Trace Partitioning Technique[1]

Jaeho Shin ⟨netj@ropas.snu.ac.kr⟩

ROPAS Weekly Show&Tell

2005-05-14

[1] This talk is based on Mauborgne and Rival's work[1].

Today's Goal

- Understand the idea of trace partitioning
- See the efficiacy of using trace partitioning
- Get ready for improving our analyzers' accuracy

Outline

- Trace Semantics
  - Preliminaries
  - Trace Abstraction
- Trace Paritioning Abstraction
  - Abstraction with Partitioning
  - Examples
- Implementation Issues
  - Abstract Values
  - Abstract Transfer Functions
- Trace Partitioning in Practice
- Conclusion

Outline

Trace Semantics
- Preliminaries
- Trace Abstraction

Trace Paritioning Abstraction
- Abstraction with Partitioning
- Examples

Implementation Issues
- Abstract Values
- Abstract Transfer Functions

Trace Partitioning in Practice

Conclusion

Program, State, Sequence

- Programs

$$\begin{array}{rccc} \text{state} & \sigma_i & \in & \mathcal{S} \\ \text{transition} & \rightsquigarrow & \subseteq & \mathcal{S} \times \mathcal{S} \\ \text{initial state} & \sigma_\iota & \in & \mathcal{S}_\iota \end{array}$$

- Finite sequence of non-empty states $\sigma \in \mathcal{S}^+$

$$\begin{array}{rc} (i+1)^{\text{th}} \text{ state} & \sigma_i \\ \text{first state} & \sigma_0 \\ \text{last state} & \sigma_\dashv \end{array}$$

- $\tau \preceq \sigma$ iff τ is a prefix of σ where $\tau, \sigma \in \mathcal{S}^+$

Program Semantics, Trace

- Program semantics as set of Traces

$$[\![P]\!] = \{\sigma \in \mathcal{S}^+ \mid \sigma_0 \in \mathcal{S}_\iota, \forall i : \sigma_i \rightsquigarrow \sigma_{i+1}\}$$

- $[\![P]\!]$ is prefix closed
- Concrete domain of $[\![P]\!]$

$$\wp_{\preceq}(\mathcal{S}^+) = \{\Sigma \subseteq \mathcal{S}^+ \mid \Sigma \text{ is prefix closed}\}$$

Abstraction of Traces

$$\wp_{\preceq}(\mathcal{S}^+) \underset{\alpha_0}{\overset{\gamma_0}{\leftrightarrows}} D^\sharp$$

For example,

$$\begin{array}{rcl}
\mathcal{L} & = & \text{control state} \\
\mathcal{M} & = & \text{memory state} \\
\mathcal{S} & = & \mathcal{L} \times \mathcal{M} \\
D^\sharp & = & \mathcal{L} \to \mathcal{M}^\sharp \\
\alpha_o & \in & \wp_{\preceq}(\mathcal{S}^+) \to D^\sharp \\
\alpha_0(\Sigma) & = & \lambda l. \bigsqcup \{\rho' \in \mathcal{M} \mid \sigma \in \Sigma, \exists i : \sigma_i = (l, \rho), \sigma_{i+1} = (l', \rho')\}
\end{array}$$

Let's assume $\mathcal{S} = \mathcal{L} \times \mathcal{M}$ in the rest of this talk.

Decomposition

Let's decompose previous abstraction into two steps to ease the refinement.

$$\wp_{\preceq}(\mathcal{S}^+) \underset{\alpha_R}{\overset{\gamma_R}{\leftrightarrows}} \wp(\mathcal{S}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} D^\sharp$$

1. Reachability Abstraction

$$\begin{aligned} \alpha_R(\Sigma) &= \{\sigma_\dashv \mid \sigma \in \Sigma\} \\ \gamma_R(T) &= \{\sigma \in \mathcal{S}^+ \mid \forall i : \sigma_i \in T\} \end{aligned}$$

2. Set of States Abstraction

$$\alpha, \gamma \text{ provided by } D^\sharp$$

Outline

Trace Semantics
- Preliminaries
- Trace Abstraction

Trace Paritioning Abstraction
- Abstraction with Partitioning
- Examples

Implementation Issues
- Abstract Values
- Abstract Transfer Functions

Trace Partitioning in Practice

Conclusion

Partitioning

$$\delta \in E \to \wp(\mathcal{S}^+)$$

- δ is a *covering* iff
$$\bigcup_{x \in E} \delta(x) = \wp(\mathcal{S}^+)$$
- δ is a *partitioning* iff

