

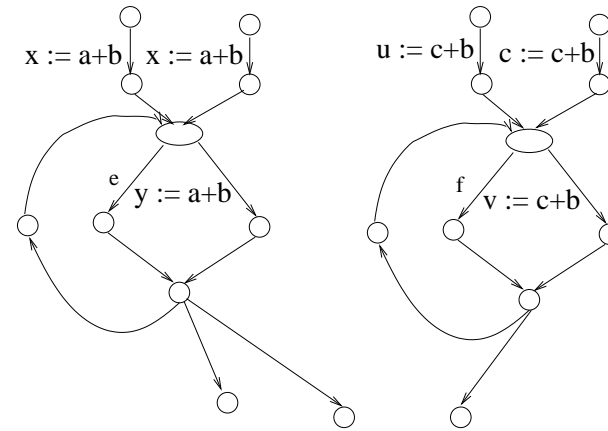
Data-Flow Analysis for Hot-Spot Program Optimization

Jens Knoop

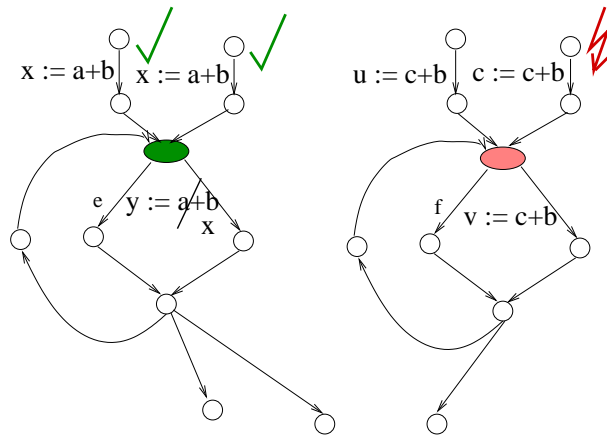
Technische Universität Wien



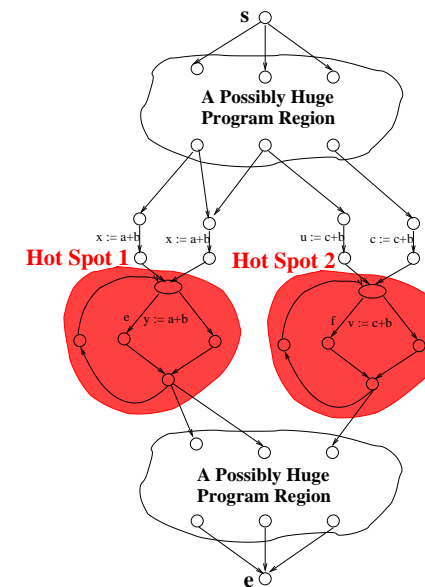
Motivation



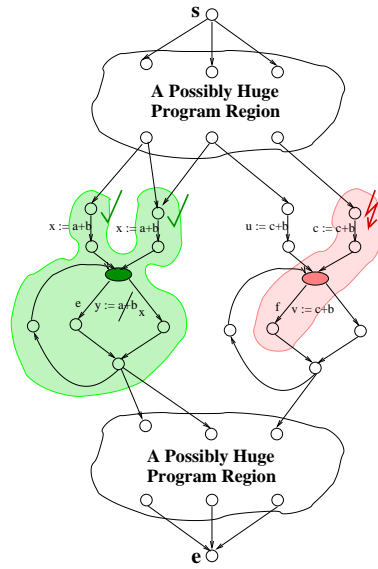
Motivation (Cont'd)



Motivation (Cont'd)

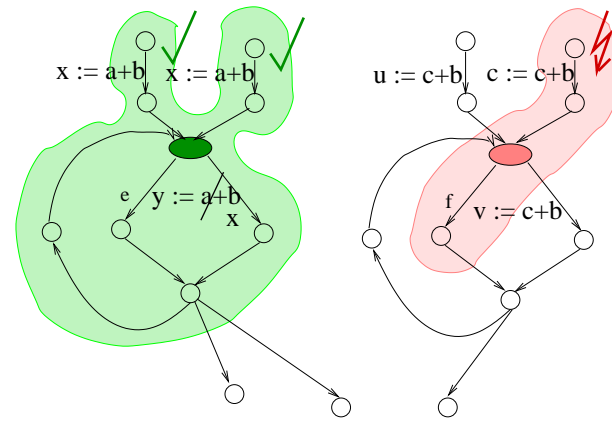


Motivation (Cont'd)



5

Motivation (Cont'd)



6

Why Standard Data-Flow Analysis Fails

Availability of terms...

