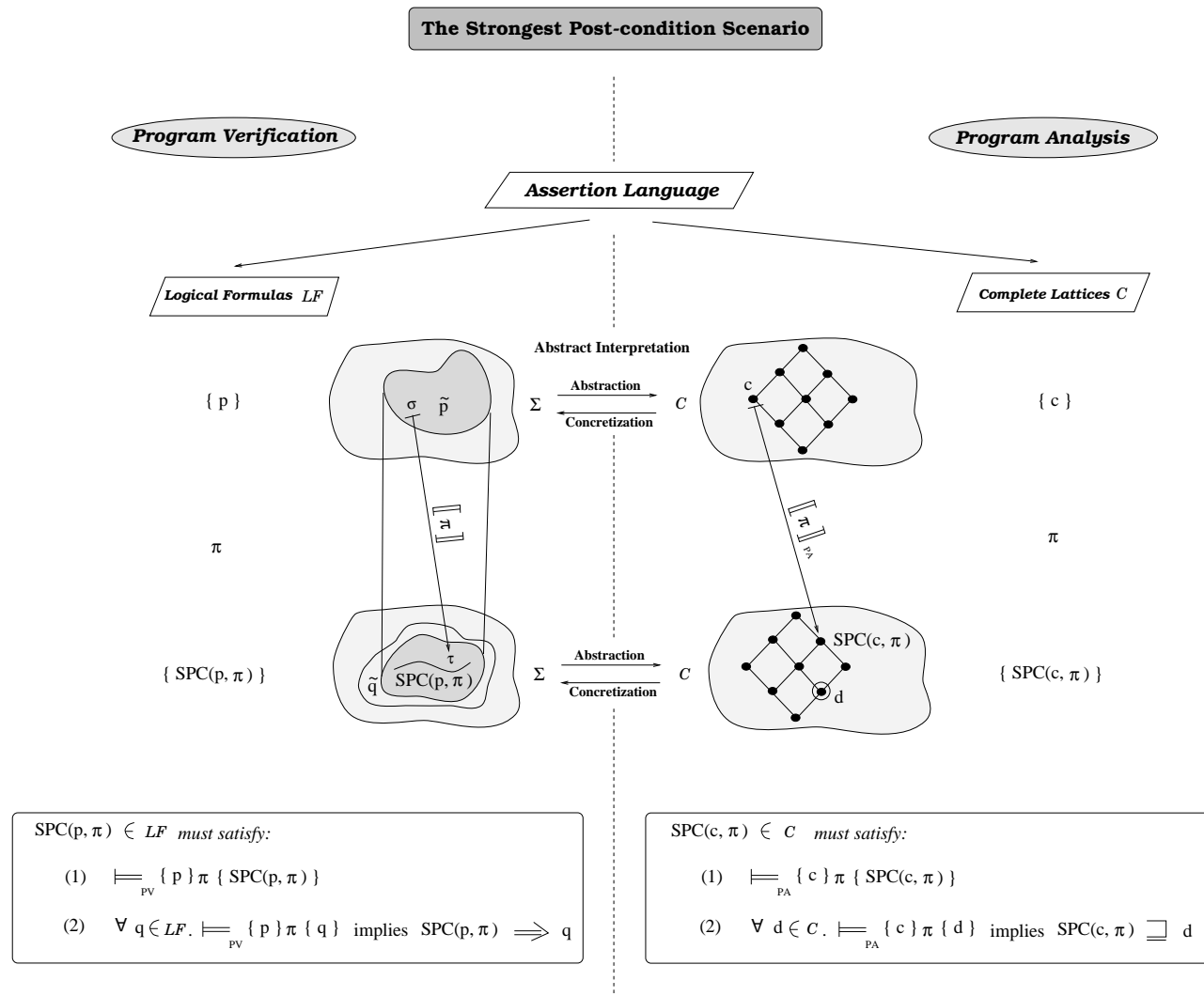
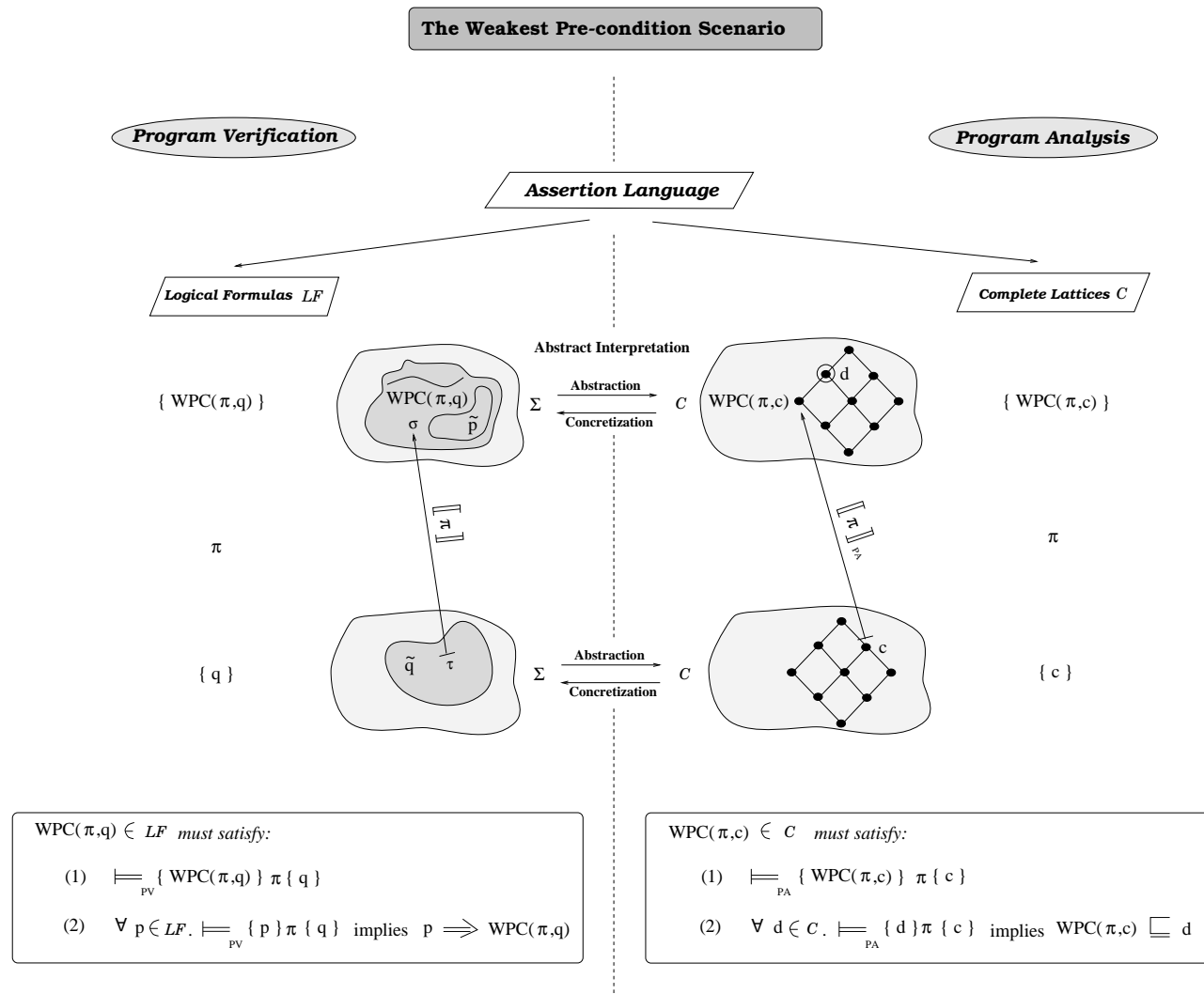


