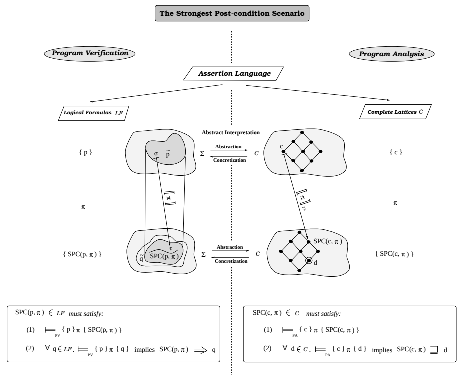
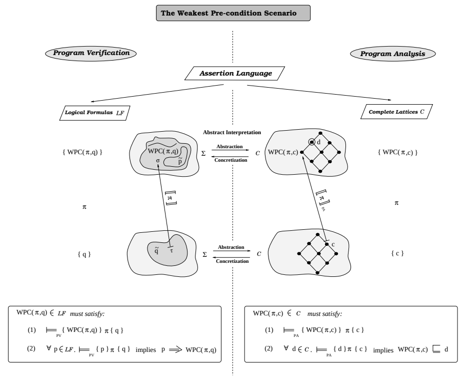
Programmverifikation vs. -analyse (1)



Programmverifikation vs. -analyse (2)



Reverse abstrakte Semantik

Reverse abstrakte Semantik

1. *Datenflussanalyseverband* $\widehat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$
2. *Reverses Datenflussanalysefunktional* $[\![\ ]\!]_R : E \to (\mathcal{C} \to \mathcal{C})$ definiert durch

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

wobei $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$ eine abstrakte Semantik auf $\mathcal{C}$ ist.

Zusammenhang von $[\![\ ]\!]$ und $[\![\ ]\!]_R$ (1)

Lemma

Sei $[\![\ ]\!]$ ein Datenflussanalysefunktional. Dann gilt für jede Kante $e \in E$:

1. $[\![\, e\,]\!]_R$ ist wohldefiniert und monoton.
2. $[\![\, e\,]\!]_R$ ist additiv, falls $[\![\, e\,]\!]$ distributiv ist.

Zusammenhang von $[\![\ ]\!]$ und $[\![\ ]\!]_R$ (2)

Lemma

Sei $[\![\ ]\!]$ ein Datenflussanalysefunktional. Dann gilt für jede Kante $e \in E$:

1. $[\![\, e\,]\!]_R \circ [\![\, e\,]\!] \sqsubseteq Id_{\mathcal{C}}$, falls $[\![\, e\,]\!]$ monoton ist.
2. $[\![\, e\,]\!] \circ [\![\, e\,]\!]_R \sqsupseteq Id_{\mathcal{C}}$, falls $[\![\, e\,]\!]$ distributiv ist.

Sprechweise in der Theorie "Abstrakter Interpretation":

- $[\![\, e\,]\!]$ und $[\![\, e\,]\!]_R$ bilden eine Galois-Verbindung.

Zusammenhang von $[\![\ ]\!]$ und $[\![\ ]\!]_R$ (3)

Hilfssatz

1. $\forall\, n \in N' \cap N.\ \mathbf{P}_{G'}[\mathbf{s}, n] = \mathbf{P}_G[\mathbf{s}, n]$
2. $\forall\, q \in N' \backslash \{\mathbf{s}\}.\ \mathbf{P}_{G'}[\mathbf{s}, q] = \mathbf{P}_G[\mathbf{s}, \mathbf{q}]$
3. $\forall\, c_{\mathbf{s}} \in \mathcal{C}\ \forall\, n \in N' \cap N.\ MOP_{(G', c_{\mathbf{s}})}(n) = MOP_{(G, c_{\mathbf{s}})}(n)$
4. $MOP_{(G, c_{\mathbf{s}})}(q) = MOP_{(G, c_{\mathbf{s}})}(\mathbf{q})$

Der R-JOP-Ansatz

Die R-JOP-Lösung:

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

Der R-$MinFP$-Ansatz

Das R-$MinFP$-Gleichungssystem:

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

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

Die R-$MinFP$-Lösung:

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

Der generische R-$MinFP$-Alg. (1)

Input: (1) A flow graph $G=(N,E,\mathbf{s},\mathbf{e})$, (2) a program point q, (3) a reverse abstract semantics (i.e., a data-flow lattice $\mathcal{C}$, and a reverse data-flow functional $[\![\ ]\!]_R : E \to (\mathcal{C}\to\mathcal{C})$ induced by a functional $[\![\ ]\!] : E \to (\mathcal{C}\to\mathcal{C})$), 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) $[\![\ ]\!]_R$ *is additive:* Variable *reqInf*[$\mathbf{s}$] 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) $[\![\ ]\!]_R$ *is monotonic:* Variable *reqInf*[$\mathbf{s}$] 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*[$\mathbf{s}$] $=\top$, which implies that c_q cannot be satisfied by any consistent context information, nothing can be concluded from the value of *reqInf*[$\mathbf{s}$].

