharp$, such that $\gamma(x^\sharp) = \bigsqcup_{y^\sharp \in \mathcal{E}} \gamma(y^\sharp)$, add $\mathcal{E}$ to $\mathbb{D}^\sharp_\vee$, and extend $\gamma_\vee$ by
$$\gamma_\vee(\mathcal{E}) = \bigsqcup_{y^\sharp \in \mathcal{E}} \gamma(y^\sharp)$$

Example 1: completion of signs

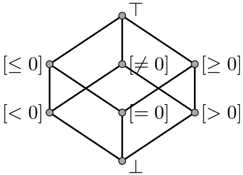
We consider concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$ and $(\mathbb{D}^\sharp, \sqsubseteq)$ be defined by:



$$\begin{array}{rlcl} \gamma : & \bot & \longmapsto & \emptyset \\ & [<0] & \longmapsto & \{k \in \mathbb{Z} \mid k < 0\} \\ & [=0] & \longmapsto & \{k \in \mathbb{Z} \mid k = 0\} \\ & [>0] & \longmapsto & \{k \in \mathbb{Z} \mid k > 0\} \\ & \top & \longmapsto & \mathbb{Z} \end{array}$$

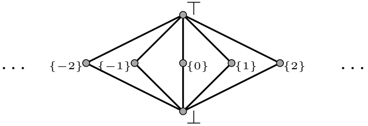
Then, the disjunctive completion is defined by adding elements corresponding to:

- $\{[<0], [=0]\}$
- $\{[<0], [>0]\}$
- $\{[=0], [>0]\}$



Example 2: completion of constants

We consider concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$ and $(\mathbb{D}^\sharp, \sqsubseteq)$ be defined by:



$$\gamma : \begin{array}{lcl} \bot & \longmapsto & \emptyset \\ \{k\} & \longmapsto & \{k\} \\ \top & \longmapsto & \mathbb{Z} \end{array}$$

Then, the disjunctive completion is the powerset:

- $\mathbb{D}^\sharp_\vee \equiv \mathcal{P}(\mathbb{Z})$
- $\gamma_\vee$ is the **identity function !**
- this lattice contains **infinite sets which are not representable**

Example 3: completion of intervals

We consider concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$ and $(\mathbb{D}^\sharp, \sqsubseteq)$ be the domain of intervals

- $\mathbb{D}^\sharp = \{\bot, \top\} \uplus \{[a, b] \mid a \leq b\}$
- $\gamma([a, b]) = [a, b]$

Then, the disjunctive completion is the set of **unions of intervals** :

- $\mathbb{D}^\sharp_\vee$ collects all the families of disjoint intervals
- this lattice contains **infinite sets which are not representable**

The disjunctive completion of $(\mathbb{D}^\sharp)^n$ is **not equivalent** to $(\mathbb{D}^\sharp_\vee)^n$

Example 3: completion of intervals and verification

We use the disjunctive completion of $(\mathbb{D}^\sharp)^3$.
The invariants below can be expressed in the disjunctive completion:

$$
\begin{array}{l}
\textbf{int}\ \mathrm{x} \in \mathbb{Z};\\
\textbf{int}\ \mathrm{s};\\
\textbf{int}\ \mathrm{y};\\
\textbf{if}(\mathrm{x} \geq 0)\{\\
\qquad\qquad (\mathrm{x} \geq 0)\\
\qquad \mathrm{s} = 1;\\
\qquad\qquad (\mathrm{x} \geq 0 \wedge \mathrm{s} = 1)\\
\}\ \textbf{else}\ \{\\
\qquad\qquad (\mathrm{x} < 0)\\
\qquad \mathrm{s} = -1;\\
\qquad\qquad (\mathrm{x} < 0 \wedge \mathrm{s} = -1)\\
\}\\
\qquad\qquad (\mathrm{x} \geq 0 \wedge \mathrm{s} = 1) \vee (\mathrm{x} < 0 \wedge \mathrm{s} = -1)\\
\mathrm{y} = \mathrm{x}/\mathrm{s};\\
\qquad\qquad (\mathrm{x} \geq 0 \wedge \mathrm{s} = 1 \wedge \mathrm{y} \geq 0) \vee (\mathrm{x} < 0 \wedge \mathrm{s} = -1 \wedge \mathrm{y} > 0)\\
\textbf{assert}(\mathrm{y} \geq 0);
\end{array}
$$

Limitations of disjunctive completion

- **Combinatorial explosion**:
  - if $\mathbb{D}^\sharp$ is infinite, $\mathbb{D}^\sharp_\vee$ may have elements that **cannot be represented**
  - even when $\mathbb{D}^\sharp$ is finite, $\mathbb{D}^\sharp_\vee$ may be **huge**
    in the worst case, if $\mathbb{D}^\sharp$ has n elements, $\mathbb{D}^\sharp_\vee$ may have 2^n elements
- **Many elements useless in practice**:
  disjunctive completion of intervals: may express any set of integers...
- **No general definition of a widening operator**
  - most common approach: bound the numbers of disjuncts
    i.e., the size of the sets added to the base domain

Outline

1. Introduction
2. Imprecisions in convex abstractions
3. Disjunctive completion
4. Cardinal power and partitioning abstractions
5. State partitioning
6. Trace partitioning
7. Conclusion

Principle

- disjuncts can usually be **characterized** by some property
  for instance:
  - **sign** of a variable
  - **value** of a **boolean** variable
  - **execution path**, e.g., side of a condition that was visited
- **solution**: perform a kind of **indexing** of disjuncts
  - use an abstraction to **describe labels**
    e.g., sign of a variable, value of a boolean, or trace property...
  - apply the abstraction that needs be completed on the images

Cardinal power abstraction

Definition

We assume $(\mathbb{D}, \sqsubseteq) = (\mathcal{P}(\mathcal{E}), \subseteq)$, and that two abstractions $(\mathbb{D}_0^\sharp, \sqsubseteq_0^\sharp), (\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp)$ given by their concretization functions:

$$\gamma_0 : \mathbb{D}_0^\sharp \longrightarrow \mathbb{D} \qquad \gamma_1 : \mathbb{D}_1^\sharp \longrightarrow \mathbb{D}$$

We let the **cardinal power abstract domain** be defined by:

- $\mathbb{D}_\rightarrow^\sharp = \mathbb{D}_0^\sharp \stackrel{\mathcal{M}}{\rightarrow} \mathbb{D}_1^\sharp$ be the set of monotone functions from $\mathbb{D}_0^\sharp$ into $\mathbb{D}_1^\sharp$
- $\sqsubseteq_\rightarrow^\sharp$ be the pointwise extension of $\sqsubseteq_1^\sharp$
- $\gamma_\rightarrow$ is defined by:

$$\begin{array}{rccl} \gamma_\rightarrow : & \mathbb{D}_\rightarrow^\sharp & \longrightarrow & \mathbb{D} \\ & X^\sharp & \longmapsto & \{y \in \mathcal{E} \mid \forall z^\sharp \in \mathbb{D}_0^\sharp,\, y \in \gamma_0(z^\sharp) \Longrightarrow y \in \gamma_1(X^\sharp(z^\sharp))\} \end{array}$$

We sometimes denote it by $\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_1^\sharp$, $\gamma_{\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_1^\sharp}$.

Use of cardinal power abstractions

Intuition: we can express properties of the form

$$\left\{\begin{array}{cccc} & p_0 & \Longrightarrow & p_0' \\ \wedge & p_1 & \Longrightarrow & p_1' \\ \vdots & \vdots & \vdots & \vdots \\ \wedge & p_n & \Longrightarrow & p_n' \end{array}\right.$$

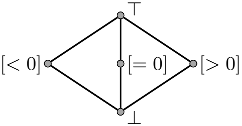
Two independent choices:

1. $\mathbb{D}_0^\sharp$: set of partitions (the "labels")
2. $\mathbb{D}_1^\sharp$: abstraction of sets of states, e.g., a numerical abstraction

Example

We consider:

- concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$
- $(\mathbb{D}^\sharp_0, \sqsubseteq^\sharp_0)$ be the **lattice of signs** (strict values only)
- $(\mathbb{D}^\sharp_1, \sqsubseteq^\sharp_1)$ be the **lattice of intervals**

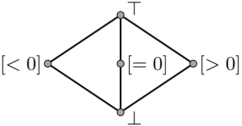


A few example abstract values:

- $[0, 8]$ is expressed by: $\left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [< 0] & \longmapsto & \bot_1 \\ [= 0] & \longmapsto & [0, 0] \\ [> 0] & \longmapsto & [1, 8] \\ \top_0 & \longmapsto & [0, 8] \end{array} \right.$
- $[-10, -3] \uplus [7, 10]$ is expressed by: $\left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [< 0] & \longmapsto & [-10, -3] \\ [= 0] & \longmapsto & \bot_1 \\ [> 0] & \longmapsto & [7, 10] \\ \top_0 & \longmapsto & [-10, 10] \end{array} \right.$

Reduction (1): tightening disjunctions

- concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$
- $(\mathbb{D}^\sharp_0, \sqsubseteq^\sharp_0)$ be the lattice of signs
- $(\mathbb{D}^\sharp_1, \sqsubseteq^\sharp_1)$ be the lattice of intervals



We let: $$X^\sharp = \left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [< 0] & \longmapsto & [-5, -1] \\ [= 0] & \longmapsto & [0, 0] \\ [> 0] & \longmapsto & [1, 5] \\ \top_0 & \longmapsto & [-10, 10] \end{array} \right. \qquad Y^\sharp = \left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [< 0] & \longmapsto & [-5, -1] \\ [= 0] & \longmapsto & [0, 0] \\ [> 0] & \longmapsto & [1, 5] \\ \top_0 & \longmapsto & [-5, 5] \end{array} \right.$$

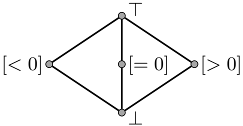
- Then, $\gamma_\rightarrow(X^\sharp) = \gamma_\rightarrow(Y^\sharp)$
- $\gamma_0([< 0]) \cup \gamma_0([= 0]) \cup \gamma([> 0]) = \gamma(\top_0)$
  but $\gamma_0(X^\sharp([< 0])) \cup \gamma_0(X^\sharp([= 0])) \cup \gamma(X^\sharp([> 0])) \subset \gamma(X^\sharp(\top_0))$