Programmverifikation vs. -analyse (1)



Programmverifikation vs. -analyse (2)



Reverse abstrakte Semantik

Reverse abstrakte Semantik

1. *Datenflussanalyseverband* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top)$

2. *Reverses Datenflussanalysefunktional*

$\llbracket \cdot \rrbracket_R : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$ definiert durch

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

wobei $\llbracket \cdot \rrbracket : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$ eine abstrakte Semantik auf \mathcal{C} ist.

Zusammenhang von $\llbracket \cdot \rrbracket$ und $\llbracket \cdot \rrbracket_R$ (1)

Lemma

Sei $\llbracket \cdot \rrbracket$ ein Datenflussanalysefunktional. Dann gilt für jede Kante $e \in E$:

1. $\llbracket e \rrbracket_R$ ist wohldefiniert und monoton.
2. $\llbracket e \rrbracket_R$ ist additiv, falls $\llbracket e \rrbracket$ distributiv ist.

Zusammenhang von $\llbracket \cdot \rrbracket$ und $\llbracket \cdot \rrbracket_R$ (2)

Lemma

Sei $\llbracket \cdot \rrbracket$ ein Datenflussanalysefunktional. Dann gilt für jede Kante $e \in E$:

1. $\llbracket e \rrbracket_R \circ \llbracket e \rrbracket \sqsubseteq Id_C$, falls $\llbracket e \rrbracket$ monoton ist.
2. $\llbracket e \rrbracket \circ \llbracket e \rrbracket_R \sqsupseteq Id_C$, falls $\llbracket e \rrbracket$ distributiv ist.

Sprechweise in der Theorie “Abstrakter Interpretation”:

- $\llbracket e \rrbracket$ und $\llbracket e \rrbracket_R$ bilden eine Galois-Verbindung.

Zusammenhang von $\llbracket \cdot \rrbracket$ und $\llbracket \cdot \rrbracket_R$ (3)

Hilfssatz

1. $\forall n \in N' \cap N. P_{G'}[s, n] = P_G[s, n]$
2. $\forall q \in N' \setminus \{s\}. P_{G'}[s, q] = P_G[s, q]$
3. $\forall c_s \in \mathcal{C} \forall n \in N' \cap N. MOP_{(G', c_s)}(n) = MOP_{(G, c_s)}(n)$
4. $MOP_{(G, c_s)}(q) = MOP_{(G, c_s)}(\mathbf{q})$

Der R -JOP-Ansatz

Die R -JOP-Lösung:

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

Der R -MinFP-Ansatz

Das R -MinFP-Gleichungssystem:

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

Bezeichne $\mathbf{reqInf}_{c_q}^*$ die kleinste Lösung dieses Gleichungssystems bzgl. $c_q \in \mathcal{C}$.

Die R -MinFP-Lösung:

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

Der generische R -MinFP-Alg. (1)

Input: (1) A flow graph $G = (N, E, s, e)$, (2) a program point q , (3) a reverse abstract semantics (i.e., a data-flow lattice \mathcal{C} , and a reverse data-flow functional $\llbracket \cdot \rrbracket_R : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$ induced by a functional $\llbracket \cdot \rrbracket : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$), and (4) a component information $c_q \in \mathcal{C}$.

Output: Under the assumption of termination (cf. Theorem ??), the R -MinFP-solution. Depending on the properties of the underlying reverse data-flow functional, this has the following interpretation.

(1) $\llbracket \cdot \rrbracket_R$ is additive: Variable $reqInf[s]$ stores the weakest context information of c_q , i.e., the least data-flow fact which must be ensured at the program entry in order to guarantee c_q at q . If this is \top , the requested component information cannot be satisfied at all.

(2) $\llbracket \cdot \rrbracket_R$ is monotonic: Variable $reqInf[s]$ stores a lower bound of the weakest context candidate of c_q . Generally, this is not a sufficient context information itself. Hence, except for the special case $reqInf[s] = \top$, which implies that c_q cannot be satisfied by any consistent context information, nothing can be concluded from the value of $reqInf[s]$.

Remark: The variable *workset* controls the iterative process. Its elements are nodes of G , whose informations annotating them have recently been updated.

Der generische *R-MinFP*-Alg. (2)

(Prologue: Initialization of the annotation array *reqInf*, and the variable *workset*)

```
FORALL  $n \in N \setminus \{q\}$  DO  $reqInf[n] := \perp$  OD;  
 $reqInf[q] := c_q$ ;  
 $workset := \{q\}$ ;
```

Der generische R -MinFP-Alg. (3)

(Main process: Iterative fixed point computation)

```
WHILE  $workset \neq \emptyset$  DO
  CHOOSE  $m \in workset$ ;
   $workset := workset \setminus \{m\}$ ;
  ( Update the predecessor-environment of node  $m$  )
  FORALL  $n \in pred(m)$  DO
     $join := \llbracket (n, m) \rrbracket_R(reqInf[m]) \sqcup reqInf[n]$ ;
    IF  $reqInf[n] \sqsubset join$ 
      THEN
         $reqInf[n] := join$ ;
         $workset := workset \cup \{n\}$ 
      FI
    OD
  ESOOHC
OD.
```

Reverses Sicherheitstheorem

Reverses Sicherheitstheorem

Die *R-MinFP*-Lösung ist eine obere (d.h. sichere) Approximation der *R-JOP*-Lösung, d.h.,