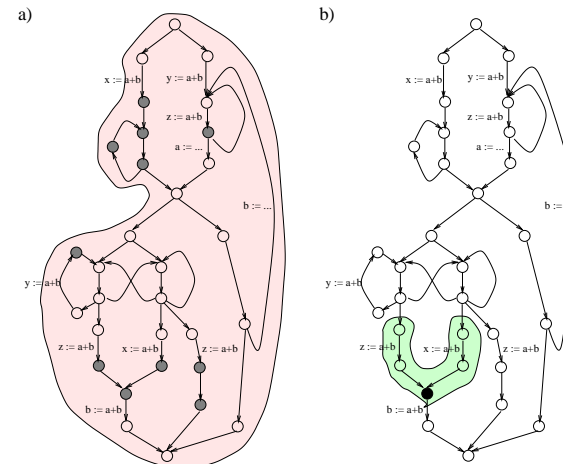
$$AVAIL(n) = \begin{cases} false & \text{if } n = s \\ \prod_{m \in pred(n)} \llbracket (m, n) \rrbracket (AVAIL(n)) & \text{otherwise} \end{cases}$$

where

$$\llbracket (m, n) \rrbracket (b) = (\mathbf{COMP}_{(m,n)} + b) * \mathbf{TRANSP}_{(m,n)}$$

7

Availability at a Single Program Point



8

Outline of the Talk

Standard vs. Reverse Data-Flow Analysis...

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

9

Background

Demand-Driven Data-Flow Analysis...

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

10

Reverse Data-Flow Analysis: The Basics

(Standard) Data-Flow Analysis...

- *Data-Flow Lattice* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top)$
- *Data-Flow Functional* $\llbracket \cdot \rrbracket : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$

Reverse Data-Flow Analysis...

- *Reverse Data-Flow Functional* (Hughes, Launchbury 1992+)
 $\llbracket \cdot \rrbracket_R : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$ defined by

$$\forall e \in E \forall c \in \mathcal{C}. \llbracket e \rrbracket_R(c) =_{df} \sqcap \{ c' \mid \llbracket e \rrbracket(c') \sqsupseteq c \}$$

11

Availability of Terms

- **Abstract semantics for *availability of terms***

1. *Data-Flow Lattice:*

$$(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top) =_{df} (\mathcal{B}, \wedge, \vee, \leq, false, true)$$

2. *Data-Flow Functional:* $\llbracket \cdot \rrbracket_{av} : E \rightarrow (\mathcal{B} \rightarrow \mathcal{B})$ defined by

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

12

On the Relationship of $\llbracket \]$ and $\llbracket \]_R$

Lemma

1. $\llbracket e \rrbracket_R$ is well-defined and monotonic.
2. $\llbracket e \rrbracket_R$ is additive, if $\llbracket e \rrbracket$ is distributive.

13

Monotonicity, Distributivity, and Additivity

...of data-flow functions.

Definition [Monotonicity, Distributivity, Additivity]

Let $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top)$ be a complete lattice and $f : \mathcal{C} \rightarrow \mathcal{C}$ 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 $\forall C' \subseteq \mathcal{C}. f(\sqcap C') = \sqcap \{f(c) \mid c \in C'\}$
(Preserving greatest lower bounds)
3. *additive* iff $\forall C' \subseteq \mathcal{C}. f(\sqcup C') = \sqcup \{f(c) \mid c \in C'\}$
(Preserving least upper bounds)

14

Often useful

...the following equivalent characterization of monotonicity:

Lemma

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

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

$$(\iff \forall C' \subseteq \mathcal{C}. f(\sqcup C') \supseteq \sqcup \{f(c) \mid c \in C'\})$$

15

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

Lemma

1. $\llbracket e \rrbracket_R \circ \llbracket e \rrbracket \sqsubseteq Id_{\mathcal{C}}$, if $\llbracket e \rrbracket$ is monotonic.
2. $\llbracket e \rrbracket \circ \llbracket e \rrbracket_R \supseteq Id_{\mathcal{C}}$, if $\llbracket e \rrbracket$ is distributive.

