Data-Flow Analysis for Hot-Spot Program Optimization

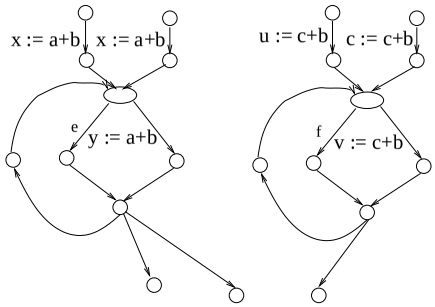
Jens Knoop

Technische Universität Wien

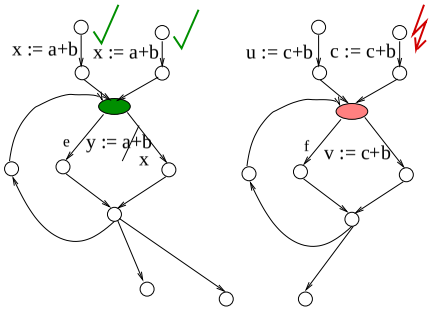
TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

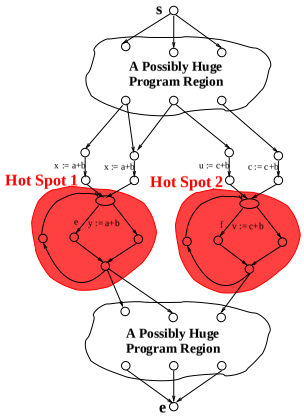
Motivation



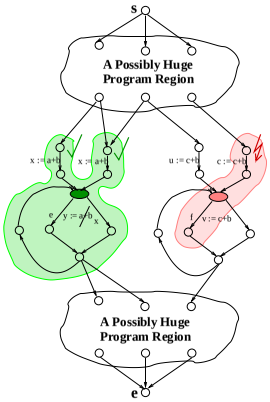
Motivation (Cont'd)



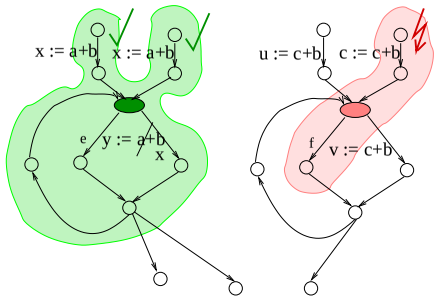
Motivation (Cont'd)



Motivation (Cont'd)



Motivation (Cont'd)



Why Standard Data-Flow Analysis Fails

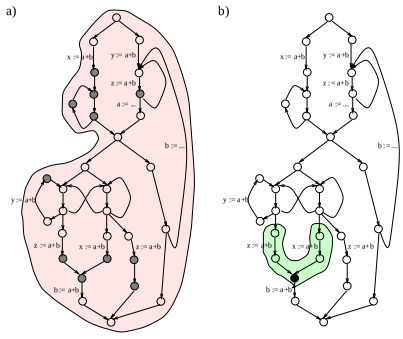
Availability of terms...

$$\mathsf{AVAIL}(n) = \begin{cases} false & \text{if } n = \mathbf{s} \\ \prod\limits_{m \in pred(n)} [\![(m,n)]\!](\mathsf{AVAIL}(n)) & \text{otherwise} \end{cases}$$

where

$$[\![(m,n)]\!](b) = (\mathbf{COMP}_{(m,n)} + b) * \mathbf{TRANSP}_{(m,n)}$$

Availability at a Single Program Point



Outline of the Talk

Standard vs. Reverse Data-Flow Analysis...

- Background
- Essentials
- The Connecting Link
- The Clou: Why does it work?
- Applications
- Conclusions

Background

Demand-Driven Data-Flow Analysis...

- Agrawal (2000)
- Horwitz, Reps, Sagiv (1994+)
- Duesterwald, Gupta, Soffa (1995+)
- ...
- Knoop (Euro-Par 1999, KPS 2007)

Reverse Data-Flow Analysis: The Basics

(Standard) Data-Flow Analysis...

- *Data-Flow Lattice* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$
- *Data-Flow Functional* $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$

Reverse Data-Flow Analysis...