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

Reverses Koinzidenztheorem

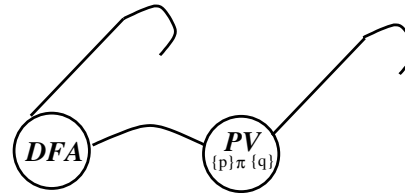
Reverses Koinzidenztheorem

Die R - $MinFP$ -Lösung stimmt mit der R - JOP -Lösung überein, d.h.,

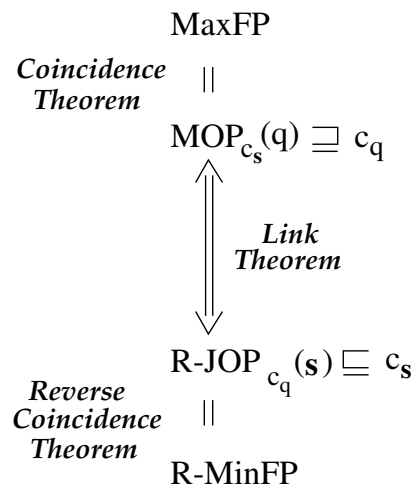
$$\forall c_q \in \mathcal{C} \ \forall n \in N. R\text{-}MinFP_{c_q}(n) = R\text{-}JOP_{c_q}(n)$$

falls $\llbracket \cdot \rrbracket$ distributiv ist.

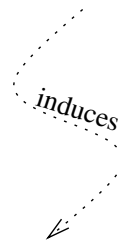
DFA vs. Verifikation: Überblick



Data-flow Analysis



$$\llbracket e \rrbracket \in \llbracket C \xrightarrow{\text{distributive}} C \rrbracket$$



$$\llbracket e \rrbracket_R(c) =_{\text{df}} \bigsqcap \{c' \mid \llbracket e \rrbracket(c') \sqsupseteq c\}$$

Program Verification

Strongest Postcondition View

$$\{p\} \pi \{?\}$$

$$\{?\} \pi \{q\}$$

Weakest Precondition View

Drei unterschiedliche Problemperspektiven (1)

The *specification* problem:

$\{?\} \pi \{q\}$

The *verification* problem:

$\{p\} \pi \{q\} ?$

... the domain of *demand-driven DFA*

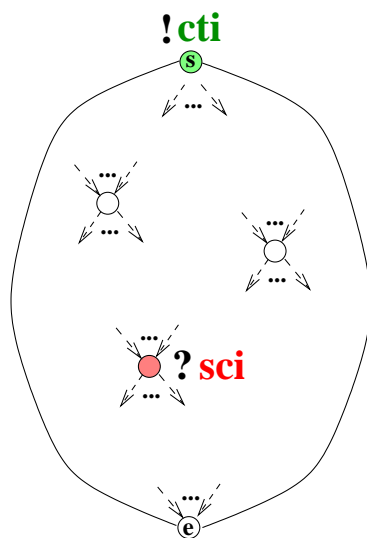
The *implementation* problem:

$\{p\} \pi \{?\}$

... the domain of *exhaustive DFA*

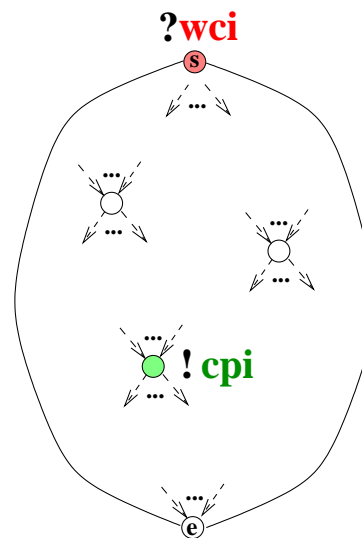
Drei unterschiedliche Problemperspektiven (2)

Implementation Problem



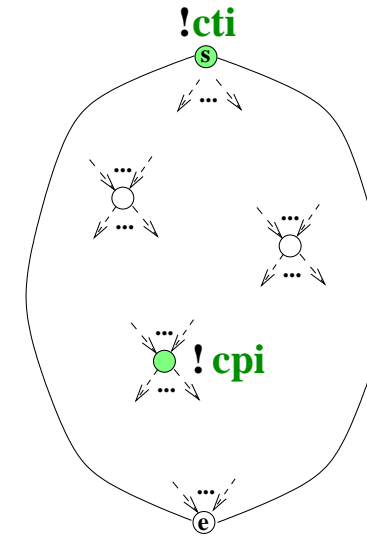
! **Given:** Context Information **cti**
 ? **Sought:** Strongest Component Information **sci**

Specification Problem



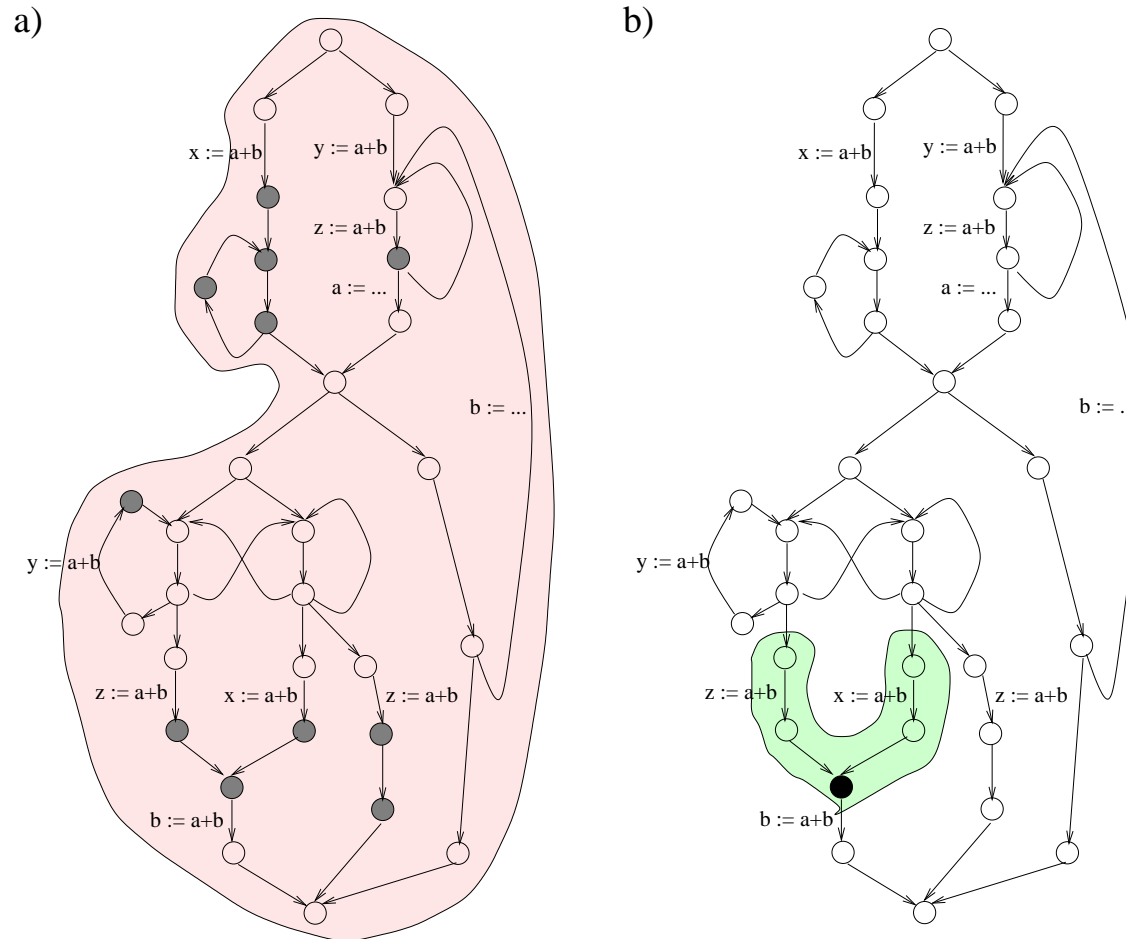
! **Given:** Component Information **cpi**
 ? **Sought:** Weakest Context Information **wci**