In terms of the theory of "abstract interpretation":

- $\llbracket e \rrbracket$ and $\llbracket e \rrbracket_R$ form a Galois-connection.

16

Reverse DFA: The $R\text{-MinFP}$ -Approach

The $R\text{-MinFP}$ -Equation System:

$$\mathbf{reqInf}(n) = \begin{cases} c_q & \text{if } n = \mathbf{q} \\ \sqcup \{ \llbracket (n, m) \rrbracket_R(\mathbf{reqInf}(m)) \mid m \in \text{succ}(n) \} & \text{otherwise} \end{cases}$$

The $R\text{-MinFP}$ -Solution:

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

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

17

Standard DFA: The $MaxFP$ -Approach

The $MaxFP$ -Equation System:

$$\mathbf{inf}(n) = \begin{cases} c_s & \text{if } n = \mathbf{s} \\ \sqcap \{ \llbracket (m, n) \rrbracket(\mathbf{inf}(m)) \mid m \in \text{pred}(n) \} & \text{otherwise} \end{cases}$$

The $MaxFP$ -Solution:

$$\forall c_s \in \mathcal{C} \forall n \in N. MaxFP_{(\llbracket \cdot \rrbracket, c_s)}(n) =_{df} \mathbf{inf}_{c_s}^*(n)$$

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

18

The Connecting Link

Link Theorem

For distributive data-flow functionals $\llbracket \cdot \rrbracket$, $q \in N$, and $c_s, c_q \in \mathcal{C}$, we have:

$$R\text{-MinFP}_{c_q}(s) \sqsubseteq c_s \iff MaxFP_{c_s}(q) \sqsupseteq c_q$$

19

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)

20

Essential

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

$$\llbracket p \rrbracket =_{df} \begin{cases} Id_{\mathcal{C}} & \text{if } q < 1 \\ \llbracket \langle e_2, \dots, e_q \rangle \rrbracket \circ \llbracket e_1 \rrbracket & \text{otherwise} \end{cases}$$

21

The MOP-Approach

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

22

Standard DFA: Main Results

Theorem [Soundness / Safety]

$$\forall c_s \in \mathcal{C} \forall n \in \mathbb{N}. MaxFP_{c_s}(n) \sqsubseteq MOP_{c_s}(n)$$

if the data-flow functional $\llbracket \cdot \rrbracket$ is monotonic.

Theorem [Completeness / Coincidence (Precision)]

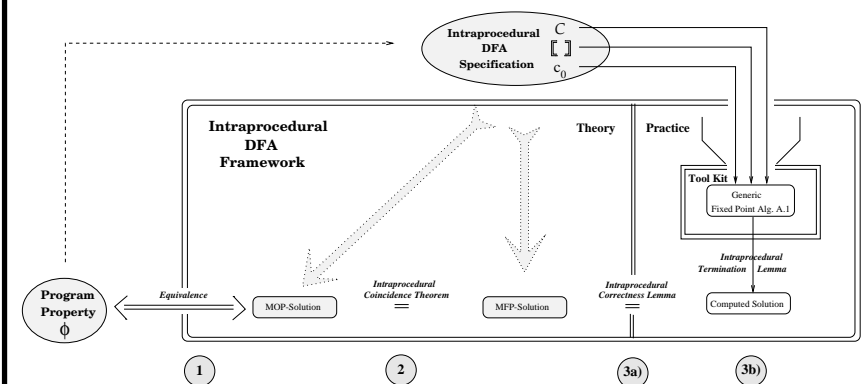
$$\forall c_s \in \mathcal{C} \forall n \in \mathbb{N}. MaxFP_{c_s}(n) = MOP_{c_s}(n)$$

if the data-flow functional $\llbracket \cdot \rrbracket$ is distributive.