- *Reverse Data-Flow Functional* (Hughes, Launchbury 1992+) $[\![\ ]\!]_R : E \to (\mathcal{C} \to \mathcal{C})$ defined by

$$\forall e \in E\, \forall c \in \mathcal{C}.\ [\![e]\!]_R(c) =_{df} \bigsqcap \{ c' \mid [\![e]\!](c') \sqsupseteq c \}$$

Availability of Terms

- **Abstract semantics for *availability of terms***
  1. *Data-Flow Lattice*: $(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top) =_{df} (\mathcal{B}, \wedge, \vee, \leq, false, true)$
  2. *Data-Flow Functional*: $[\![\ ]\!]_{av} : E \to (\mathcal{B} \to \mathcal{B})$ defined by

$$\forall e \in E.\ [\![e]\!]_{av} =_{df} \begin{cases} Cst_{true} & \text{if } Comp_e \wedge Transp_e \\ Id_{\mathcal{B}} & \text{if } \neg Comp_e \wedge Transp_e \\ Cst_{false} & \text{otherwise} \end{cases}$$

On the Relationship of $[\![\ ]\!]$ and $[\![\ ]\!]_R$

Lemma

1. $[\![e]\!]_R$ is well-defined and monotonic.
2. $[\![e]\!]_R$ is additive, if $[\![e]\!]$ is distributive.

Monotonicity, Distributivity, and Additivity

...of data-flow functions.

Definition [Monotonicity, Distributivity, Additivity]
Let $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$ be a complete lattice and $f : \mathcal{C} \to \mathcal{C}$ a function on $\mathcal{C}$. Then: f is

1. *monotonic* iff $\forall c, c' \in \mathcal{C}.\ c \sqsubseteq c' \Rightarrow f(c) \sqsubseteq f(c')$
   (Preserving the order of elements)
2. distributive iff $\forall C' \subseteq \mathcal{C}.\ f(\bigsqcap C') = \bigsqcap \{ f(c) \mid c \in C' \}$
   (Preserving greatest lower bounds)
3. *additive* iff $\forall C' \subseteq \mathcal{C}.\ f(\bigsqcup C') = \bigsqcup \{ f(c) \mid c \in C' \}$
   (Preserving least upper bounds)

Often useful

...the following equivalent characterization of monotonicity:

Lemma

Let $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$ be a complete lattice and $f : \mathcal{C} \to \mathcal{C}$ a function on $\mathcal{C}$. Then:

f is monotonic $\iff \forall C' \subseteq \mathcal{C}.\ f(\bigsqcap C') \sqsubseteq \bigsqcap \{ f(c) \mid c \in C' \}$
$(\iff \forall C' \subseteq \mathcal{C}.\ f(\bigsqcup C') \sqsupseteq \bigsqcup \{ f(c) \mid c \in C' \})$

On the Relationship of $[\![\ ]\!]$ and $[\![\ ]\!]_R$ (Cont'd)

Lemma

1. $[\![e]\!]_R \circ [\![e]\!] \sqsubseteq Id_{\mathcal{C}}$, if $[\![e]\!]$ is monotonic.
2. $[\![e]\!] \circ [\![e]\!]_R \sqsupseteq Id_{\mathcal{C}}$, if $[\![e]\!]$ is distributive.

In terms of the theory of "abstract interpretation":

- $[\![e]\!]$ and $[\![e]\!]_R$ form a Galois-connection.

Reverse DFA: The R-$MinFP$-Approach

The R-$MinFP$-Equation System:

$$\textit{reqInf}\,(n) = \begin{cases} c_q & \text{if } n = \mathbf{q} \\ \bigsqcup \{\, [\![\, (n,m) \,]\!]_R(\textit{reqInf}\,(m)) \mid m \in succ(n) \,\} & \\ & \text{otherwise} \end{cases}$$

The R-$MinFP$-Solution:

$$\forall\, c_q \in \mathcal{C}\, \forall\, n \in N.\ R\text{-}MinFP_{c_q}(n) =_{df} \textit{reqInf}^{\,*}_{c_q}(n)$$

where $\textit{reqInf}^{*}_{c_q}$ denotes the least solution of the R-$MinFP$-equation system wrt $c_q \in \mathcal{C}$.