Verification Problem

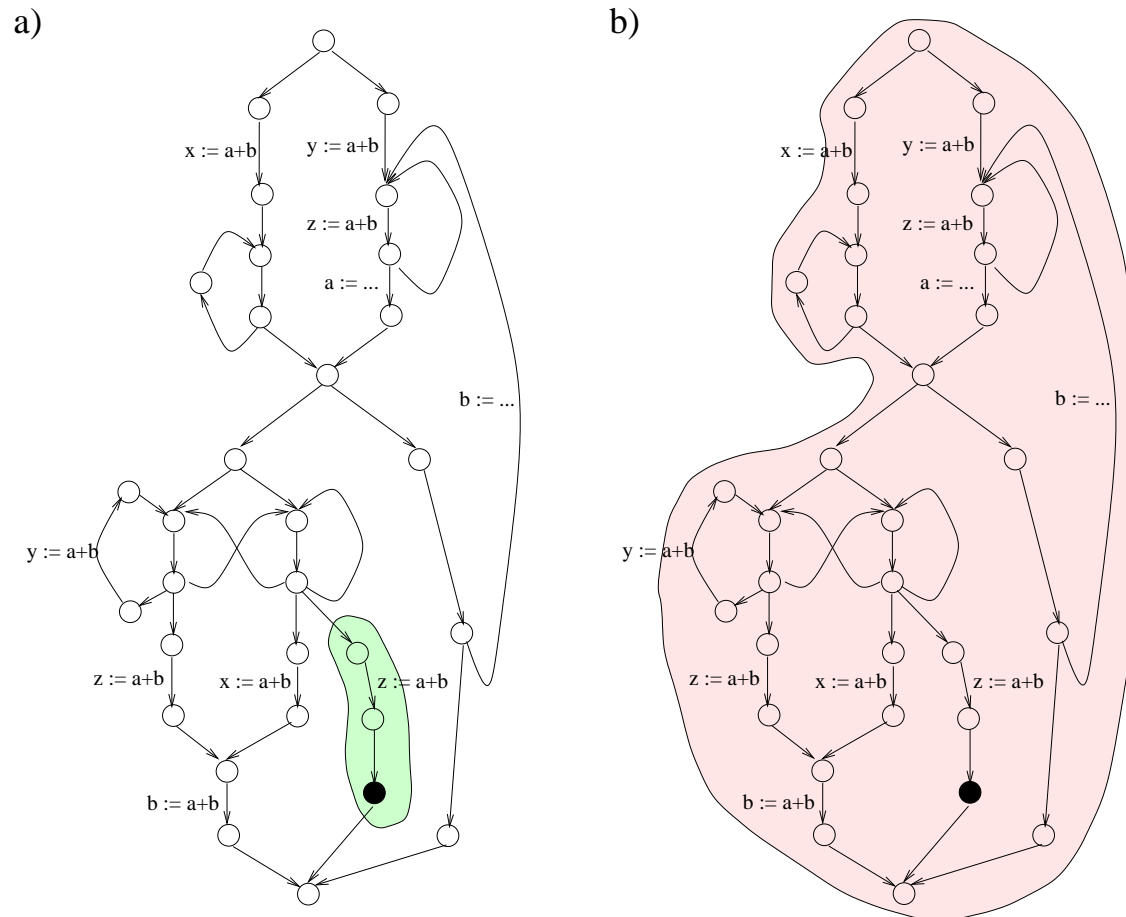


! **Given:** Context Information **cti**
 Component Information **cpi**
 ? **Sought:** Validity of **cpi** with respect of **cti**

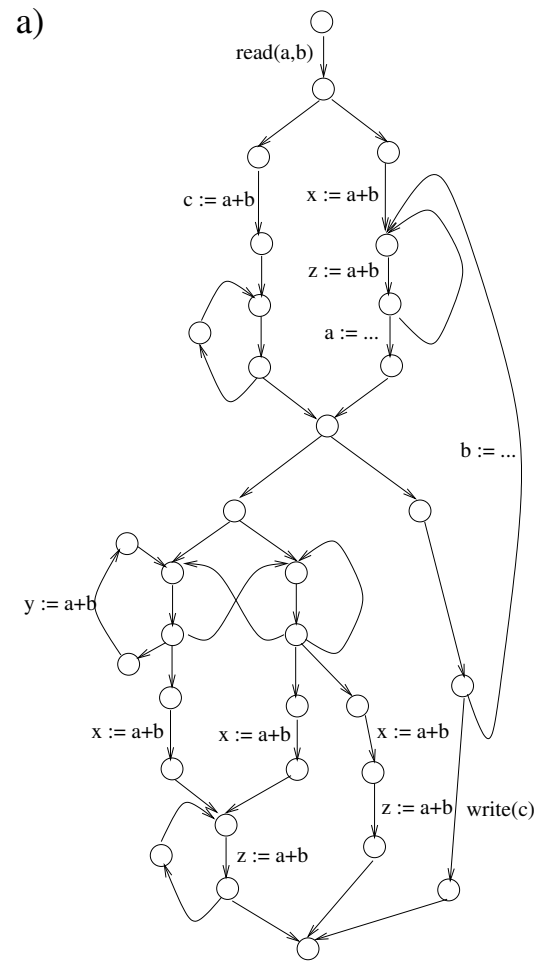
Bsp: Verfügbarkeit an einem Punkt (1)