δ is a covering, and
$\forall x, y \in E : x \neq y \Rightarrow \delta(x) \cap \delta(y) = \emptyset$

Trace Discriminating Reachability Domain

Using a well chosen $\delta \in E \to \wp(\mathcal{S}^+)$,
Trace Discriminating Reachability Domain D_R^δ is defined as

$$\wp_\preceq(\mathcal{S}^+) \underset{\alpha_R^\delta}{\overset{\gamma_R^\delta}{\leftrightarrows}} D_R^\delta = E \to \wp(\mathcal{S})$$

where

$$\begin{array}{rcl} \alpha_R^\delta(\Sigma) & = & \lambda x.\{\sigma_\dashv \mid \sigma \in \Sigma \cap \delta(x)\} \\ \gamma_R^\delta(f) & = & \{\sigma \mid \forall \tau \preceq \sigma, \forall x : \tau \in \delta(x) \Rightarrow \tau_\dashv \in f(x)\} \end{array}$$

and ordered pointwise.

Final Control State Partitioning

$$\begin{array}{rcl} \delta_{\mathcal{L}} & \in & \mathcal{L} \to \wp(\mathcal{S}^+) \\ \delta_{\mathcal{L}}(l) & = & \{\sigma \in \mathcal{S}^+ \mid \exists \rho : \sigma_{\dashv} = (l, \rho)\} \end{array}$$

- Very common
- Usually done silently when designing the abstract semantics
- Leads to the abstraction $(D^\sharp_{\mathcal{L}}, \gamma_{\mathcal{L}})$ where

$$\wp_{\preceq}(\mathcal{S}^+) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} D^\sharp_{\mathcal{L}} = \mathcal{L} \to D^\sharp$$
$$\gamma_{\mathcal{L}}(\mathcal{I}) = \{\sigma \in \wp_{\preceq}(\mathcal{S}^+) \mid \forall i : \sigma_i = (l_i, \rho_i), \rho_i \in \gamma(\mathcal{I}(l_i))\}$$

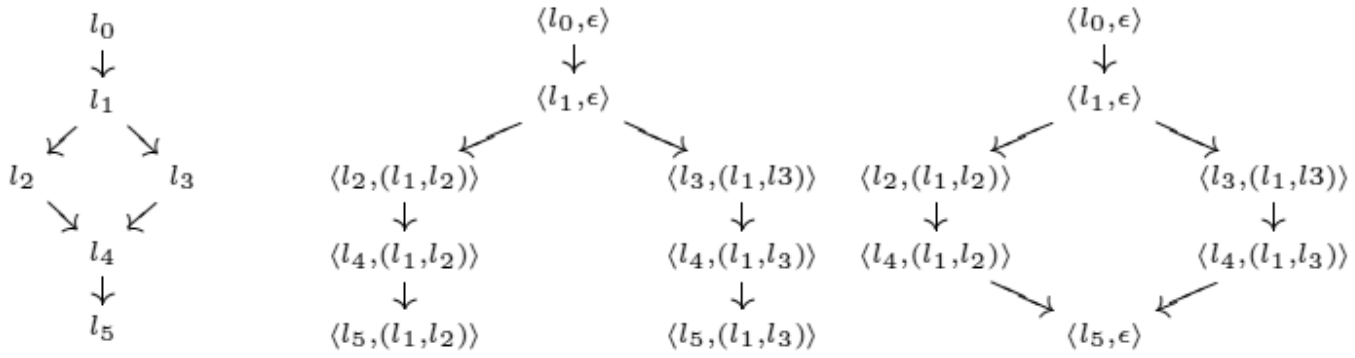
Control Flow Based Partitioning

$$\begin{aligned} \delta_{cf} &\in \mathcal{L} \times \mathcal{C}^* \to \wp(\mathcal{S}^+) \\ \delta_{cf}(l, \beta) &= \{\sigma \in \delta_{\mathcal{L}}(l) \mid cf(\sigma) = \beta\} \end{aligned}$$

where

$$\begin{aligned} \text{set of branch points} \quad \mathcal{B} &\subseteq \mathcal{L} \\ \text{branch choices} \quad \mathcal{C} &= \{(b, l) \in \mathcal{B} \times \mathcal{L} \mid \\ &\qquad \exists \rho, \rho' \in \mathcal{M} : (b, \rho) \rightsquigarrow (l, \rho')\} \\ \text{control flow abstraction} \quad cf &\in \mathcal{S}^+ \to \mathcal{C}^* \end{aligned}$$

Examples



a. Control State Partition **b.** Control Flow Partition **c.** Merging Control Flow Partition