Tightening of mapping $(\sqcup\{z^\sharp \mid z^\sharp \in \mathcal{E}\}) \mapsto x^\sharp_1$

- $\bigcup\{\gamma_0(z^\sharp) \mid z^\sharp \in \mathcal{E}\} = \gamma_0(\sqcup\{z^\sharp \mid z^\sharp \in \mathcal{E}\})$
- $\exists y^\sharp,\ \bigcup\{\gamma_1(X^\sharp(z^\sharp)) \mid z^\sharp \in \mathcal{E}\} \subseteq \gamma_1(y^\sharp) \subset \gamma_1(X^\sharp(\sqcup\{z^\sharp \mid z^\sharp \in \mathcal{E}\}))$

Reduction (2): relation between the two domains

- concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$
- $(\mathbb{D}^\sharp_0, \sqsubseteq^\sharp_0)$ be the lattice of signs
- $(\mathbb{D}^\sharp_1, \sqsubseteq^\sharp_1)$ be the lattice of intervals



We let:

$$X^\sharp = \left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [<0] & \longmapsto & [1,8] \\ [=0] & \longmapsto & [1,8] \\ [>0] & \longmapsto & \bot_1 \\ \top_0 & \longmapsto & [1,8] \end{array} \right. \quad Y^\sharp = \left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [<0] & \longmapsto & [2,45] \\ [=0] & \longmapsto & [-5,-2] \\ [>0] & \longmapsto & [-5,-2] \\ \top_0 & \longmapsto & \top_1 \end{array} \right. \quad Z^\sharp = \left\{ \begin{array}{lcl} \bot_0 & \longmapsto & \bot_1 \\ [<0] & \longmapsto & \bot_1 \\ [=0] & \longmapsto & \bot_1 \\ [>0] & \longmapsto & \bot_1 \\ \top_0 & \longmapsto & \bot_1 \end{array} \right.$$

Then, $\gamma_\rightarrow(X^\sharp) = \gamma_\rightarrow(Y^\sharp) = \gamma_\rightarrow(Z^\sharp) = \emptyset$

Relation between $\mathbb{D}^\sharp_0$ elements and $\mathbb{D}^\sharp_1$ elements

Binding $y^\sharp_0 \mapsto y^\sharp_1$ can be improved if $\exists z^\sharp_1 \neq y^\sharp_1,\ \gamma(y^\sharp_1) \cap \gamma(y^\sharp_0) \subseteq \gamma(z^\sharp_1)$

More compact representation of the cardinal power

- if $\mathbb{D}_0^\sharp$ has N elements, then an abstract value in $\mathbb{D}_\rightarrow^\sharp$ requires N elements of $\mathbb{D}_1^\sharp$
- if $\mathbb{D}_0^\sharp$ is infinite, and $\mathbb{D}_1^\sharp$ is non trivial, then $\mathbb{D}_\rightarrow^\sharp$ has elements that cannot be represented
- the 1st reduction shows it is unnecessary to represent bindings for all elements of $\mathbb{D}_0^\sharp$
- the 2nd reduction shows it is unnecessary to represent a binding for $\perp_0$

Compact representation

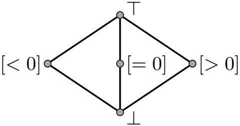
Reduced cardinal power of $\mathbb{D}_0^\sharp$ and $\mathbb{D}_1^\sharp$ can be represented by considering only a subset $\mathcal{C} \subseteq \mathbb{D}_0^\sharp$ where

$$\forall x^\sharp \in \mathbb{D}_0^\sharp,\ \exists \mathcal{E} \subseteq \mathcal{C},\ \gamma_0(x^\sharp) = \cup\{\gamma_0(y^\sharp) \mid y^\sharp \in \mathcal{E}\}$$

- in particular, we should let $\perp_0 \notin \mathcal{C}$

Example: compact cardinal power over signs

- concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$
- $(\mathbb{D}^\sharp_0, \sqsubseteq^\sharp_0)$ be the **lattice of signs**
- $(\mathbb{D}^\sharp_1, \sqsubseteq^\sharp_1)$ be the **lattice of intervals**



We remark that:

- $\bot_0$ does not need be considered
- $\gamma_0([< 0]) \cup \gamma_0([= 0]) \cup \gamma([> 0]) = \gamma(\top_0)$ thus $\top_0$ does not need be considered

Thus, we let $\mathcal{C} = \{[< 0], [= 0], [> 0]\}$; then:

- $[0, 8]$ is expressed by: $\left\{ \begin{array}{lcl} [< 0] & \longmapsto & \bot_1 \\ [= 0] & \longmapsto & [0, 0] \\ [> 0] & \longmapsto & [1, 8] \end{array} \right.$
- $[-10, -3] \uplus [7, 10]$ is expressed by: $\left\{ \begin{array}{lcl} [< 0] & \longmapsto & [-10, -3] \\ [= 0] & \longmapsto & \bot_1 \\ [> 0] & \longmapsto & [7, 10] \end{array} \right.$

Lattice operations

Infimum:

- we assume that $\bot_1$ is the infimum of $\mathbb{D}_1^\sharp$
- then, $\bot_\rightarrow = \lambda(z^\sharp \in \mathbb{D}_0^\sharp) \cdot \bot_1$ is the **infimum** of $\mathbb{D}_\rightarrow^\sharp$

Ordering:

- we let $\sqsubseteq_\rightarrow^\sharp$ denote the **pointwise ordering**:

$$X_0^\sharp \sqsubseteq_\rightarrow^\sharp X_1^\sharp \quad \stackrel{def}{\Longleftrightarrow} \quad \forall z^\sharp \in \mathbb{D}_0^\sharp,\ X_0^\sharp(z^\sharp) \sqsubseteq_1^\sharp X_1^\sharp(z^\sharp)$$

- then, $X_0^\sharp \sqsubseteq_\rightarrow^\sharp X_1^\sharp \Longrightarrow \gamma_\rightarrow(X_0^\sharp) \subseteq \gamma_\rightarrow(X_1^\sharp)$

Join operation:

- we assume that $\sqcup_1$ is a sound upper bound operator in $\mathbb{D}_1^\sharp$
- then, $\sqcup_\rightarrow$ defined below is a **sound upper bound operator** in $\mathbb{D}_\rightarrow^\sharp$:

$$X_0^\sharp \sqcup_\rightarrow X_1^\sharp \quad \stackrel{def}{::=} \quad \lambda(z^\sharp \in \mathbb{D}_0^\sharp) \cdot (X_0^\sharp(z^\sharp) \sqcup_1 X_1^\sharp(z^\sharp))$$

- the same construction applies to widening, if $\mathbb{D}_0^\sharp$ is finite

Composition with another abstraction

We assume three abstractions

- $(\mathbb{D}_0^\sharp, \sqsubseteq_0^\sharp)$, with concretization $\gamma_0 : \mathbb{D}_0^\sharp \longrightarrow \mathbb{D}$
- $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp)$, with concretization $\gamma_1 : \mathbb{D}_1^\sharp \longrightarrow \mathbb{D}$
- $(\mathbb{D}_2^\sharp, \sqsubseteq_2^\sharp)$, with concretization $\gamma_2 : \mathbb{D}_2^\sharp \longrightarrow \mathbb{D}_1^\sharp$

Cardinal power abstract domains $\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_1^\sharp$ and $\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_2^\sharp$ can be bound by an **abstraction relation** defined by concretization function γ:

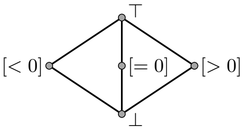
$$
\begin{array}{rlcl}
\gamma : & (\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_2^\sharp) & \longrightarrow & (\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_1^\sharp) \\
 & X^\sharp & \longmapsto & \lambda(z^\sharp \in \mathbb{D}_0^\sharp) \cdot \gamma(X^\sharp(z^\sharp))
\end{array}
$$

Applications:

- start with $\mathbb{D}_1^\sharp$ as the **identity abstraction**
- **compose several** cardinal power abstractions
  (or partitioning abstractions)

Composition with another abstraction

- concrete lattice $\mathbb{D} = \mathcal{P}(\mathbb{Z})$, with $\sqsubseteq = \subseteq$
- $(\mathbb{D}_0^\sharp, \sqsubseteq_0^\sharp)$ be the lattice of signs
- $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp)$ be the identity abstraction
  $\mathbb{D}_1^\sharp = \mathcal{P}(\mathbb{Z})$, $\gamma_1 = \mathbf{Id}$
- $(\mathbb{D}_2^\sharp, \sqsubseteq_2^\sharp)$ be the lattice of intervals



Then, $[-10, -3] \uplus [7, 10]$ is abstracted in two steps:

- in $\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_1^\sharp$, $\left\{ \begin{array}{lcl} [<0] & \longmapsto & [-10,-3] \\ [=0] & \longmapsto & \emptyset \\ [>0] & \longmapsto & [7,10] \end{array} \right.$
- in $\mathbb{D}_0^\sharp \rightrightarrows \mathbb{D}_2^\sharp$, $\left\{ \begin{array}{lcl} [<0] & \longmapsto & [-10,-3] \\ [=0] & \longmapsto & \perp_1 \\ [>0] & \longmapsto & [7,10] \end{array} \right.$

Outline

1 Introduction

2 Imprecisions in convex abstractions

3 Disjunctive completion

4 Cardinal power and partitioning abstractions

5 State partitioning
- Definition and examples
- Control states partitioning and iteration techniques
- Abstract interpretation with boolean partitioning

6 Trace partitioning

7 Conclusion

Definition

We consider **concrete domain** $\mathbb{D} = \mathcal{P}(\mathbb{S})$ where

- $\mathbb{S} = \mathbb{L} \times \mathbb{M}$
- $\mathbb{M} = \mathbb{X} \longrightarrow \mathbb{V}$

State partitioning

A **state partitioning** abstraction is defined as the cardinal power of two abstractions $(\mathbb{D}_0^\sharp, \sqsubseteq_0^\sharp, \gamma_0)$ and $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp, \gamma_1)$ of sets of states:

- $(\mathbb{D}_0^\sharp, \sqsubseteq_0^\sharp, \gamma_0)$ defines the **partitions**
- $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp, \gamma_1)$ defines the **abstraction of each element of partitions**

- either $\mathbb{D}_1^\sharp = \mathcal{P}(\mathbb{S})$, ordered with the inclusion
- or an abstraction of sets of memory states: numerical abstraction can be obtained by composing another abstraction on top of $(\mathcal{P}(\mathbb{S}), \subseteq)$

Instantiation with a partition

We consider a partition $\mathcal{E}$ of $\mathcal{P}(\mathbb{S})$:

$$\forall e, e' \in \mathcal{E},\ e \neq e' \Longrightarrow e \cap e' = \emptyset$$
$$\mathbb{S} = \bigcup \mathcal{E}$$

It induces the **partitioning abstraction**

$$\mathbb{D}_0^\sharp = \mathcal{E}$$
$$\gamma_0 : e \mapsto e$$

Application 1: flow sensitive abstraction

Principle: abstract separately the states at distinct control states

Flow sensitive abstraction

We apply the cardinal power based partitioning abstraction with:

- $\mathbb{D}_0^\sharp = \mathbb{L}$
- $\gamma_0 : \ell \mapsto \{\ell\} \times \mathbb{M}$

It is induced by partition $\{\{\ell\} \times \mathbb{M} \mid \ell \in \mathbb{L}\}$

Then, if $X^\sharp$ is an element of the reduced cardinal power,

$$\begin{array}{rcl}\gamma_\rightarrow(X^\sharp) & = & \{s \in \mathbb{S} \mid \forall x \in \mathbb{D}_0^\sharp,\ s \in \gamma_0(x) \Longrightarrow s \in \gamma_1(X^\sharp(x))\} \\ & = & \{(l, m) \in \mathbb{S} \mid m \in \gamma_1(X^\sharp(l))\}\end{array}$$

- after this abstraction step, $\mathbb{D}_1^\sharp$ may simply represent sets of memory states (numeric abstractions...)
- this abstraction step is *very common* as part of the design of abstract interpreters

Application 1: flow insensitive abstraction

- representing one set of memory states per program point may be costly for some applications (e.g., compilation)
- **context insensitive** abstraction simply **forgets about control states**

Flow sensitive abstraction

We apply the cardinal power based partitioning abstraction with:

- $\mathbb{D}_0^\sharp = \{\cdot\}$
- $\gamma_0 : \cdot \mapsto \mathbb{S}$
- $\mathbb{D}_1^\sharp = \mathcal{P}(\mathbb{M})$
- $\gamma_1 : M \mapsto \{(\ell, m) \mid \ell \in \mathbb{L}, m \in M\}$

It is induced by a trivial partition of $\mathcal{P}(\mathbb{S})$

- used for some ultra-fast pointer analyses
  (very quick analyses used for, e.g., compiler optimization)

Application 2: context sensitive abstraction

- We consider the language **with procedures**
- Thus, $\mathbb{S} = \mathbb{K} \times \mathbb{L} \times \mathbb{M}$, where $\mathbb{K}$ is the set of **call strings**

$$
\begin{array}{rcll}
\kappa & \in & \mathbb{K} & \text{calling contexts} \\
\kappa & ::= & \epsilon & \text{empty call stack} \\
 & | & (f, \ell) \cdot \kappa & \text{call to } f \text{ from stack } \kappa \text{ at point } \ell
\end{array}
$$

- We assume that inside each function, we use the **flow sensitive** abstraction

Application 2: context sensitive abstraction

Various level of sensitivity can be defined by partitioning:

Fully context sensitive abstraction (∞-CFA)

- $\mathbb{D}_0^\sharp = \mathbb{K} \times \mathbb{L}$
- $\gamma_0 : (\kappa, \ell) \mapsto \{(\kappa, \ell, m) \mid m \in \mathbb{M}\}$

Partially context sensitive abstraction (k-CFA)

- $\mathbb{D}_0^\sharp = \{\kappa \in \mathbb{K} \mid \mathbf{length}(\kappa) \leq k\} \times \mathbb{L}$
- $\gamma_0 : (\kappa, \ell) \mapsto \{(\kappa \cdot \kappa', \ell, m) \mid \kappa' \in \mathbb{K}, m \in \mathbb{M}\}$

Non context sensitive abstraction (0-CFA)

- $\mathbb{D}_0^\sharp = \mathbb{L}$
- $\gamma_0 : \ell \mapsto \{(\kappa, \ell, m) \mid \kappa \in \mathbb{K}, m \in \mathbb{M}\}$

Application 2: context sensitive abstraction

∞-**CFA**:

- one invariant per calling context
- **very precise** (used, e.g., in Astrée)
- **infinite in presence of recursion** (i.e., not practical in this case)

0-**CFA**:

- merges **all** calling contexts to a same procedure
- **very coarse** abstraction
- but **usually quite efficient to compute**

k-**CFA**:

- usually **intermediate** level of precision and efficiency
- can be applied to programs with **recursive procedures**

Application 3: partitioning by a boolean condition

- so far, we only used abstractions of the context to partition
- we now consider abstractions of memory states properties

Function guided memory states partitioning

We let:

- $\mathbb{D}_0^\sharp = \mathcal{P}(A)$ for some set A, and $\phi : \mathbb{M} \to A$
- γ_0 be of the form $(x^\sharp \in \mathbb{D}_0^\sharp) \mapsto \{(\ell, m) \in \mathbb{S} \mid \phi(m) \in x^\sharp\}$

Many choices of functions are possible:

- **value** of one or several variables (boolean or scalar)
- **sign** of a variable
- ...

Application 3: partitioning by a boolean condition

We assume:

- $\mathbb{X} = \mathbb{X}_{\rm bool} \uplus \mathbb{X}_{\rm int}$, where $\mathbb{X}_{\rm bool}$ (resp., $\mathbb{X}_{\rm int}$) collects **boolean** (resp., **integer**) variables
- $\mathbb{X}_{\rm bool} = \{\mathtt{b}_0, \ldots, \mathtt{b}_{k-1}\}$
- $\mathbb{X}_{\rm int} = \{\mathtt{x}_0, \ldots, \mathtt{x}_{l-1}\}$

Thus, $\mathbb{M} = \mathbb{X} \rightarrow \mathbb{V} \equiv (\mathbb{X}_{\rm bool} \rightarrow \mathbb{V}_{\rm bool}) \times (\mathbb{X}_{\rm int} \rightarrow \mathbb{V}_{\rm int}) \equiv \mathbb{V}_{\rm bool}^k \times \mathbb{V}_{\rm int}^l$

Boolean partitioning abstract domain

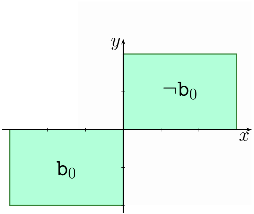
We apply the cardinal power abstraction, with a domain of partition defined by a function, with:

- $A = \mathbb{B}^k$
- $\phi(m) = (m(\mathtt{b}_0), \ldots, m(\mathtt{b}_{k-1}))$
- $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp, \gamma_1)$ an abstraction of $\mathcal{P}(\mathbb{V}_{\rm int}^l)$

Application 3: example

With $\mathbb{X}_{\mathrm{bool}} = \{\mathtt{b}_0, \mathtt{b}_1\}, \mathbb{X}_{\mathrm{int}} = \{\mathtt{x}, \mathtt{y}\}$, we can express:

$$\begin{cases} \mathtt{b}_0 \wedge \mathtt{b}_1 & \Longrightarrow \quad \mathtt{x}_0 \in [-3, 0] \wedge \mathtt{y} \in [0, 2] \\ \mathtt{b}_0 \wedge \neg\mathtt{b}_1 & \Longrightarrow \quad \mathtt{x}_0 \in [-3, 0] \wedge \mathtt{y} \in [0, 2] \\ \neg\mathtt{b}_0 \wedge \mathtt{b}_1 & \Longrightarrow \quad \mathtt{x}_0 \in [0, 3] \wedge \mathtt{y} \in [-2, 0] \\ \neg\mathtt{b}_0 \wedge \neg\mathtt{b}_1 & \Longrightarrow \quad \mathtt{x}_0 \in [0, 3] \wedge \mathtt{y} \in [-2, 0] \end{cases}$$



- this abstract value expresses a **relation** between $\mathtt{b}_0$ and $\mathtt{x}, \mathtt{y}$ (which induces a relation between $\mathtt{x}$ and $\mathtt{y}$)
- **alternative**: partition with respect to only **some** variables

Application 3: example

- Left side abstraction shown in blue: boolean partitioning for $\mathtt{b}_0, \mathtt{b}_1$
- Right side abstraction shown in green: interval abstraction

bool $\mathtt{b}_0$, $\mathtt{b}_1$;
int $\mathtt{x}$, $\mathtt{y}$; (uninitialized)
$\mathtt{b}_0 = \mathtt{x} \geq 0$;
$$(\mathtt{b}_0 \Longrightarrow \mathtt{x} \geq 0) \wedge (\neg \mathtt{b}_0 \Longrightarrow \mathtt{x} < 0)$$
$\mathtt{b}_1 = \mathtt{x} \leq 0$;
$$(\mathtt{b}_0 \wedge \mathtt{b}_1 \Longrightarrow \mathtt{x} = 0) \wedge (\mathtt{b}_0 \wedge \neg \mathtt{b}_1 \Longrightarrow \mathtt{x} > 0) \wedge (\neg \mathtt{b}_0 \wedge \mathtt{b}_1 \Longrightarrow \mathtt{x} < 0)$$
if($\mathtt{b}_0$ && $\mathtt{b}_1$){
$$(\mathtt{b}_0 \wedge \mathtt{b}_1 \Longrightarrow \mathtt{x} = 0)$$
$\mathtt{y} = 0$;
$$(\mathtt{b}_0 \wedge \mathtt{b}_1 \Longrightarrow \mathtt{x} = 0 \wedge \mathtt{y} = 0)$$
}**else**{
$$(\mathtt{b}_0 \wedge \neg \mathtt{b}_1 \Longrightarrow \mathtt{x} > 0) \wedge (\neg \mathtt{b}_0 \wedge \mathtt{b}_1 \Longrightarrow \mathtt{x} < 0)$$
$\mathtt{y} = 100/\mathtt{x}$;
$$(\mathtt{b}_0 \wedge \neg \mathtt{b}_1 \Longrightarrow \mathtt{x} > 0 \wedge \mathtt{y} \geq 0) \wedge (\neg \mathtt{b}_0 \wedge \mathtt{b}_1 \Longrightarrow \mathtt{x} < 0 \wedge \mathtt{y} \leq 0)$$
}