Standard DFA: The $MaxFP$ -Approach

The $MaxFP$-Equation System:

$$\textit{inf}\,(n) \;=\; \begin{cases} c_{\mathbf{s}} & \text{if } n = \mathbf{s} \\ \bigsqcap \{\, [\![\, (m,n) \,]\!](\textit{inf}\,(m)) \mid m \in pred(n) \,\} & \\ & \text{otherwise} \end{cases}$$

The $MaxFP$-Solution:

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\, \forall\, n \in N.\ MaxFP_{([\![\,]\!],c_{\mathbf{s}})}(n) =_{df} \textit{inf}^{\,*}_{c_{\mathbf{s}}}(n)$$

where $\textit{inf}^{\,*}_{c_{\mathbf{s}}}$ denotes the greatest solution of the $MaxFP$ -equation system wrt $c_{\mathbf{s}} \in \mathcal{C}$.

The Connecting Link

Link Theorem

For distributive data-flow functionals $[\![\ ]\!]$, $q \in N$, and $c_{\mathbf{s}}, c_q \in \mathcal{C}$, we have:

$$R\text{-}MinFP_{c_q}(\mathbf{s}) \sqsubseteq c_{\mathbf{s}} \iff MaxFP_{c_{\mathbf{s}}}(q) \sqsupseteq c_q$$

Continuing the Analogy

...of Standard and Reverse Data-Flow Analysis regarding

- Soundness & Completeness (in terms of program verification) / Safety & Coincidence (Precision) (in terms of data-flow analysis)

Essential

...the extensibility of data-flow functionals to paths

$$[\![\, p \,]\!] =_{df} \begin{cases} \mathit{Id}_{\mathcal{C}} & \text{if } q < 1 \\ [\![\, \langle e_2, \ldots, e_q \rangle \,]\!] \circ [\![\, e_1 \,]\!] & \text{otherwise} \end{cases}$$

The MOP-Approach

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\, \forall\, n \in N.\ MOP_{c_{\mathbf{s}}}(n) = \bigsqcap \{\, [\![\, p \,]\!](c_{\mathbf{s}}) \mid p \in \mathbf{P}[\mathbf{s}, n] \,\}$$

Standard DFA: Main Results

Theorem [Soundness / Safety]

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\, \forall\, n \in N.\ MaxFP_{c_{\mathbf{s}}}(n) \sqsubseteq MOP_{c_{\mathbf{s}}}(n)$$

if the data-flow functional $[\![\ ]\!]$ is monotonic.

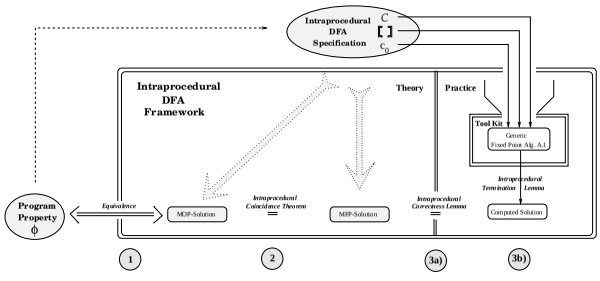
Theorem [Completeness / Coincidence (Precision)]

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\, \forall\, n \in N.\ MaxFP_{c_{\mathbf{s}}}(n) = MOP_{c_{\mathbf{s}}}(n)$$

if the data-flow functional $[\![\ ]\!]$ is distributive.

Standard DFA: The Tool Kit View

...at a glance:



Of course...

Reverse data-flow functionals can be extended to paths, too:

$$[\![p]\!]_R =_{df} \begin{cases} Id_{\mathcal{C}} & \text{if } q < 1 \\ [\![\langle e_1, \ldots, e_{q-1} \rangle]\!]_R \circ [\![e_q]\!]_R & \text{otherwise} \end{cases}$$

The R-JOP-Approach

The R-JOP-Solution:

$$\forall c_q \in \mathcal{C} \, \forall n \in N. \; R\text{-}JOP_{c_q}(n) =_{df} \bigsqcup \{ [\![p]\!]_R(c_q) \mid p \in \mathbf{P}[n, \mathbf{q}] \}$$

Reverse DFA: Main Results

Theorem [Soundness / Reverse Safety]

$$\forall c_q \in \mathcal{C} \, \forall n \in N. \; R\text{-}MinFP_{c_q}(n) \sqsupseteq R\text{-}JOP_{c_q}(n)$$

Theorem [Completeness / Reverse Coincidence (Precision)]

$$\forall c_q \in \mathcal{C} \, \forall n \in N. \; R\text{-}MinFP_{c_q}(n) = R\text{-}JOP_{c_q}(n)$$

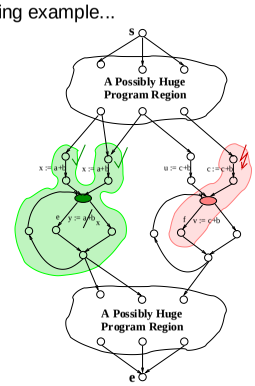
if $[\![\;]\!]$ is distributive.