Fig. 1. Some partitions for the program $l_0 : s_0;\ l_1 : \textbf{if}(c)\{\ l_2 : s_1;\ \}\,\textbf{else}\,\{\ l_3 : s_2;\ \}\,l_4 : s_3;\ l_5 : s_4$

Outline

Trace Semantics
- Preliminaries
- Trace Abstraction

Trace Paritioning Abstraction
- Abstraction with Partitioning
- Examples

Implementation Issues
- Abstract Values
- Abstract Transfer Functions

Trace Partitioning in Practice

Conclusion

The Big Picture

Recall set of states abstraction α, γ.
Let's apply this to the trace discriminating reachability domain D_R^δ.

$$\wp_\preceq(\mathcal{S}^+) \underset{\alpha_R^\delta}{\overset{\gamma_R^\delta}{\leftrightarrows}} D_R^\delta = E \to \wp(\mathcal{S}) \underset{\alpha\circ}{\overset{\gamma\circ}{\leftrightarrows}} E \to D^\sharp = D_E^\sharp$$

Implementation will deal with $D_E^\sharp$ and $E = \mathcal{L} \times \mathcal{T}$.
Hence, $D_E^\sharp = \mathcal{L} \times \mathcal{T} \to D^\sharp$ and $\delta \in \mathcal{L} \times \mathcal{T} \to \wp(\mathcal{S}^+)$.
Equivalently, $D_E^\sharp = \mathcal{L} \to \mathcal{T} \to D^\sharp$ and $\delta \in \mathcal{L} \to \mathcal{T} \to \wp(\mathcal{S}^+)$.

Tokens

We see from

$$\delta \in \mathcal{L} \rightarrow \mathcal{T} \rightarrow \wp(\mathcal{S}^+)$$

token $t \in \mathcal{T}$ stands for an element of the partitions observed at some control state $l \in \mathcal{L}$.

$$\begin{array}{rrcl} \text{Pre-token} & p & \in & \mathcal{P} \\ \text{Token} & t & \in & \mathcal{T} = \mathcal{P}^* \end{array}$$

$$\begin{array}{rcl} p & ::= & If_{true}(l) \mid If_{false}(l) \mid Value(x,k,l) \\ & \mid & While_{unroll}(l,k) \mid While_{iter}(l) \mid Call(f,l) \mid \cdots \\ t & ::= & \epsilon \mid p.t \end{array}$$

Tokens as Trees

- Tokens observed at a control state share common parts
- Pre-tokens created last will be merged first (e.g. branches)
- These LIFO structures of tokens can be represented efficiently by tree $d \in D_p^\sharp$ defined as

$$d \quad ::= \quad Leaf(v \in D^\sharp) \mid Node(\phi \in \mathcal{P} \to D_p^\sharp)$$

- Tree $d \in D_p^\sharp$ encodes the function $\mathcal{T} \to D^\sharp$
- With d for each control state, we have abstract values in $\mathcal{L} \to D_p^\sharp$ which are encodings of $\mathcal{L} \to \mathcal{T} \to D^\sharp$, i.e. $D_E^\sharp$.

Abstract Transfer Function

- Counterpart for $\rightsquigarrow$ to compute the approximation of $[\![P]\!]$
- Abstract transfer function corresponding to the edge (l, l')

$$\vartheta_{l,l'} \in D_p^\sharp \to D_p^\sharp$$

is sound iff

$$\begin{aligned}&\forall d \in D_p^\sharp : \forall \rho, \rho' \in \mathcal{M} : \\ &\rho \in \gamma_p(d), (l, \rho) \rightsquigarrow (l', \rho') \Rightarrow \rho' \in \gamma_p(\vartheta_{l,l'}(d))\end{aligned}$$

where $\gamma_p \in D_p^\sharp \to \mathcal{M}$

- Sound abstract semantics function can be derived from $(\vartheta_{l,l'})$

Creating Partitions

Given $guard_0 \in \mathcal{G} \times D^\sharp \to D^\sharp$ provided by $D^\sharp$ where $\mathcal{G}$ is the set of conditional expressions,

- Condition testing

$$\begin{aligned} guard &\in \mathcal{G} \times D_p^\sharp \to D_p^\sharp \\ guard(C, Leaf(v)) &= Leaf(guard_0(C, v)) \\ guard(C, Node(\phi)) &= Node(\lambda p.guard(D, \phi(p))) \end{aligned}$$