Application 3: partitioning by the sign of a variable

We assume:

- $\mathbb{X} = \mathbb{X}_{\mathrm{int}}$, i.e., all variables have **integer** type
- $\mathbb{X}_{\mathrm{int}} = \{\mathtt{x}_0, \ldots, \mathtt{x}_{l-1}\}$

Thus, $\mathbb{M} = \mathbb{X} \to \mathbb{V} \equiv \mathbb{V}_{\mathrm{int}}^{l}$

Sign partitioning abstract domain

We apply the cardinal power abstraction, with a domain of partition defined by a function, with:

- $A = \{[<0], [=0], [>0]\}$
- $\phi(m) = \begin{cases} [<0] & \text{if } x_0 < 0 \\ [=0] & \text{if } x_0 = 0 \\ [>0] & \text{if } x_0 > 0 \end{cases}$
- $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp, \gamma_1)$ an abstraction of $\mathcal{P}(\mathbb{V}_{\mathrm{int}}^{l-1})$ (no need to abstract $\mathtt{x}_0$ twice)

Application 3: example

- Abstraction fixing partitions shown in blue
- Right side abstraction shown in green: interval abstraction

int $\mathrm{x} \in \mathbb{Z}$;
int s;
int y;
if$(\mathrm{x} \geq 0)\{$
$(\mathrm{x} < 0 \Rightarrow \bot) \wedge (\mathrm{x} = 0 \Rightarrow \top) \wedge (\mathrm{x} > 0 \Rightarrow \top)$
$\mathrm{s} = 1;$
$(\mathrm{x} < 0 \Rightarrow \bot) \wedge (\mathrm{x} = 0 \Rightarrow \mathrm{s} = 1) \wedge (\mathrm{x} > 0 \Rightarrow \mathrm{s} = 1)$
$\}$ **else** $\{$
$(\mathrm{x} < 0 \Rightarrow \top) \wedge (\mathrm{x} = 0 \Rightarrow \bot) \wedge (\mathrm{x} > 0 \Rightarrow \bot)$
$\mathrm{s} = -1;$
$(\mathrm{x} < 0 \Rightarrow \mathrm{s} = -1) \wedge (\mathrm{x} = 0 \Rightarrow \bot) \wedge (\mathrm{x} > 0 \Rightarrow \bot)$
$\}$
$(\mathrm{x} < 0 \Rightarrow \mathrm{s} = -1) \wedge (\mathrm{x} = 0 \Rightarrow \mathrm{s} = 1) \wedge (\mathrm{x} > 0 \Rightarrow \mathrm{s} = 1)$
① $\mathrm{y} = \mathrm{x}/\mathrm{s};$
$(\mathrm{x} < 0 \Rightarrow \mathrm{s} = -1 \wedge \mathrm{y} > 0) \wedge (\mathrm{x} = 0 \Rightarrow \mathrm{s} = 1 \wedge \mathrm{y} = 0) \wedge (\mathrm{x} > 0 \Rightarrow \mathrm{s} = 1 \wedge \mathrm{y} > 0)$
② **assert**$(\mathrm{y} \geq 0);$

Outline

1 Introduction

2 Imprecisions in convex abstractions

3 Disjunctive completion

4 Cardinal power and partitioning abstractions

5 State partitioning
- Definition and examples
- Control states partitioning and iteration techniques
- Abstract interpretation with boolean partitioning

6 Trace partitioning

7 Conclusion

Computation of abstract semantics and partitioning

- we first consider **partitioning by control states**
- we rely on the two steps partitioning abstraction
  i.e., to be **composed** with an abstraction of $\mathcal{P}(\mathbb{M})$
- the techniques considered below **extend to other forms of partitioning**

This abstraction corresponds to a **Galois connection**:

$$(\mathcal{P}(\mathbb{L} \times \mathbb{M}), \subseteq) \underset{\alpha_{\text{part}}}{\overset{\gamma_{\text{part}}}{\leftrightarrows}} (\mathbb{D}^{\sharp}_{\text{part}}, \dot{\subseteq})$$

where $\mathbb{D}^{\sharp}_{\text{part}} = \mathbb{L} \to \mathcal{P}(\mathbb{M})$ and:

$$\begin{array}{rcll}
\alpha_{\text{part}} : & \mathcal{P}(\mathbb{L} \times \mathbb{M}) & \longrightarrow & \mathbb{D}^{\sharp}_{\text{part}} \\
 & \mathcal{E} & \longmapsto & \lambda(\ell \in \mathbb{L}) \cdot \{ m \in \mathbb{M} \mid (\ell, m) \in \mathcal{E} \} \\
\gamma_{\text{part}} : & \mathbb{D}^{\sharp}_{\text{part}} & \longrightarrow & \mathcal{P}(\mathbb{L} \times \mathbb{M}) \\
 & X^{\sharp} & \longmapsto & \{ (\ell, m) \in \mathbb{S} \mid m \in X^{\sharp}(\ell) \}
\end{array}$$

Fixpoint form of a partitioned semantics

- We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$
- The reachable states are computed as $[\![\mathcal{S}]\!]_{\mathcal{R}} = \mathbf{lfp}_{\mathbb{S}_{\mathcal{I}}} F$ where

$$\begin{array}{rlcl} F: & \mathcal{P}(\mathbb{S}) & \longrightarrow & \mathcal{P}(\mathbb{S}) \\ & X & \longmapsto & \{s \in \mathbb{S} \mid \exists s' \in X,\ s' \rightarrow s\} \end{array}$$

Semantic function over the partitioned system

We let F_{part} be defined over $\mathbb{D}^{\sharp}_{\mathrm{part}}$ by:

$$\begin{array}{rlcl} F_{\mathrm{part}}: & \mathbb{D}^{\sharp}_{\mathrm{part}} & \longrightarrow & \mathbb{D}^{\sharp}_{\mathrm{part}} \\ & X^{\sharp} & \longmapsto & \lambda(\ell \in \mathbb{L}) \cdot \{m \in \mathbb{M} \mid \exists \ell' \in \mathbb{L},\ \exists m' \in X^{\sharp}(\ell'), \\ & & & \qquad (\ell', m') \rightarrow (\ell, m)\} \end{array}$$

Then $F_{\mathrm{part}} \circ \alpha_{\mathrm{part}} = \alpha_{\mathrm{part}} \circ F$, and

$$\alpha_{\mathrm{part}}([\![\mathcal{S}]\!]_{\mathcal{R}}) = \mathbf{lfp}_{\alpha_{\mathrm{part}}(\mathbb{S}_{\mathcal{I}})} F_{\mathrm{part}}$$

Abstract equations form of a partitioned semantics

- we look for a set of equivalent abstract equations
- let us consider the system of semantic equations over sets of states $\mathcal{E}_1, \ldots, \mathcal{E}_s \in \mathcal{P}(\mathbb{M})$:

$$\left\{ \begin{array}{rcl} \mathcal{E}_1 & = & \bigcup_i \{ m \in \mathbb{M} \mid \exists m' \in \mathcal{E}_i,\ (l_i, m') \rightarrow (l_1, m) \} \\ & \vdots & \\ \mathcal{E}_s & = & \bigcup_i \{ m \in \mathbb{M} \mid \exists m' \in \mathcal{E}_i,\ (l_i, m') \rightarrow (l_s, m) \} \end{array} \right.$$

If we let $F_i : (\mathcal{E}_1, \ldots, \mathcal{E}_s) \mapsto \bigcup_i \{ m \in \mathbb{M} \mid \exists m' \in \mathcal{E}_i,\ (l_i, m') \rightarrow (l_i, m) \}$, then, we can prove that:

$$\alpha_{\mathrm{part}}(\llbracket \mathcal{S} \rrbracket_{\mathcal{R}}) \text{ is the least solution of the system } \left\{ \begin{array}{rcl} \mathcal{E}_1 & = & F_1(\mathcal{E}_1, \ldots, \mathcal{E}_s) \\ & \vdots & \\ \mathcal{E}_s & = & F_s(\mathcal{E}_1, \ldots, \mathcal{E}_s) \end{array} \right.$$

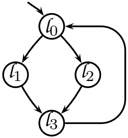
Partitioned systems and fixpoint computation

How to compute an abstract invariant for a partitioned systme described by a set of abstract equations ?

(for now, we assume no convergence issue, i.e., that the abstract lattice is of finite height)

- In practice F_i depends **only on a few of its arguments**
  i.e., $\mathcal{E}_k$ depends only on the predecessors of l_k in the control flow graph of the program under consideration
- **Example** of a simple system of abstract equations:

$$\left\{\begin{array}{lcl}\mathcal{E}_0 & = & \mathcal{I} \cup F_0(\mathcal{E}_3) \\ \mathcal{E}_1 & = & F_1(\mathcal{E}_0) \\ \mathcal{E}_2 & = & F_2(\mathcal{E}_0) \\ \mathcal{E}_3 & = & F_3(\mathcal{E}_1, \mathcal{E}_2)\end{array}\right.$$



where $\alpha_{\mathrm{part}}(\mathbb{S}_{\mathcal{I}}) = (\mathbb{S}_{\mathcal{I}}, \bot, \bot, \bot)$ (i.e., init states are at point l_0)

Partitioned systems and fixpoint computation

Following the fixpoint transfer, we obtain the following abstract iterates $(\mathcal{E}^\sharp_n)_{n\in\mathbb{N}}$:

$$\begin{array}{lcllll}
\mathcal{E}^\sharp_0 & = & (\mathbb{I}, & \bot, & \bot, & \bot) \\
\mathcal{E}^\sharp_1 & = & (\mathbb{I}, & F^\sharp_1(\mathbb{I}), & F^\sharp_2(\mathbb{I}), & \bot) \\
\mathcal{E}^\sharp_2 & = & (\mathbb{I}, & F^\sharp_1(\mathbb{I}), & F^\sharp_2(\mathbb{I}), & F^\sharp_3(F^\sharp_1(\mathbb{I}), F^\sharp_2(\mathbb{I}))) \\
\mathcal{E}^\sharp_3 & = & (\mathbb{I} \sqcup F^\sharp_0(F^\sharp_3(F^\sharp_1(\mathbb{I}), F^\sharp_2(\mathbb{I}))), & F^\sharp_1(\mathbb{I}), & F^\sharp_2(\mathbb{I}), & F^\sharp_3(F^\sharp_1(\mathbb{I}), F^\sharp_2(\mathbb{I})))
\end{array}$$

- Each iteration causes **the recomputation of all components**
- Though, each iterate differs from the previous one **in only a few components**

Chaotic iterations: principle

Fairness

Let K be a finite set. A sequence $(k_n)_{n\in\mathbb{N}}$ of elements of K is fair if and only if, for all $k \in K$, the set $\{n \in \mathbb{N} \mid k_n = k\}$ is infinite.