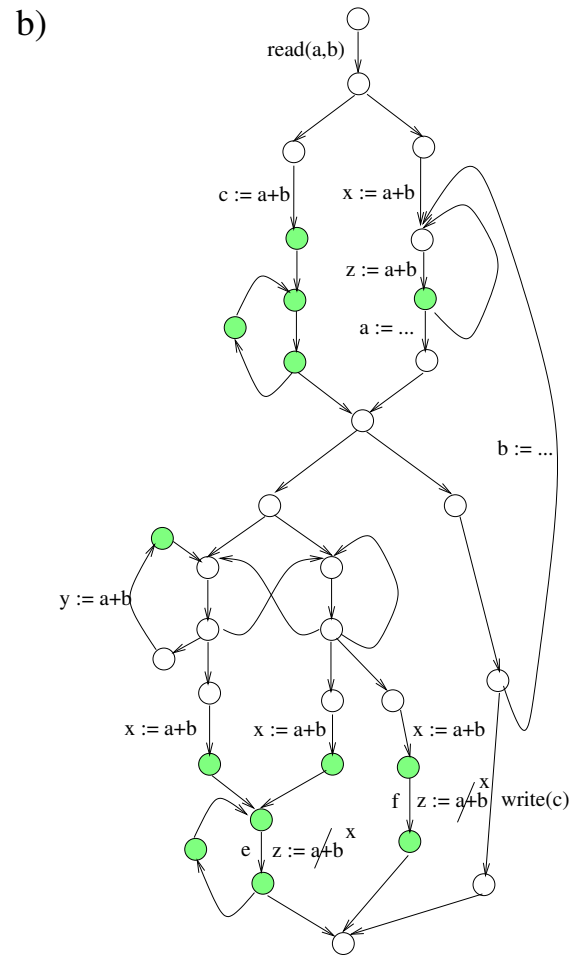
Bsp: Verfügbarkeit an einem Punkt (2)



Anwendung: Einfacher Optimierer (1)



Flow graph

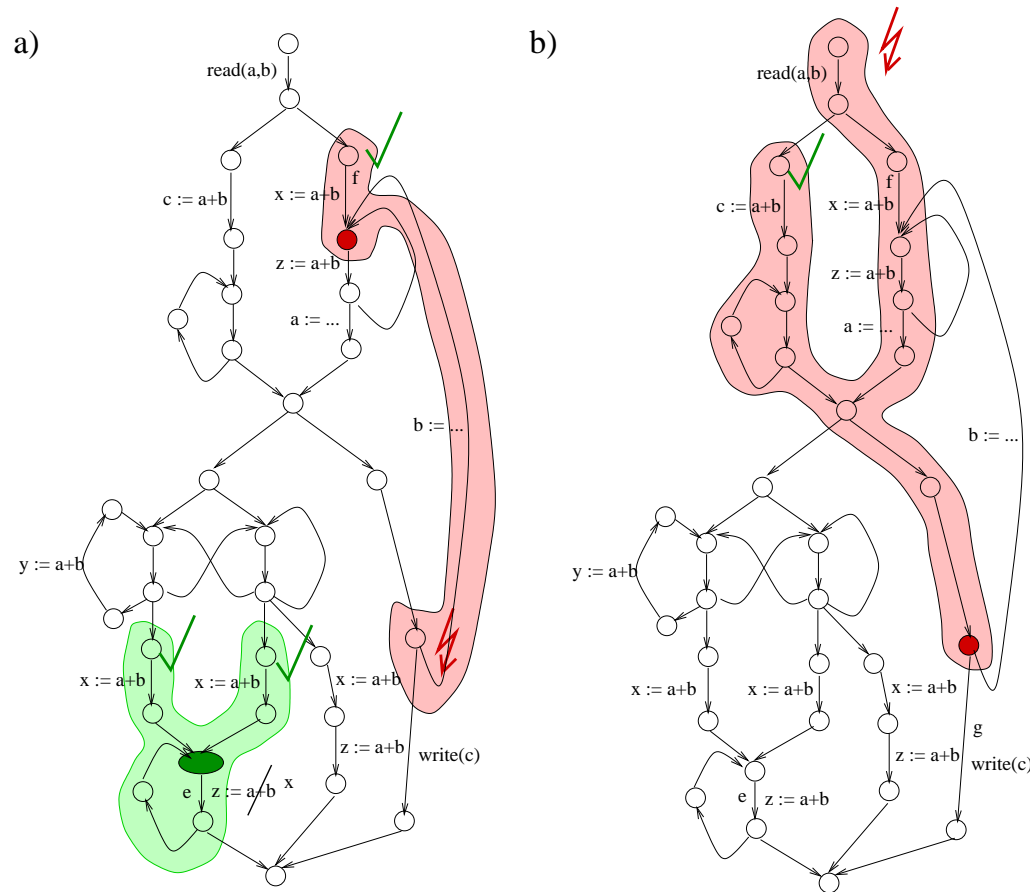


Data-flow information

Program points

satisfying **availability** ●

Anwendung: "Hot Spot" Optimierer und Debugger (2)



"Hot Spot" Optimizer

Program point ● satisfies **availability**,
while ● does not!