23

Standard DFA: The Tool Kit View

...at a glance:



24

Of course...

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

$$\llbracket p \rrbracket_R =_{df} \begin{cases} Id_{\mathcal{C}} & \text{if } q < 1 \\ \llbracket \langle e_1, \dots, e_{q-1} \rangle \rrbracket_R \circ \llbracket e_q \rrbracket_R & \text{otherwise} \end{cases}$$

25

The *R-JOP*-Approach

The *R-JOP*-Solution:

$$\forall c_q \in \mathcal{C} \forall n \in N. R\text{-JOP}_{c_q}(n) =_{df} \sqcup \{ \llbracket p \rrbracket_R(c_q) \mid p \in \mathbf{P}[n, \mathbf{q}] \}$$

26

Reverse DFA: Main Results

Theorem [Soundness / Reverse Safety]

$$\forall c_q \in \mathcal{C} \forall n \in N. R\text{-MinFP}_{c_q}(n) \sqsupseteq R\text{-JOP}_{c_q}(n)$$

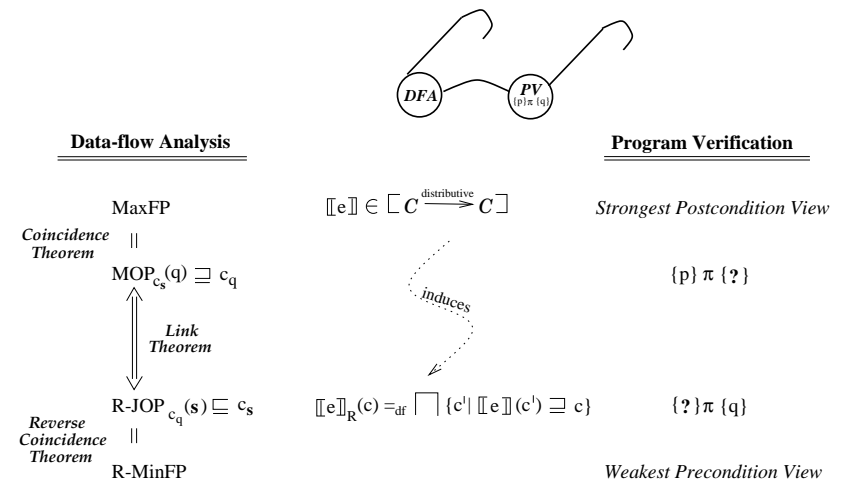
Theorem [Completeness / Reverse Coincidence (Precision)]

$$\forall c_q \in \mathcal{C} \forall n \in N. R\text{-MinFP}_{c_q}(n) = R\text{-JOP}_{c_q}(n)$$

if $\llbracket \cdot \rrbracket$ is distributive.

27

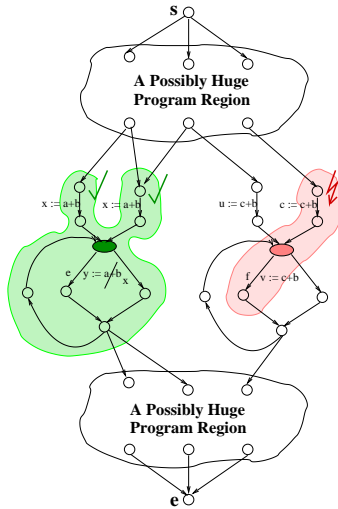
Putting it together...



28

Are We Done?

Recall the motivating example...



29

Mastering the Road to Success

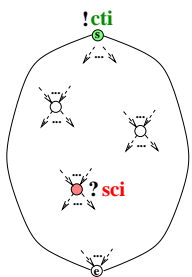
...requires more. It requires us to conclude from “weakest pre-conditions” on “strongest post-conditions”.