- Other alternate definition: $\forall k \in K, \forall n_0 \in \mathbb{N}, \exists n \in \mathbb{N}, n > n_0 \wedge k_n = k$
- i.e., all elements of K is encountered infintely often

Chaotic iterations

A chaotic sequence of iterates is a sequence of abstract iterates $(X^\sharp_n)_{n\in\mathbb{N}}$ in $\mathbb{D}^\sharp_{\mathrm{part}}$ such that there exists a sequence $(k_n)_{n\in\mathbb{N}}$ of elements of $\{1, \ldots s\}$ such that:

$$X^\sharp_{n+1} = \lambda(l_i \in \mathbb{L}) \cdot \begin{cases} X^\sharp_n(l_i) & \text{if } i \neq k_n \\ X^\sharp_n(l_i) \sqcup F^\sharp(X^\sharp_n(l_1), \ldots, X^\sharp_n(l_s)) & \text{if } i = k_n \end{cases}$$

Chaotic iterations: soundness

Soundness

Assuming the abstract lattice satisfies the ascending chain condition, any sequence of chaotic iterates computes the abstract fixpoint:

$$\lim\,(X^\sharp_n)_{n\in\mathbb{N}} = \alpha_{\mathrm{part}}(\llbracket \mathcal{S} \rrbracket_{\mathcal{R}})$$

Proof: exercise

- **Applications**: we can recompute only what is necessary
- **Back to the example**, where only the **recomputed components** are **colored**:

$$\begin{array}{rcllll}
\mathcal{E}^\sharp_0 & = & (\mathbb{I}, & \bot, & \bot, & \bot) \\
\mathcal{E}^\sharp_1 & = & (\mathbb{I}, & F^\sharp_1(\mathbb{I}), & \bot, & \bot) \\
\mathcal{E}^\sharp_2 & = & (\mathbb{I}, & F^\sharp_1(\mathbb{I}), & F^\sharp_2(\mathbb{I}), & \bot) \\
\mathcal{E}^\sharp_3 & = & (\mathbb{I}, & F^\sharp_1(\mathbb{I}), & F^\sharp_2(\mathbb{I}), & F^\sharp_3(F^\sharp_1(\mathbb{I}), F^\sharp_2(\mathbb{I}))) \\
\mathcal{E}^\sharp_4 & = & (\mathbb{I} \sqcup F^\sharp_0(F^\sharp_3(F^\sharp_1(\mathbb{I}), F^\sharp_2(\mathbb{I}))), & F^\sharp_1(\mathbb{I}), & F^\sharp_2(\mathbb{I}), & F^\sharp_3(F^\sharp_1(\mathbb{I}), F^\sharp_2(\mathbb{I})))
\end{array}$$

Chaotic iterations: worklist algorithm

Worklist algorithms

Principle:

- maintain a **queue of partitions to update**
- initialize the queue with the **entry label** of the program and the local invariant at that point at $\alpha_{\mathrm{num}}(\mathbb{S}_{\mathcal{I}})$
- for each iterate, **update the first partition in the queue** (after removing it), and add to the queue all its successors *unless* the updated invariant is equal to the former one
- **terminate** when the queue is **empty**

This algorithm implements a **chaotic iteration** strategy, thus it is sound

- **Application**: only partitions that need be updated are recomputed
- **Implemented in many static analyzers**

Selection of a set of widening points for a partitioned system

- We compose an abstraction $\mathbb{D}^{\sharp}_{\mathrm{num}}$, with concretization $\gamma_{\mathrm{num}} : \mathbb{D}^{\sharp}_{\mathrm{num}} \rightarrow \mathcal{P}(\mathbb{M})$, that may not satisfy ascending chain condition
- We assume $\mathbb{D}^{\sharp}_{\mathrm{num}}$ provides widening operator $\triangledown$

How to adapt the chaotic iteration strategy, i.e. guarantee termination and soundness ?

Enforcing termination of chaotic iterates

Let $K_{\triangledown} \subseteq \{1, \ldots, s\}$ such that each cycle in the control flow graph of the program contains at least one point in $K_{\triangledown}$; we define the chaotic abstract iterates with widening as follows:

$$X^{\sharp}_{n+1} = \lambda(l_i \in \mathbb{L}) \cdot \begin{cases} X^{\sharp}_{n}(l_i) & \text{if } i \neq k_n \\ X^{\sharp}_{n}(l_i) \sqcup F^{\sharp}(X^{\sharp}_{n}(l_1), \ldots, X^{\sharp}_{n}(l_s)) & \text{if } i = k_n \wedge l_i \notin K_{\triangledown} \\ X^{\sharp}_{n}(l_i) \triangledown F^{\sharp}(X^{\sharp}_{n}(l_1), \ldots, X^{\sharp}_{n}(l_s)) & \text{if } i = k_n \wedge l_i \in K_{\triangledown} \end{cases}$$

Selection of a set of widening points for a partitioned system

Soundness and termination

Under the assumption of a fair iteration strategy, sequence $(X_n^\sharp)_{n \in \mathbb{N}}$ terminates and computes a sound abstract post-fixpoint:

$$\exists n_0 \in \mathbb{N}, \left\{ \begin{array}{l} \forall n \geq n_0,\ X_{n_0}^\sharp = X_n^\sharp \\ [\![\mathcal{S}]\!]_{\mathcal{R}} \subseteq \gamma_{\mathrm{part}}(X_{n_0}) \end{array} \right.$$

Proof: exercise

Outline

1 Introduction

2 Imprecisions in convex abstractions

3 Disjunctive completion

4 Cardinal power and partitioning abstractions

5 State partitioning
- Definition and examples
- Control states partitioning and iteration techniques
- Abstract interpretation with boolean partitioning

6 Trace partitioning

7 Conclusion

Computation of abstract semantics and partitioning

We now **compose two forms of partitioning**

- by **control states** (as previously), using a chaotic iteration strategy
- by **the values of the boolean variables**

Thus, the abstract domain is of the form

$$\mathbb{L} \longrightarrow (\mathbb{V}_{\mathrm{bool}}^{k} \longrightarrow \mathbb{D}_{0}^{\sharp})$$

- we could do a partitioning by $\mathbb{L} \times \mathbb{V}_{\mathrm{bool}}^{k}$
- yet, it is not practical, as transitions from "boolean states" are not know before the analysis
- thus, we seek for an approximation, for all pair $\ell, \ell' \in \mathbb{L}$ of

$$\begin{array}{rrcl} \delta_{\ell,\ell'}: & \mathbb{M} & \longrightarrow & \mathcal{P}(\mathbb{M}) \\ & m & \longmapsto & \{m' \in \mathbb{M} \mid (\ell, m) \rightarrow (\ell', m')\} \end{array}$$

Transfer functions: scalar test and assignment

Assignment $\ell_0 : \mathtt{x} = \mathtt{e}; \ell_1$ affecting **only integer variables** (i.e., e depends only on $\mathtt{x}_0, \ldots, \mathtt{x}_l$):

- **concrete transition** δ_{ℓ_0,ℓ_1} defined by
$$\delta_{\ell_0,\ell_1}(m) = \{m[\mathtt{x} \leftarrow [\![\mathtt{e}]\!](m)]\}$$
- the values of the boolean variables are unchanged
thus the partitions are preserved (**pointwise** transfer function):
$$\mathit{assign}_{\rightarrow}(\mathtt{x}, \mathtt{e}, X^\sharp) = \lambda(z^\sharp \in \mathbb{D}_0^\sharp) \cdot \mathit{assign}_1(\mathtt{x}, \mathtt{e}, X^\sharp(z^\sharp))$$

Soundness

If assign_1 is sound, so is $\mathit{assign}_{\rightarrow}$, in the sense that:

$$\forall X^\sharp \in \mathbb{D}_{\rightarrow}^\sharp,\ \forall m \in \gamma_{\rightarrow}(X^\sharp),\ m[\mathtt{x} \leftarrow [\![\mathtt{e}]\!](m)] \in \gamma_{\rightarrow}(\mathit{assign}_{\rightarrow}(\mathtt{x}, \mathtt{e}, X^\sharp))$$

Transfer functions: scalar test and assignment

Condition test $\ell_0 : \mathbf{if}(\mathrm{c})\{\ell_1 : \ldots\}$ affecting **only scalar variables** (i.e., c depends only on $\mathrm{x}_0, \ldots, \mathrm{x}_l$):

- **concrete transition** δ_{ℓ_0,ℓ_1} defined by
$$\delta_{\ell_0,\ell_1}(m) = \begin{cases} \{m\} & \text{if } [\![\mathrm{c}]\!](m) = \mathtt{TRUE} \\ \emptyset & \text{if } [\![\mathrm{c}]\!](m) = \mathtt{FALSE} \end{cases}$$
- the partitions are preserved, thus we get a **pointwise** transfer function:
$$\mathit{test}_{\rightarrow}(\mathrm{c}, X^\sharp) = \lambda(z^\sharp \in \mathbb{D}_0^\sharp) \cdot \mathit{test}_1(\mathrm{c}, X^\sharp(z^\sharp))$$
- **example**:
$$\mathit{test}_{\rightarrow}\left(\mathtt{x} \geq 8, \left\{ \begin{array}{ccccc} & \mathrm{b} & \Rightarrow & x \geq 0 \\ \wedge & \neg\mathrm{b} & \Rightarrow & x \leq 0 \end{array} \right\}\right) = \left\{ \begin{array}{ccccc} & \mathrm{b} & \Rightarrow & x \geq 8 \\ \wedge & \neg\mathrm{b} & \Rightarrow & \top \end{array} \right\}$$