Putting it together...

Data-flow Analysis		Program Verification
MaxFP	$[\![e]\!] \in \sqsubset C \xrightarrow{\text{distributive}} C \sqsupset$	*Strongest Postcondition View*
Coincidence Theorem ‖		
$MOP_{c_s}(q) \sqsupseteq c_q$		$\{p\} \, \pi \, \{?\}$
Link Theorem	*Induces*	
R-JOP $_{c_q}(s) \sqsubseteq c_s$	$[\![e]\!]_R(c) =_{df} \bigsqcap \{c' \mid [\![e]\!](c') \sqsupseteq c\}$	$\{?\} \, \pi \, \{q\}$
Reverse Coincidence Theorem ‖		
R-MinFP		*Weakest Precondition View*

Are We Done?

Recall the motivating example...

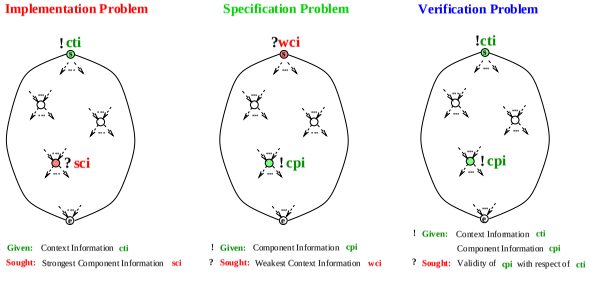


Mastering the Road to Success

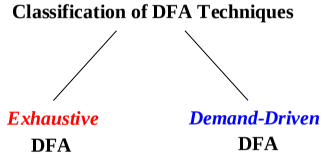
...requires more. It requires us to conclude from “weakest pre-conditions” on “strongest post-conditions”.

...essentially, this means to replace the analysis problem by a verification problem.

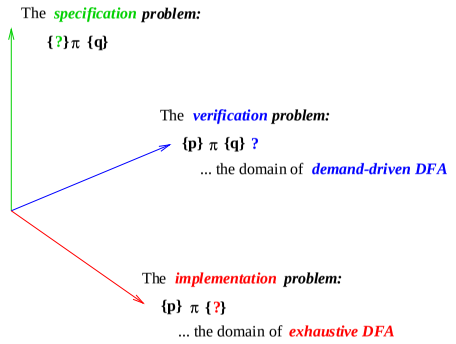
Changing the Perspective



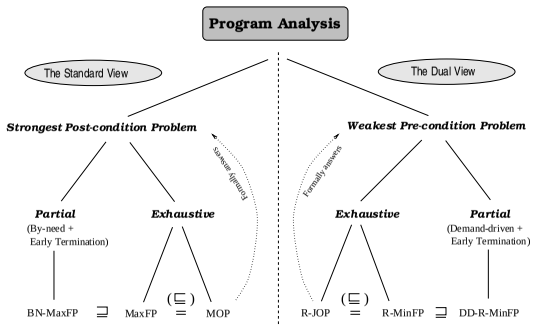
Changing the Perspective: The Standard Taxonomy



Changing the Perspective: Conclusions Derived



(R)DFA-Frameworks / (R)DFA-Tool Kits



Gen/Kill-Problems

...allow us to master *the road to success*: The SPC-analysis problem boils down to a WPC-verification problem.

This is important because...

- Redundant Expression/Assignment Elimination
- Dead-Code Elimination
- Strength Reduction
- ...

are based on Gen-Kill-problems.