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

30

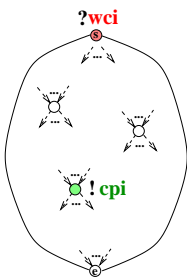
Changing the Perspective

Implementation Problem



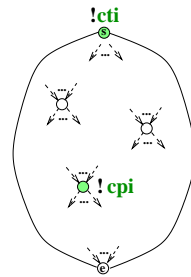
! Given: Context Information *cti*
 ? Sought: Strongest Component Information *sci*

Specification Problem



! Given: Component Information *cpi*
 ? Sought: Weakest Context Information *wci*

Verification Problem



! Given: Context Information *cti*
 Component Information *cpi*
 ? Sought: Validity of *cpi* with respect of *cti*

31

Changing the Perspective: The Standard Taxonomy

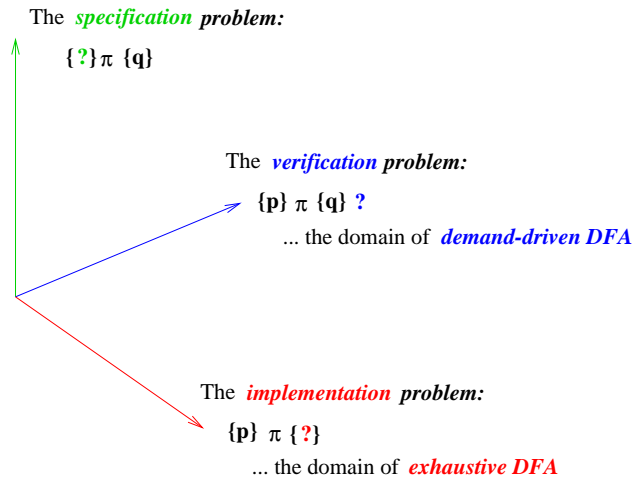
Conventional Classification of DFA Techniques

Exhaustive
DFA

Demand-Driven
DFA

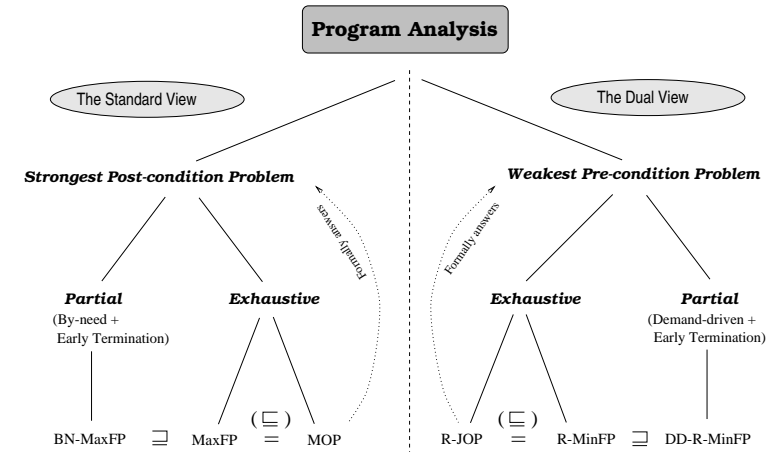
32

Changing the Perspective: Conclusions Derived



33

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



34

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.

35

Concluding the Example: Availability

Abstract semantics for *availability*

1. Data-flow lattice:

$(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top) =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, false, failure)$
 with $\perp = false \sqsubset true \sqsubset failure = \top$

2. Data-flow functional: $\llbracket \cdot \rrbracket_{av} : E \rightarrow (\mathcal{B}_X \rightarrow \mathcal{B}_X)$ defined by

$$\forall e \in E. \llbracket e \rrbracket_{av} =_{df} \begin{cases} Cst_{true}^X & \text{if } Comp_e \wedge Transp_e \\ Id_{\mathcal{B}_X} & \text{if } \neg Comp_e \wedge Transp_e \\ Cst_{false}^X & \text{otherwise} \end{cases}$$

36

Reverse Availability

Reverse abstract semantics for availability

1. Data-flow lattice:

$$(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top) =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, false, failure)$$

2. Reverse data-flow functional: $\llbracket \cdot \rrbracket_{av_R} : E \rightarrow (\mathcal{B}_X \rightarrow \mathcal{B}_X)$

defined by

$$\forall e \in E. \llbracket e \rrbracket_{av_R} =_{df} \begin{cases} R\text{-}Cst_{true}^X & \text{if } \llbracket e \rrbracket_{av} = Cst_{true}^X \\ R\text{-}Id_{\mathcal{B}_X} & \text{if } \llbracket e \rrbracket_{av} = Id_{\mathcal{B}_X} \\ R\text{-}Cst_{false}^X & \text{if } \llbracket e \rrbracket_{av} = Cst_{false}^X \end{cases}$$

37

Supporting Functions

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

(i.e., if $b = failure$)

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

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

38

Summing Up / Extensions

In this talk...