Soundness

If test_1 is sound, so is $\mathit{test}_{\rightarrow}$, in the sense that:

$$\forall X^\sharp \in \mathbb{D}_{\rightarrow}^\sharp,\ \forall m \in \gamma_{\rightarrow}(X^\sharp),\ [\![\mathrm{c}]\!](m) = \mathtt{TRUE} \Longrightarrow m \in \gamma_{\rightarrow}(\mathit{test}_{\rightarrow}(\mathtt{x}, \mathrm{e}, X^\sharp))$$

Transfer functions: boolean condition test

Condition test $l_0 : \mathbf{if}(\mathrm{c})\{l_1 : \ldots\}$ affecting **only boolean variables** (i.e., c depends only on $\mathrm{b}_0, \ldots, \mathrm{b}_k$):

- then, we simply need to filter the boolean partitions **satisfying** c:

$$test_{\rightarrow}(\mathrm{c}, X^\sharp) = \lambda(z^\sharp \in \mathbb{D}_0^\sharp) \cdot \begin{cases} X^\sharp(z^\sharp) & \text{if } test_0(\mathrm{c}, X^\sharp(z^\sharp)) \neq \bot_0 \\ \bot_1 & \text{otherwise} \end{cases}$$

- **for instance**:

$$test_{\rightarrow}\left(\neg \mathrm{b}_1, \left\{ \begin{array}{ccccc} & \mathrm{b}_0 \wedge \mathrm{b}_1 & \Rightarrow & 15 \leq \mathtt{x} \\ \wedge & \mathrm{b}_0 \wedge \neg \mathrm{b}_1 & \Rightarrow & 9 \leq \mathtt{x} \leq 14 \\ \wedge & \neg \mathrm{b}_0 \wedge \mathrm{b}_1 & \Rightarrow & 6 \leq \mathtt{x} \leq 8 \\ \wedge & \neg \mathrm{b}_0 \wedge \neg \mathrm{b}_1 & \Rightarrow & \mathtt{x} \leq 5 \end{array} \right\}\right) = \left\{ \begin{array}{cccc} & \mathrm{b}_0 \wedge \mathrm{b}_1 & \Rightarrow & \bot_1 \\ \wedge & \mathrm{b}_0 \wedge \neg \mathrm{b}_1 & \Rightarrow & 9 \leq \mathtt{x} \leq 14 \\ \wedge & \neg \mathrm{b}_0 \wedge \mathrm{b}_1 & \Rightarrow & \bot_1 \\ \wedge & \neg \mathrm{b}_0 \wedge \neg \mathrm{b}_1 & \Rightarrow & \mathtt{x} \leq 5 \end{array} \right\}$$

Soundness

If $test_0$ is sound, so is $test_{\rightarrow}$, in the sense that:

$$\forall X^\sharp \in \mathbb{D}_{\rightarrow}^\sharp,\ \forall m \in \gamma_{\rightarrow}(X^\sharp),\ [\![\mathrm{c}]\!](m) = \mathtt{TRUE} \Longrightarrow m \in \gamma_{\rightarrow}(test_{\rightarrow}(\mathtt{x}, \mathrm{e}, X^\sharp))$$

Transfer functions: mixed assignment

Assignment $\ell_0 : \mathtt{b} = \mathtt{e}; \ell_1$ to a **boolean variable**, where the right hand side contains **only integer variables** (i.e., $\mathtt{e}$ depends only on $\mathtt{x}_0, \ldots, \mathtt{x}_l$):

- let $z^\sharp \in \mathbb{D}_0^\sharp$, such that $z^\sharp(\mathtt{b}) = \mathtt{TRUE}$
  $\mathit{assign}_\rightarrow(\mathtt{b}, \mathtt{e}[\mathtt{x}_0, \ldots, \mathtt{x}_i], X^\sharp)(z^\sharp)$ should account for all states where $\mathtt{b}$ becomes true, other boolean variables remaining unchanged:
  $$\mathit{assign}_\rightarrow(\mathtt{b}, \mathtt{e}, X^\sharp)(z^\sharp) = \left\{ \begin{array}{cl} & \mathit{test}_1(\mathtt{e}, X^\sharp(z^\sharp)) \\ \sqcup_1 & \mathit{test}_1(\mathtt{e}, X^\sharp(z^\sharp[\mathtt{b} \leftarrow \mathtt{FALSE}])) \end{array} \right.$$
- same computation for cases where $z^\sharp(\mathtt{b}) = \mathtt{FALSE}$
- **for instance**:
  $$\mathit{assign}_\rightarrow\left(\mathtt{b}_0, \mathtt{x} \leq 7, \left\{ \begin{array}{cccc} & \mathtt{b}_0 \wedge \mathtt{b}_1 & \Rightarrow & 15 \leq \mathtt{x} \\ \wedge & \mathtt{b}_0 \wedge \neg\mathtt{b}_1 & \Rightarrow & 9 \leq \mathtt{x} \leq 14 \\ \wedge & \neg\mathtt{b}_0 \wedge \mathtt{b}_1 & \Rightarrow & 6 \leq \mathtt{x} \leq 8 \\ \wedge & \neg\mathtt{b}_0 \wedge \neg\mathtt{b}_1 & \Rightarrow & \mathtt{x} \leq 5 \end{array} \right\}\right) = \left\{ \begin{array}{cccc} & \mathtt{b}_0 \wedge \mathtt{b}_1 & \Rightarrow & 6 \leq \mathtt{x} \leq 7 \\ \wedge & \mathtt{b}_0 \wedge \neg\mathtt{b}_1 & \Rightarrow & \mathtt{x} \leq 5 \\ \wedge & \neg\mathtt{b}_0 \wedge \mathtt{b}_1 & \Rightarrow & 8 \leq \mathtt{x} \\ \wedge & \neg\mathtt{b}_0 \wedge \neg\mathtt{b}_1 & \Rightarrow & 9 \leq \mathtt{x} \leq 14 \end{array} \right\}$$

The partitions get modified (this is a **costly step**, involving join)

Choice of boolean partitions

- Boolean partitioning allows to **express relations between boolean and scalar variables**
- These relations are **expensive**:
  1. Partitioning with respect to N boolean variables translates into a 2^N space cost factor
  2. After assignments, partitions need be recomputed
- **Packing** addresses the first issue:
  - select groups of variables for which relations would be **useful**
  - can be based on **syntactic** or **semantic** criteria

  Whatever the packs, the transfer functions will produce a sound result (but possibly not the most precise one)

- How to alleviate the second issue ?

Outline

1 Introduction

2 Imprecisions in convex abstractions

3 Disjunctive completion

4 Cardinal power and partitioning abstractions

5 State partitioning

6 Trace partitioning

7 Conclusion

Definition of trace partitioning

Assumptions: we start from a **trace semantics**
and use **an abstraction of execution history for partitioning**

- **concrete domain**: $\mathbb{D} = \mathcal{P}(\mathbb{S}^\star)$
- **left side abstraction** $\gamma_0 : \mathbb{D}_0^\sharp \to \mathbb{D}$: a **trace abstraction**
- **right side abstraction**, as a **composition** of two abstractions:
  - the **final state abstraction** defined by $(\mathbb{D}_1^\sharp, \sqsubseteq_1^\sharp) = (\mathcal{P}(\mathbb{S}), \subseteq)$ and:

$$\begin{array}{rccl} \gamma_1 : & \mathbb{D}_1^\sharp & \longrightarrow & \mathcal{P}(\mathbb{S}^\star) \\ & M & \longmapsto & \{\langle s_0, \ldots, s_k, (\ell, m)\rangle \mid m \in M, \ell \in \mathbb{L}, s_0, \ldots, s_k \in \mathbb{S}\} \end{array}$$

  - a **store abstraction** applied to the traces final memory state
    $\gamma_2 : \mathbb{D}_2^\sharp \to \mathbb{D}_1^\sharp$

Trace partitioning

Cardinal power abstraction defined by an abstraction of sets of traces
$\gamma_0 : \mathbb{D}_0^\sharp \longrightarrow \mathcal{P}(\mathbb{S}^\star)$

Application 1: partitioning by control states

Flow sensitive abstraction

- We let $\mathbb{D}_0^\sharp = \mathbb{L}$
- Concretization is defined by:

$$\begin{array}{rlcl} \gamma_0 : & \mathbb{D}_0^\sharp & \longrightarrow & \mathcal{P}(\mathbb{S}^\star) \\ & \ell & \longmapsto & \mathbb{S}^\star \cdot (\{\ell\} \times \mathbb{M}) \end{array}$$

This produces the same flow sensitive abstraction as with state partitioning; in the following we always compose context sensitive abstraction with other abstractions

Trace partitioning is more general than state partitioning

It can also express

- **context-sensitivity**, **partial context sensitivity**
- partitioning guided by a **boolean condition**...