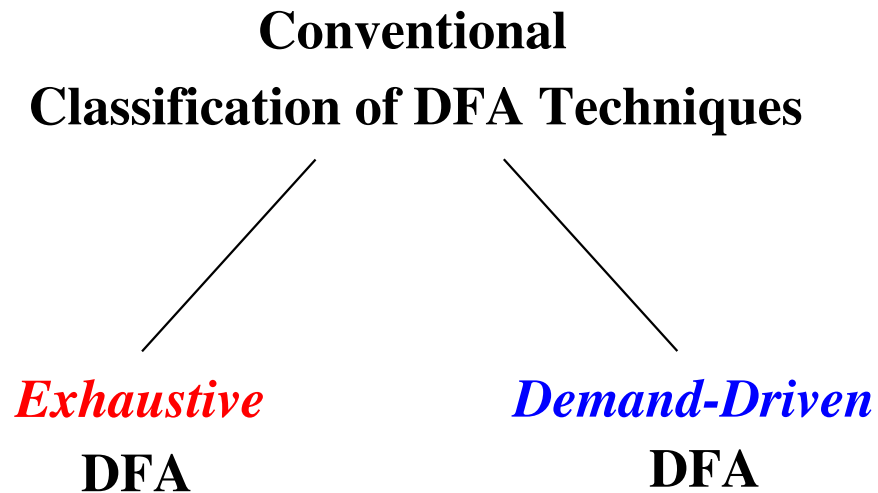


Debugger

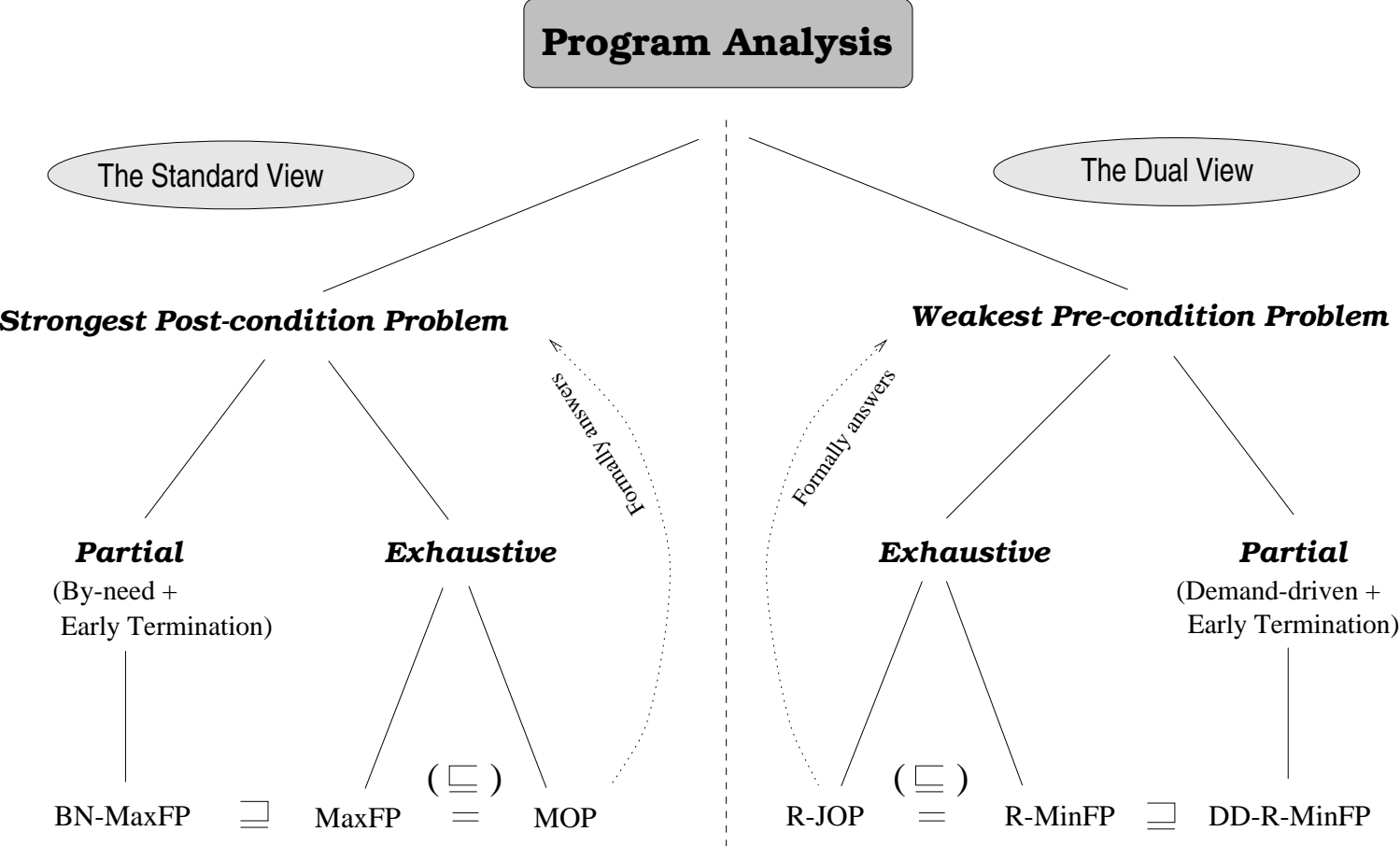
Variable **c** is not initialized
along some paths reaching
program point ●



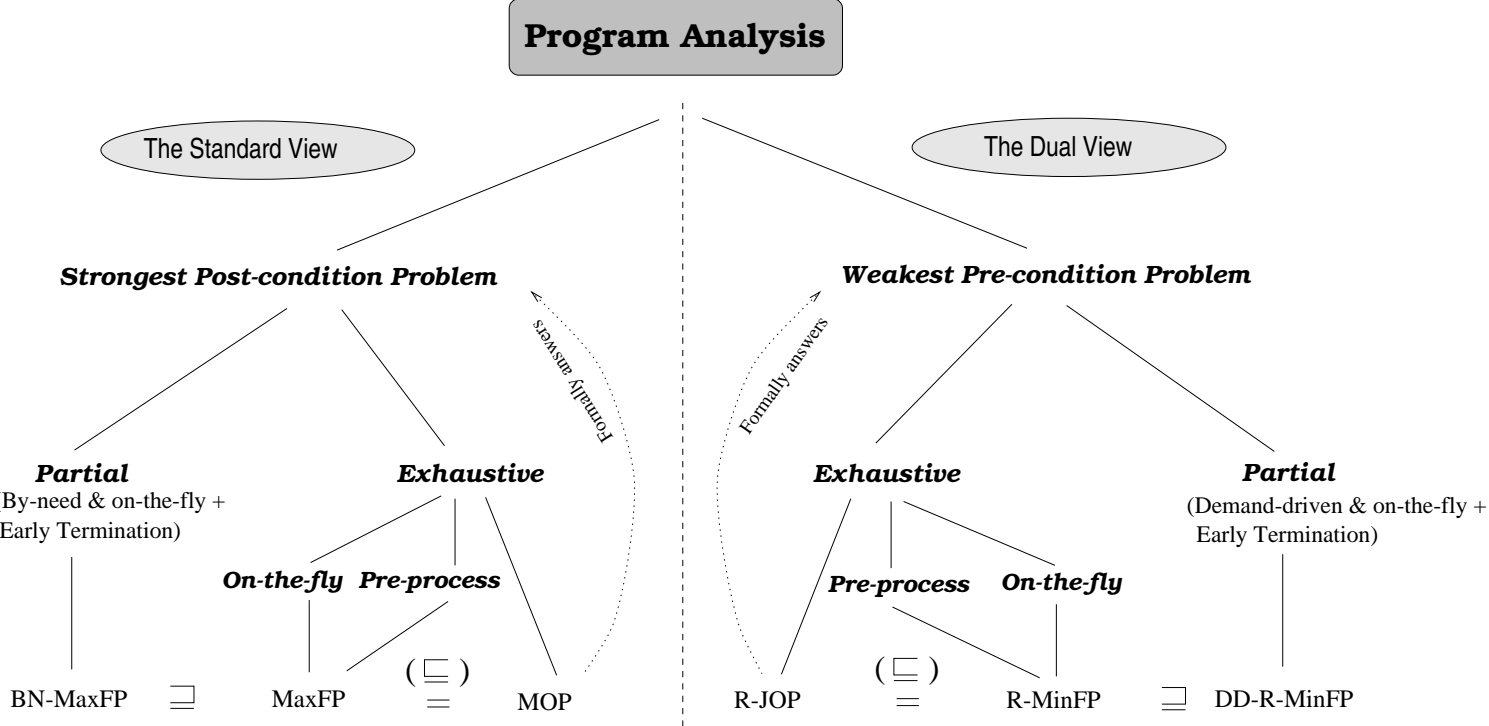
Erschöpfende vs. anforderungsgetriebene DFA (1)