Concluding the Example: Availability

Abstract semantics for *availability*

1. *Data-flow lattice*:
   $(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top) =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, false, failure)$
   with $\bot = false \sqsubset true \sqsubset failure = \top$
2. *Data-flow functional*: $[\![\ ]\!]_{av} : E \to (\mathcal{B}_X \to \mathcal{B}_X)$ defined by

$$\forall e \in E.\ [\![e]\!]_{av} =_{df} \begin{cases} Cst^X_{true} & \text{if } Comp_e \wedge Transp_e \\ Id_{\mathcal{B}_X} & \text{if } \neg Comp_e \wedge Transp_e \\ Cst^X_{false} & \text{otherwise} \end{cases}$$

Reverse Availability

Reverse abstract semantics for *availability*

1. *Data-flow lattice*:
   $(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top) =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, false, failure)$
2. *Reverse data-flow functional*: $[\![\ ]\!]_{av_R} : E \to (\mathcal{B}_X \to \mathcal{B}_X)$ defined by

$$\forall e \in E.\ [\![e]\!]_{av_R} =_{df} \begin{cases} R\text{-}Cst^X_{true} & \text{if } [\![e]\!]_{av} = Cst^X_{true} \\ R\text{-}Id_{\mathcal{B}_X} & \text{if } [\![e]\!]_{av} = Id_{\mathcal{B}_X} \\ R\text{-}Cst^X_{false} & \text{if } [\![e]\!]_{av} = Cst^X_{false} \end{cases}$$

Supporting Functions

$$\forall b \in \mathcal{B}_X.\ R\text{-}Cst^X_{true}(b) =_{df} \begin{cases} false & \text{if } b \in \mathcal{B} \\ failure & \text{otherwise} \\ & \text{(i.e., if } b = failure) \end{cases}$$

$$\forall b \in \mathcal{B}_X.\ R\text{-}Cst^X_{false}(b) =_{df} \begin{cases} false & \text{if } b = false \\ failure & \text{otherwise} \end{cases}$$

$$R\text{-}Id_{\mathcal{B}_X} =_{df} Id_{\mathcal{B}_X}$$

Summing Up / Extensions

In this talk...

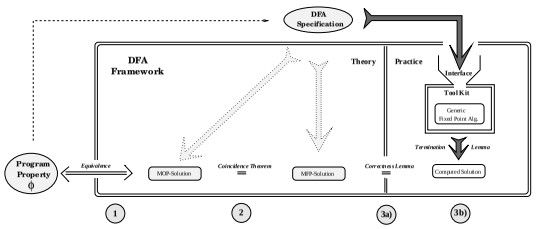
- The intraprocedural basic setting of (R)DFA (Knoop, KPS 2007)

Extensions are possible...

- Interprocedural setting (Knoop, CC 1992, LNCS 1428 (1998))
- Parallel setting (Knoop, Euro-Par 1999)
- ...

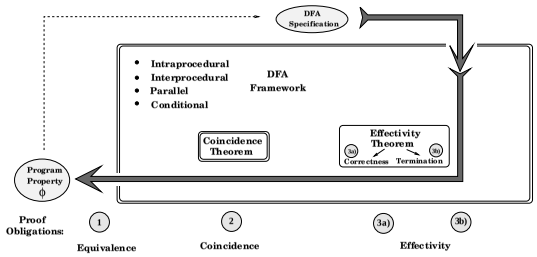
(R)DFA-Frameworks / (R)DFA-Tool Kits (Cont'd)

...the general pattern:

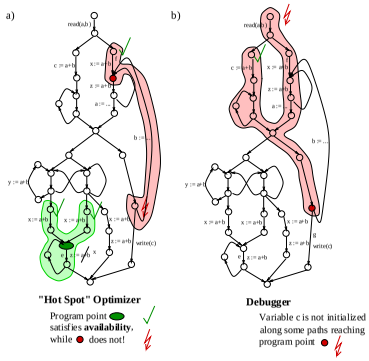


(R)DFA-Frameworks / (R)DFA-Tool Kits

...the general pattern more abstract:



Applications



From Applications towards Conclusions

Reverse Data-Flow Analysis especially well-suited for...

- Hot-Spot Optimization
- Debugging
- Just-in-time Compilation

based on answering data-flow queries.

Hence...

- Data-Flow Analysis for Debugging
- Data-Flow Analysis for Just-in-time Compilation

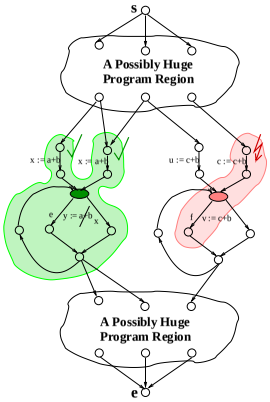
were titles considered optionally.

Conclusions (Cont'd)

As an appealing add-on...

- RDFA is tailored for parallelization!

Recall again...



Conclusions and Perspectives

Data-Flow Analysis for Multi-Core Architectures