Application 2: partitioning guided by a condition

We consider a program with a **conditional statement**:

$$
\begin{array}{ll}
\ell_0: & \mathbf{if}(\mathrm{c})\{ \\
\ell_1: & \quad \ldots \\
\ell_2: & \}\mathbf{else}\{ \\
\ell_3: & \quad \ldots \\
\ell_4: & \} \\
\ell_5: & \ldots
\end{array}
$$

Domain of partitions

The partitions are defined by $\mathbb{D}_0^\sharp = \{\mathrm{if}_{\mathbf{t}}, \mathrm{if}_{\mathbf{f}}, \top\}$ and:

$$
\begin{array}{rlcl}
\gamma_0: & \mathrm{if}_{\mathbf{t}} & \longmapsto & \{\langle(\ell_0, m), (\ell_1, m'), \ldots\rangle \mid m \in \mathbb{M}, m' \in \mathbb{M}\} \\
& \mathrm{if}_{\mathbf{f}} & \longmapsto & \{\langle(\ell_0, m), (\ell_3, m'), \ldots\rangle \mid m \in \mathbb{M}, m' \in \mathbb{M}\} \\
& \top & \longmapsto & \mathbb{S}^\star
\end{array}
$$

Application: **discriminate the executions depending on the branch they visited**

Application 2: partitioning guided by a condition

This partitioning **resolves the second example** (we do not represent $\top$ when it gives no information):

$$
\begin{array}{l}
\textbf{int}\ \mathrm{x} \in \mathbb{Z};\\
\textbf{int}\ \mathrm{s};\\
\textbf{int}\ \mathrm{y};\\
\textbf{if}(\mathrm{x} \geq 0)\{\\
\qquad\qquad \mathrm{if}_{\mathbf{t}} \Rightarrow (0 \leq \mathrm{x}) \wedge \mathrm{if}_{\mathbf{f}} \Rightarrow \bot\\
\qquad \mathrm{s} = 1;\\
\qquad\qquad \mathrm{if}_{\mathbf{t}} \Rightarrow (0 \leq \mathrm{x} \wedge \mathrm{s} = 1) \wedge \mathrm{if}_{\mathbf{f}} \Rightarrow \bot\\
\}\ \textbf{else}\ \{\\
\qquad\qquad \mathrm{if}_{\mathbf{f}} \Rightarrow (\mathrm{x} < 0) \wedge \mathrm{if}_{\mathbf{t}} \Rightarrow \bot\\
\qquad \mathrm{s} = -1;\\
\qquad\qquad \mathrm{if}_{\mathbf{f}} \Rightarrow (\mathrm{x} < 0 \wedge \mathrm{s} = -1) \wedge \mathrm{if}_{\mathbf{t}} \Rightarrow \bot\\
\}\\
\qquad\qquad \left\{\begin{array}{ccc} & \mathrm{if}_{\mathbf{t}} & \Rightarrow & (0 \leq \mathrm{x} \wedge \mathrm{s} = 1)\\ \wedge & \mathrm{if}_{\mathbf{f}} & \Rightarrow & (\mathrm{x} < 0 \wedge \mathrm{s} = -1)\end{array}\right.\\
\mathrm{y} = \mathrm{x}/\mathrm{s};\\
\qquad\qquad \left\{\begin{array}{ccc} & \mathrm{if}_{\mathbf{t}} & \Rightarrow & (0 \leq \mathrm{x} \wedge \mathrm{s} = 1 \wedge 0 \leq \mathrm{y})\\ \wedge & \mathrm{if}_{\mathbf{f}} & \Rightarrow & (\mathrm{x} < 0 \wedge \mathrm{s} = -1 \wedge 0 < \mathrm{y})\end{array}\right.
\end{array}
$$

Application 3: partitioning guided by a loop

We consider a program with a **conditional statement**:

$$
\begin{array}{ll}
\ell_0: & \textbf{while}(\mathtt{c})\{ \\
\ell_1: & \quad \ldots \\
\ell_2: & \} \\
\ell_3: & \ldots
\end{array}
$$

Domain of partitions

For a given $k \in \mathbb{N}$, the partitions are defined by $\mathbb{D}_0^\sharp = \{\mathrm{loop}_0, \mathrm{loop}_1, \ldots, \mathrm{loop}_k, \top\}$ and:

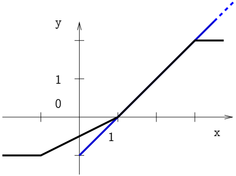
$$
\begin{array}{rlll}
\gamma_0: & \mathrm{loop}_i & \longmapsto & \text{traces that visit } \ell_1 \ i \text{ times} \\
 & \top & \longmapsto & \mathbb{S}^\star
\end{array}
$$

Application: **discriminate executions depending on the number of iterations in a loop**

Application 3: partitioning guided by a loop

An interpolation function:

$$y = \begin{cases} -1 & \text{if } x \leq -1 \\ -\frac{1}{2} + \frac{x}{2} & \text{if } x \in [-1, 1] \\ -1 + x & \text{if } x \in [1, 3] \\ 2 & \text{if } 3 \leq x \end{cases}$$



Typical implementation:

- use tables of coefficients and loops to search for the range of x

```
int i = 0;
while(i < 4 && x > t_x[i + 1]){
    i + +;
}
```

$$\begin{cases} \text{loop}_0 & \Rightarrow & x \leq -1 \\ \text{loop}_1 & \Rightarrow & -1 \leq x \leq 1 \\ \text{loop}_2 & \Rightarrow & 1 \leq x \leq 3 \\ \text{loop}_3 & \Rightarrow & 3 \leq x \end{cases}$$

$y = t_c[i] \times (x - t_x[i]) + t_y[i]$

Application 4: partitioning guided by the value of a variable

We consider a program with an integer **variable** x, and a **program point** ℓ:

$$\textbf{int } \mathrm{x}; \ldots; \ell : \ldots$$

Domain of partitions: partitioning by the value of a variable

For a given $\mathcal{E} \subseteq \mathbb{V}_{\mathrm{int}}$ finite set of integer values, the partitions are defined by $\mathbb{D}_0^\sharp = \{\mathrm{val}_i \mid i \in \mathcal{E}\} \uplus \{\top\}$ and:

$$\begin{array}{rlcl} \gamma_0 : & \mathrm{val}_k & \longmapsto & \{\langle \ldots, (\ell, m), \ldots \rangle \mid m(\mathrm{x}) = k\} \\ & \top & \longmapsto & \mathbb{S}^\star \end{array}$$

Domain of partitions: partitioning by the property of a variable

For a given abstraction $\gamma : (V^\sharp, \sqsubseteq^\sharp) \rightarrow (\mathcal{P}(\mathbb{V}_{\mathrm{int}}), \subseteq)$, the partitions are defined by $\mathbb{D}_0^\sharp = \{\mathrm{var}_{v^\sharp} \mid v^\sharp \in V^\sharp\}$ and:

$$\gamma_0 : \quad \mathrm{val}_{v^\sharp} \longmapsto \{\langle \ldots, (\ell, m), \ldots \rangle \mid m(\mathrm{x}) \in \mathrm{var}_{v^\sharp}\}$$

Application 4: partitioning guided by the value of a variable

- Left side abstraction shown in blue: **sign of $\mathtt{x}$ at entry**
- Right side abstraction shown in green:
  non relational abstraction (we omit the information about $\mathtt{x}$)
- **Same precision** and **similar results** as boolean partitioning,
  but **very different abstraction**, fewer partitions, no re-partitioning

bool $\mathtt{b}_0, \mathtt{b}_1$;
int $\mathtt{x}, \mathtt{y}$; (uninitialized)

① $(\mathtt{x} < 0@① \Rightarrow \top) \wedge (\mathtt{x} = 0@① \Rightarrow \top) \wedge (\mathtt{x} > 0@① \Rightarrow \top)$

$\mathtt{b}_0 = \mathtt{x} \geq 0;$

$(\mathtt{x} < 0@① \Rightarrow \neg\mathtt{b}_0) \wedge (\mathtt{x} = 0@① \Rightarrow \mathtt{b}_0) \wedge (\mathtt{x} > 0@① \Rightarrow \mathtt{b}_0)$

$\mathtt{b}_1 = \mathtt{x} \leq 0;$

$(\mathtt{x} < 0@① \Rightarrow \neg\mathtt{b}_0 \wedge \mathtt{b}_1) \wedge (\mathtt{x} = 0@① \Rightarrow \mathtt{b}_0 \wedge \mathtt{b}_1) \wedge (\mathtt{x} > 0@① \Rightarrow \mathtt{b}_0 \wedge \neg\mathtt{b}_1)$

if$(\mathtt{b}_0$ && $\mathtt{b}_1)\{$

$(\mathtt{x} < 0@① \Rightarrow \bot) \wedge (\mathtt{x} = 0@① \Rightarrow \mathtt{b}_0 \wedge \mathtt{b}_1) \wedge (\mathtt{x} > 0@① \Rightarrow \bot)$

$\quad \mathtt{y} = 0;$

$(\mathtt{x} < 0@① \Rightarrow \bot) \wedge (\mathtt{x} = 0@① \Rightarrow \mathtt{b}_0 \wedge \mathtt{b}_1 \wedge \mathtt{y} = 0) \wedge (\mathtt{x} > 0@① \Rightarrow \bot)$

$\}$ **else** $\{$

$(\mathtt{x} < 0@① \Rightarrow \neg\mathtt{b}_0 \wedge \mathtt{b}_1) \wedge (\mathtt{x} = 0@① \Rightarrow \bot) \wedge (\mathtt{x} > 0@① \Rightarrow \mathtt{b}_0 \wedge \neg\mathtt{b}_1)$

$\quad \mathtt{y} = 100/\mathtt{x};$

$(\mathtt{x} < 0@① \Rightarrow \neg\mathtt{b}_0 \wedge \mathtt{b}_1 \wedge \mathtt{y} \leq 0) \wedge (\mathtt{x} = 0@① \Rightarrow \bot) \wedge (\mathtt{x} > 0@① \Rightarrow \mathtt{b}_0 \wedge \neg\mathtt{b}_1 \wedge \mathtt{y} \geq 0)$

$\}$

Trace partitioning induced by a refined transition system

We search for **general** way to **generate** and **compute** partitions

- we **augment** control states **with partitioning tokens**: $\mathbb{L}' = \mathbb{L} \times \mathbb{D}_0^\sharp$ and let $\mathbb{S}' = \mathbb{L}' \times \mathbb{M}$
- let $\rightarrow' \subseteq \mathbb{S}' \times \mathbb{S}'$ be an **extended transition relation**

Partition of a transition system

System $\mathcal{S}' = (\mathbb{S}', \rightarrow', \mathbb{S}'_{\mathcal{I}})$ is a **partition** of transition system $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$ (and note $\mathcal{S}' \prec \mathcal{S}$) if and only if

- $\forall (\ell, m) \in \mathbb{S}_{\mathcal{I}},\ \exists \mathrm{tok} \in \mathbb{D}_0^\sharp,\ ((\ell, \mathrm{tok}), m) \in \mathbb{S}'_{\mathcal{I}}$
- $\forall (\ell, m), (\ell', m') \in \mathbb{S},\ \forall \mathrm{tok} \in \mathbb{D}_0^\sharp,$
$$(\ell, m) \rightarrow (\ell', m') \Longrightarrow \exists \mathrm{tok}' \in \mathbb{D}_0^\sharp,\ ((\ell, \mathrm{tok}), m) \rightarrow ((\ell', \mathrm{tok}'), m')$$