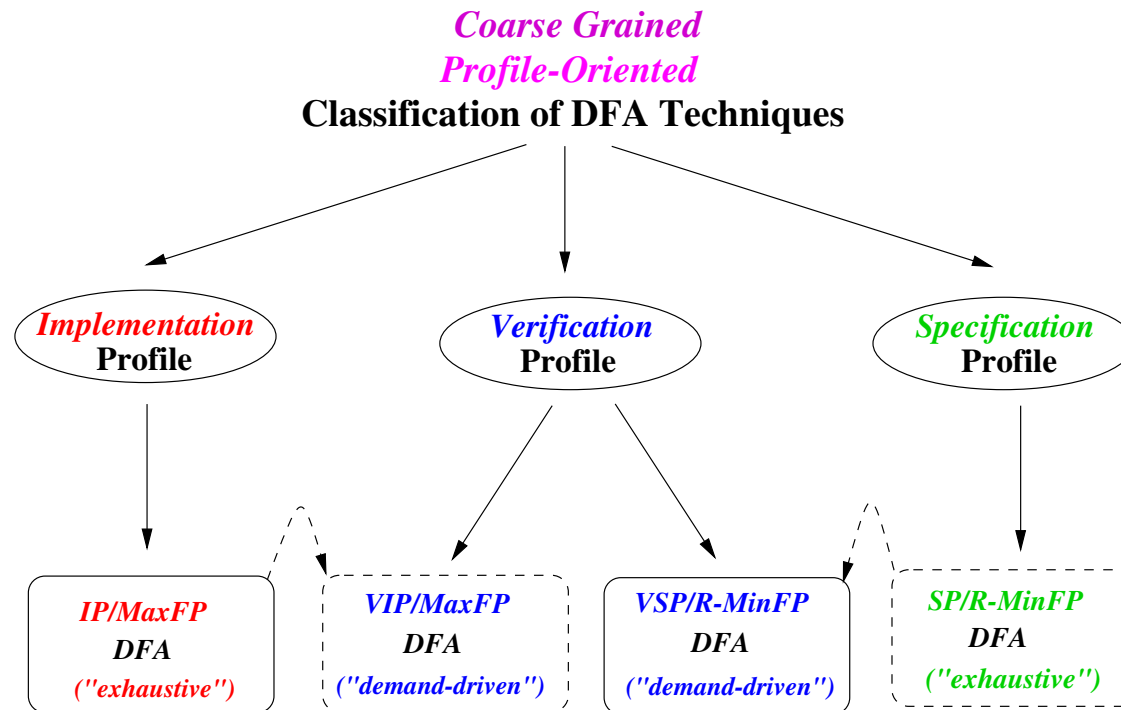
Erschöpfende vs. anforderungsgetriebene DFA (2)



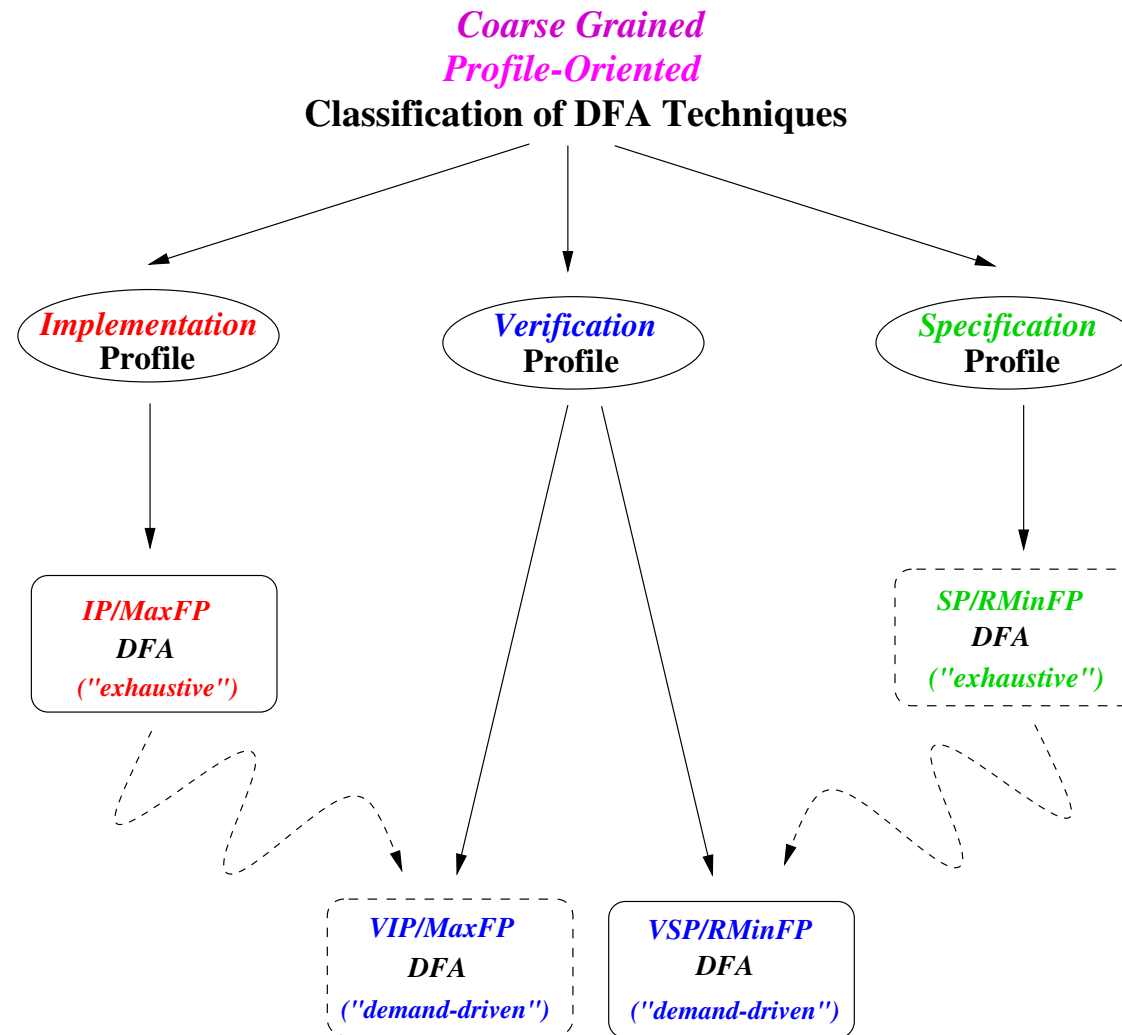
Erschöpfende vs. anforderungsgetriebene DFA (3)