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

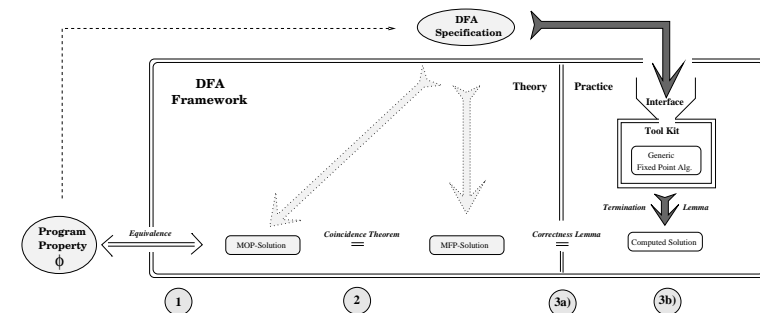
Extensions are possible...

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

39

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

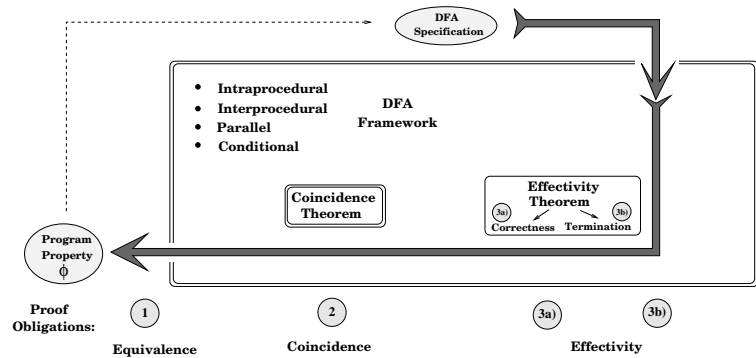
...the general pattern:



40

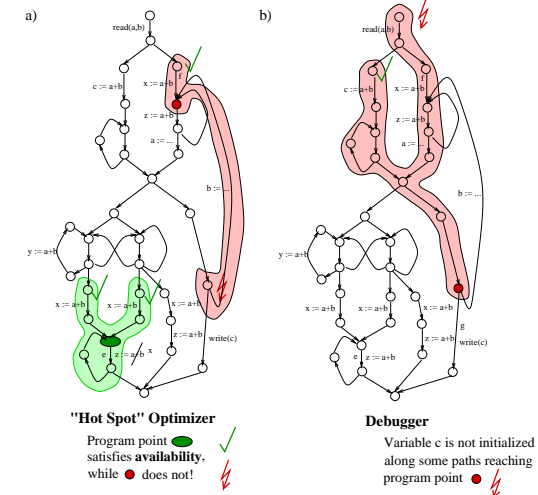
(R)DFA-Frameworks / (R)DFA-Tool Kits

...the general pattern more abstract:



41

Applications



42

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.

43

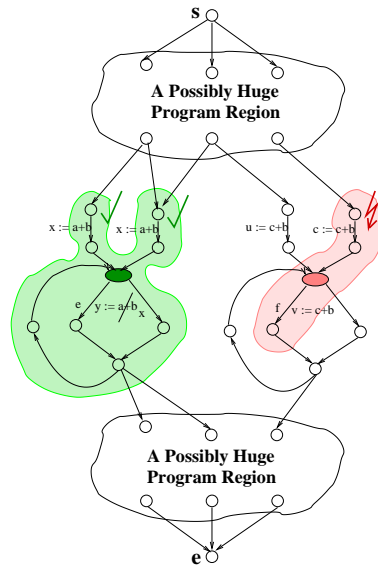
Conclusions (Cont'd)

As an appealing add-on...

- **RDFA is tailored for parallelization!**

44

Recall again...



Conclusions and Perspectives

Data-Flow Analysis for Multi-Core Architectures