Then:

$$\forall \langle (\ell_0, m_0), \ldots, (\ell_n, m_n) \rangle \in [\![\mathcal{S}]\!]_{\mathcal{R}},$$
$$\exists \mathrm{tok}_0, \ldots, \mathrm{tok}_n \in \mathbb{D}_0^\sharp,\ \langle ((\ell_0, \mathrm{tok}_0), m_0), \ldots, ((\ell_n, \mathrm{tok}_n), m_n) \rangle \in [\![\mathcal{S}']\!]_{\mathcal{R}},$$

Trace partitioning induced by a refined transition system

- we assume $(\mathbb{S}', \rightarrow', \mathbb{S}'_{\mathcal{I}}) \prec (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$
- **erasure function**: $\Psi : (\mathbb{S}')^{\star} \rightarrow \mathbb{S}^{\star}$ removes the tokens

Definition of a trace partitioning

The abstraction defining partitions is defined by:

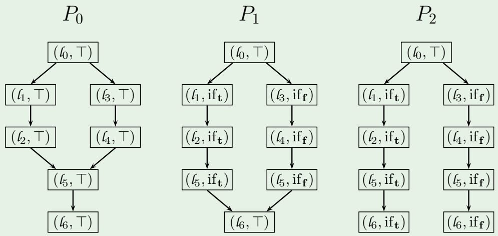
$$\begin{array}{rccl} \gamma_0 : & \mathbb{D}_0^{\sharp} & \longrightarrow & \mathcal{P}(\mathbb{S}^{\star}) \\ & \mathrm{tok} & \longmapsto & \{\sigma \in \mathbb{S}^{\star} \mid \exists \sigma' = \langle \ldots, ((\ell, \mathrm{tok}), m) \rangle \in (\mathbb{S}')^{\star},\ \Psi(\sigma') = \sigma\} \end{array}$$

- not all instances of trace partitionings can be expressed that way
- ... but many interesting instances can

Trace partitioning induced by a refined transition system

Example of the **partitioning guided by a condition**:

ℓ_0 **if**$(\mathrm{x} < 0)\{$
ℓ_1 $\quad \mathrm{s} = -1;$
ℓ_2 $\}$ **else** $\{$
ℓ_3 $\quad \mathrm{s} = 1;$
ℓ_4 $\}$
ℓ_5 $\mathrm{y} = \mathrm{x}/\mathrm{s};$
ℓ_6 $\ldots$



- each system induces a partitioning, with different merging points:

$$P_1 \prec P_0 \qquad\qquad P_2 \prec P_1$$

- these systems induce **hierarchy** of refining control structures

$$P_2 \prec P_1$$

- this approach **also applies to**:
  - partitioning **induced by a loop**
  - partitioning **induced by the value of a variable at a given point**...

Abstract interpretation of a partitioned trnsition system

- let $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$, and a refining system $\mathcal{S}' = (\mathbb{S}', \rightarrow', \mathbb{S}'_{\mathcal{I}})$, with $\mathbb{S} = \mathbb{L} \times \mathbb{M}$, $\mathbb{S}' = (\mathbb{L} \times \mathbb{D}^{\sharp}_{0}) \times \mathbb{M}$
- **transfer functions of $\mathcal{S}'$**:
  $\delta_{\ell,\ell'} : (\mathbb{D}^{\sharp}_{0} \rightarrow \mathbb{D}^{\sharp}_{1}) \longrightarrow (\mathbb{D}^{\sharp}_{0} \rightarrow \mathbb{D}^{\sharp}_{1})$ over-approximating $\rightarrow'$

Partition irrelevant transfer function

ℓ, ℓ' induces a **partition irrelevant transfer function** if and only if:

$$\begin{array}{l} \forall \mathrm{tok}, \mathrm{tok}' \in \mathbb{D}^{\sharp}_{0},\ \forall m, m' \in \mathbb{M}, \\ \qquad ((\ell, \mathrm{tok}), m) \rightarrow' ((\ell', \mathrm{tok}'), m') \Longrightarrow \mathrm{tok} = \mathrm{tok}' \end{array}$$

- partition irrelevant transfer functions: **pointwise operators of $\mathbb{D}^{\sharp}_{1}$**
  for our examples of partitioning: this is the **most common case**
- **other transfer functions**: usually for partition **creation** or **fusion**
  or **simple composition** of a creation / fusion + partition irrelevant t.f.

Transfer functions: example

$\textbf{int}\ x \in \mathbb{Z};$
$\textbf{int}\ s;$
$\textbf{int}\ y;$
$\textbf{if}(x \geq 0)\{$

$\quad \text{if}_{\mathbf{t}} \Rightarrow (0 \leq x) \wedge \text{if}_{\mathbf{f}} \Rightarrow \bot$ — partition creation: $\text{if}_{\mathbf{t}}$

$\quad s = 1;$

$\quad \text{if}_{\mathbf{t}} \Rightarrow (0 \leq x \wedge s = 1) \wedge \text{if}_{\mathbf{f}} \Rightarrow \bot$ — no modification of partitions

$\}\ \textbf{else}\ \{$

$\quad \text{if}_{\mathbf{f}} \Rightarrow (x < 0) \wedge \text{if}_{\mathbf{t}} \Rightarrow \bot$ — partition creation: $\text{if}_{\mathbf{f}}$

$\quad s = -1;$

$\quad \text{if}_{\mathbf{f}} \Rightarrow (x < 0 \wedge s = -1) \wedge \text{if}_{\mathbf{t}} \Rightarrow \bot$ — no modification of partitions

$\}$

$$\left\{\begin{array}{ccc} & \text{if}_{\mathbf{t}} & \Rightarrow & (0 \leq x \wedge s = 1) \\ \wedge & \text{if}_{\mathbf{f}} & \Rightarrow & (x < 0 \wedge s = -1) \end{array}\right.$$
no modification of partitions

$y = x/s;$

$$\left\{\begin{array}{ccc} & \text{if}_{\mathbf{t}} & \Rightarrow & (0 \leq x \wedge s = 1 \wedge 0 \leq y) \\ \wedge & \text{if}_{\mathbf{f}} & \Rightarrow & (x < 0 \wedge s = -1 \wedge 0 < y) \end{array}\right.$$
no modification of partitions

$\ldots$

$\quad \_ \Rightarrow s \in [-1, 1] \wedge 0 \leq y$ — fusion of partitions

In general, partitions are rarely modified (only *some* branching points)

Transfer functions: partition creation

Analysis of an if statement, with partitioning

$$\begin{array}{ll} \ell_0: & \mathbf{if}(\mathsf{c})\{ \\ \ell_1: & \quad \ldots \\ \ell_2: & \}\mathbf{else}\{ \\ \ell_3: & \quad \ldots \\ \ell_4: & \} \\ \ell_5: & \ldots \end{array}$$

$$\begin{array}{rcl} \delta^{\sharp}_{\ell_0,\ell_1}(X^{\sharp}) & = & [\mathrm{if}_{\mathbf{t}} \mapsto \mathit{test}(\mathsf{c}, \sqcup X^{\sharp}(\ell_0)(\mathrm{t})), \top \mapsto \bot] \\ \delta^{\sharp}_{\ell_0,\ell_3}(X^{\sharp}) & = & [\mathrm{if}_{\mathbf{t}} \mapsto \mathit{test}(\neg\mathsf{c}, \sqcup_{\mathrm{t}} X^{\sharp}(\ell_0)(\mathrm{t})), \top \mapsto \bot] \\ \delta^{\sharp}_{\ell_2,\ell_5}(X^{\sharp}) & = & X^{\sharp} \\ \delta^{\sharp}_{\ell_4,\ell_5}(X^{\sharp}) & = & X^{\sharp} \end{array}$$

- in the body of the condition: either $\mathrm{if}_{\mathbf{t}}$ or $\mathrm{if}_{\mathbf{f}}$
- effect at point ℓ_5: **both $\mathrm{if}_{\mathbf{t}}$ and $\mathrm{if}_{\mathbf{f}}$ exist**

Transfer functions: partition fusion

When **partitions are not useful anymore, they can be merged**

$$\delta^\sharp_{l_0,l_1}(X^\sharp) \;=\; [\_ \mapsto \sqcup_{\mathrm{t}} X^\sharp(l_0)(\mathrm{t})]$$

- at this point, all partitions are **effectively collapsed** into just one set
- **example**: fusion of the partition of a condition when not useful
- **choice of fusion point**:
  - **precision**: merge point should not occur as long as partitions are useful
  - **efficiency**: merge point should occur as early as partitions are not needed anymore

Choice of partitions

How are the partitions chosen ?

Static partitioning

- a fixed partitioning abstraction $\mathbb{D}_0^\sharp, \gamma_0$ is **fixed before the analysis**
- usually $\mathbb{D}_0^\sharp, \gamma_0$ are chosen by a pre-analysis

- static partitioning is rather easy to formalize and implement
- but it might be limiting, when the choice of partitions is hard

Dynamic partitioning

- the partitioning abstraction $\mathbb{D}_0^\sharp, \gamma_0$ is **not fixed before the analysis**
- instead, it is **computed as part of the analysis**
- i.e., the analysis uses on a lattice of partitioning abstractions $\mathcal{D}^\sharp$ and computes $(\mathbb{D}_0^\sharp, \gamma_0)$ as an element of this lattice

Outline

1. Introduction
2. Imprecisions in convex abstractions
3. Disjunctive completion
4. Cardinal power and partitioning abstractions
5. State partitioning
6. Trace partitioning
7. Conclusion

Adding disjunctions in static analyses

- **Disjunctive completion** is **too expensive** in practice
- The **cardinal power abstraction** expresses collections of implications between abstract facts in **two abstract domains**
- **State partitioning** and **trace partitioning** are particular cases of cardinal power abstraction
- State partitioning is **easier** to use when the criteria for partitioning can be easily expressed at the state level
- Trace partitioning is **more expressive** in general
  it can also allow the use of **simpler partioning criteria**, with less "repartitioning"