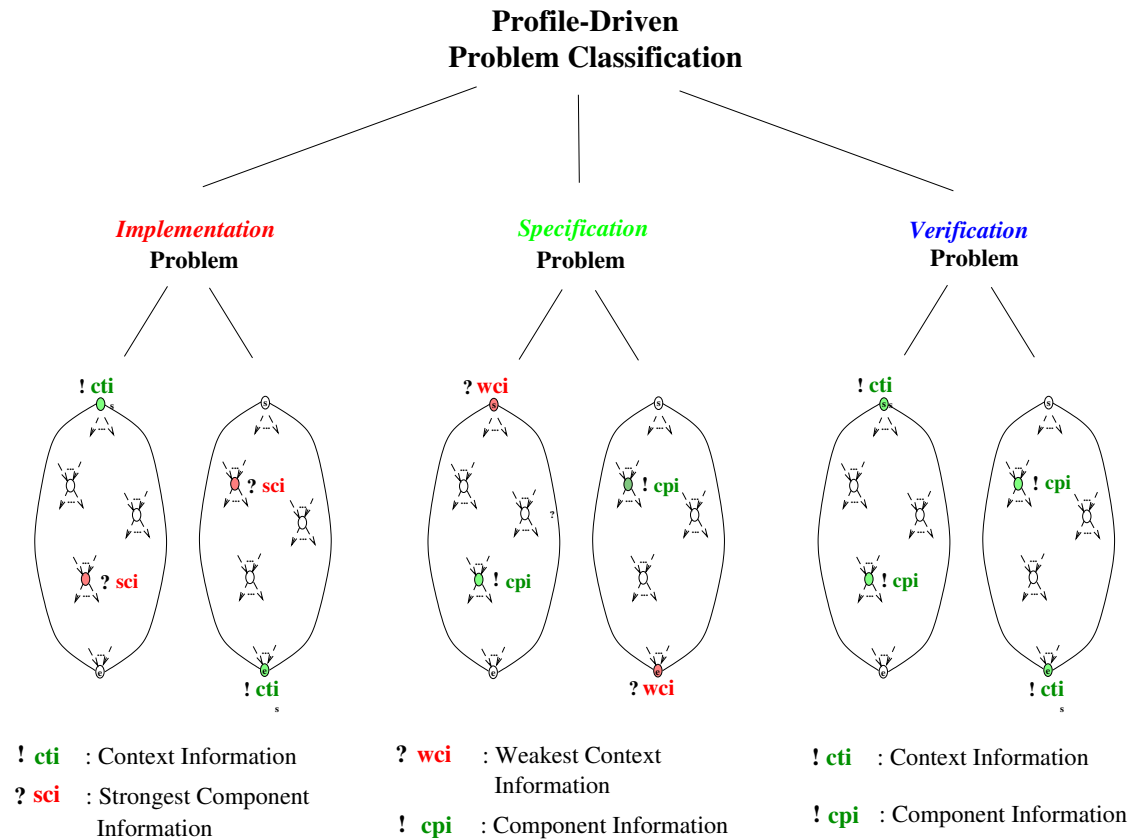
Eine andere Sicht (1)



Eine andere Sicht (2)

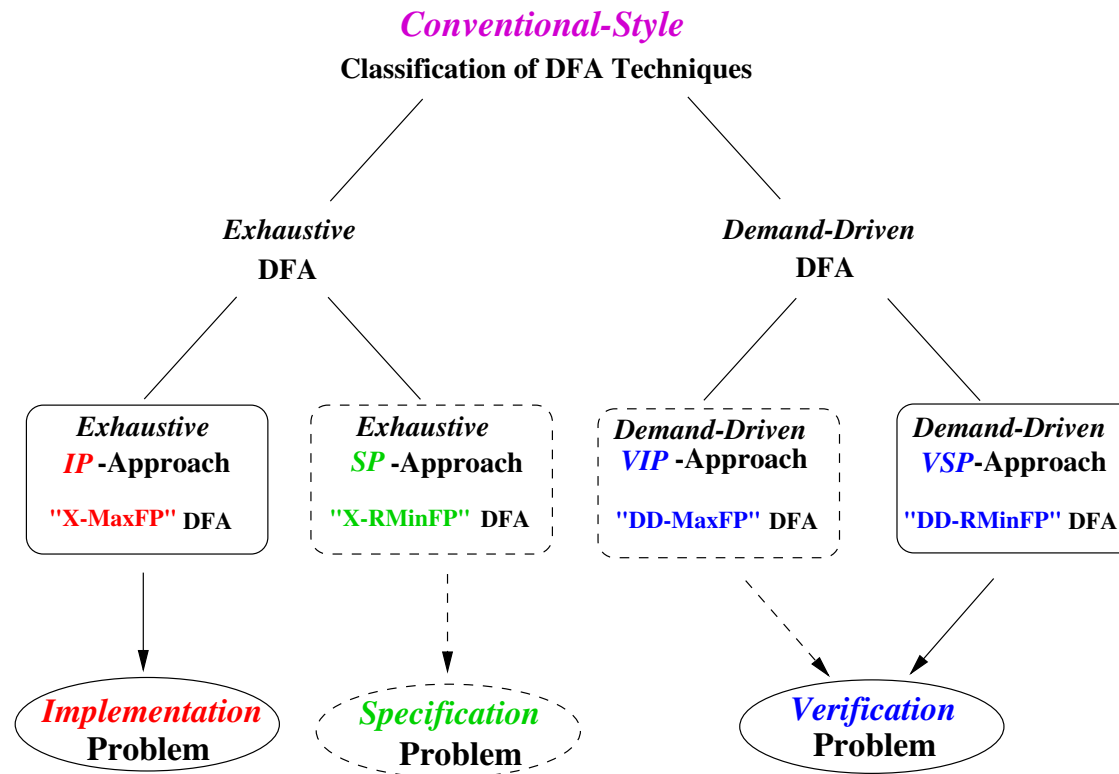


Im Überblick



... where "!" and "?" means **given** and **sought**, respectively.

Zum Abschluss: Algorithmenorientiert (1)



Zum Abschluss: Problemorientiert (2)