- Creating partition

$$\begin{aligned} create &\in (\mathcal{T} \to \mathcal{G}) \times D_p^\sharp \to D_p^\sharp \\ create(C, d) &= create_0(C, \epsilon, d) \\ create_0(C, t, Leaf(v)) &= Node(\lambda p.Leaf(guard_0(C(t), v))) \\ create_0(C, t, Node(\phi)) &= Node(\lambda p.create_0(C, t.p, \phi(p))) \end{aligned}$$

Merging Partitions

$$
\begin{array}{rcl}
merge & \in & \wp(\mathcal{T}) \times D_p^\sharp \to D_p^\sharp \\
merge(X, d) & = & merge_0(X, \epsilon, d) \\
merge_0(X, t, Leaf(v)) & = & Leaf(v) \\
merge_0(X, t, Node(\phi)) & = & \text{if } t \in X \\
 & & \text{then } Leaf(\bigsqcup\{v \mid Node(\phi) \\
 & & \qquad \text{has } Leaf(v)\}) \\
 & & \text{else } Node(\lambda p.merge_0(X, t.p, \phi(p)))
\end{array}
$$

$$
\begin{array}{rcl}
Node(\phi) \text{ has } Leaf(v) & = & \exists p : \phi(p) \text{ has } Leaf(v) \\
Leaf(v') \text{ has } Leaf(v) & = & v' = v
\end{array}
$$

Example

With previous three abstract transfer functions, various partitioning $\delta \in \mathcal{L} \to \mathcal{T} \to \wp(\mathcal{S}^+)$ and domain $D^\sharp_E = \mathcal{L} \to \mathcal{T} \to D^\sharp$ can be defined.

Outline

Trace Semantics
- Preliminaries
- Trace Abstraction

Trace Paritioning Abstraction
- Abstraction with Partitioning
- Examples

Implementation Issues
- Abstract Values
- Abstract Transfer Functions

Trace Partitioning in Practice

Conclusion

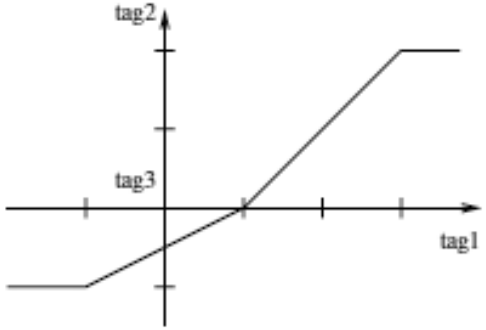
Experimental Results

Mauborgne and Rival's analysis results without and then **with partitioning** (3 GHz Bi-opteron with 8 GB of RAM):

Program	test 1		test 2		test 3		test 4	
Code size (LOCs)	70 000		65 000		215 000		380 000	
Iterations	48	**43**	33	**32**	80	**59**	163	**62**
Analysis time (minutes)	44	**70**	21	**28**	180	**330**	970	**680**
Memory peak (Mb)	520	**550**	340	**390**	1 100	**1 300**	1 800	**2 200**
Alarms	658	**0**	552	**2**	4 963	**1**	6 693	**0**

Linear Interpolations

$$
\begin{array}{lll}
l_0: & \text{int } i = 0; & \\
l_1: & \textbf{while}(i < n \ \&\& \ x > tx[i+1]) & tc = \{0; 0.5; 1; 0\} \\
l_2: & \quad i++; & tx = \{0; -1; 1; 3\} \\
l_3: & y = tc[i] \times (x - tx[i]) + ty[i] & ty = \{-1; -0.5; -1; 2\} \\
l_4: & \ldots &
\end{array}
$$



$$
y = \begin{cases} -1 & \text{if } x \leq -1 \\ -0.5 + 0.5 \times x & \text{if } -1 \leq x \leq 1 \\ -1 + x & \text{if } 1 \leq x \leq 3 \\ 2 & \text{if } 3 \leq x \end{cases}
$$

Fig. 2. Linear interpolation

Barycenter

$$
\begin{array}{ll}
l_0: & \text{int } r = 0;\ \text{float } x = 0.0; \\
l_1: & \textbf{while}(\text{true})\{ \\
l_2: & \qquad r = \text{random}(0, 50); \\
l_3: & \qquad x = (x * r + \text{random}(-100, 100))/(r + 1); \\
l_4: & \}
\end{array}
$$

Conclusion

- We have overviewed a general framework for using trace partitioning on abstract domains
- Partitioning traces based on control flow or values of variables can improve the analysis precision significantly while maintaining the cost low
- It must not be hard to incorporate ideas in this talk into Airac and design of other analyzers

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