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 $n \in N\backslash\{\mathbf{q}\}$ DO *reqInf*$[n] := \bot$ OD;
reqInf$[\mathbf{q}] := c_q$;
workset $:= \{\mathbf{q}\}$;

Der generische R-$MinFP$-Alg. (3)

(Main process: Iterative fixed point computation)

```
WHILE workset ≠ ∅ DO
   CHOOSE m ∈ workset;
     workset := workset\{m};
     ( Update the predecessor-environment of node m )
     FORALL n ∈ pred(m) DO
       join := [[ (n,m) ]]_R(reqInf[m]) ⊔ reqInf[n];
       IF reqInf[n] ⊏ join
         THEN
           reqInf[n] := join;
           workset := workset ∪ {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.,

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

Reverses Koinzidenztheorem

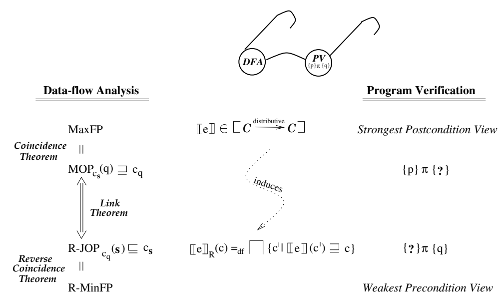
Reverses Koinzidenztheorem

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

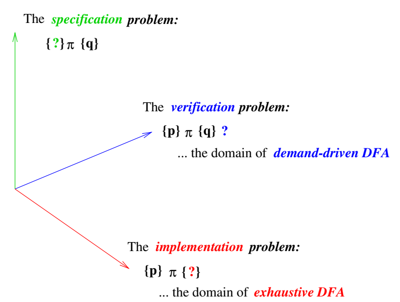
$$\forall\, c_q \in \mathcal{C}\ \forall\, n \in N.\ R\text{-}MinFP_{c_q}(n) = R\text{-}JOP_{c_q}(n)$$

falls $[\![\ ]\!]$ distributiv ist.

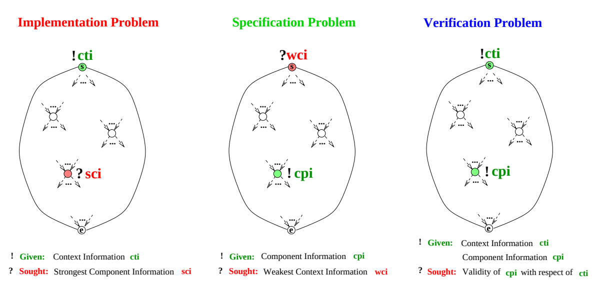
DFA vs. Verifikation: Überblick



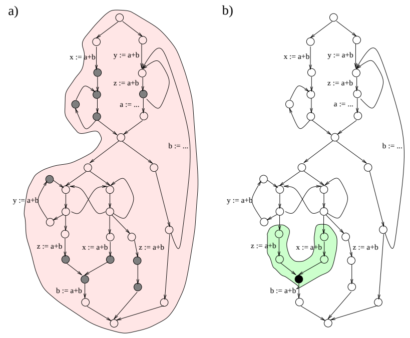
Drei unterschiedliche Problemperspektiven (1)



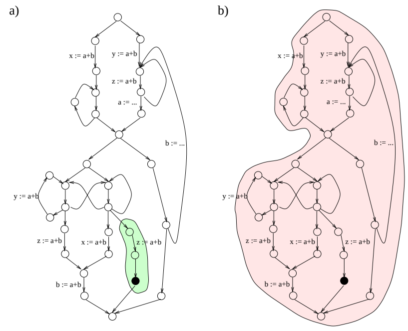
Drei unterschiedliche Problemperspektiven (2)



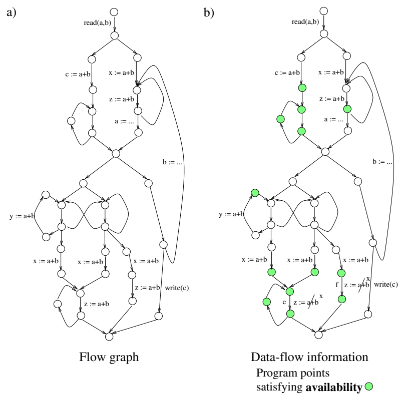
Bsp: Verfügbarkeit an einem Punkt (1)



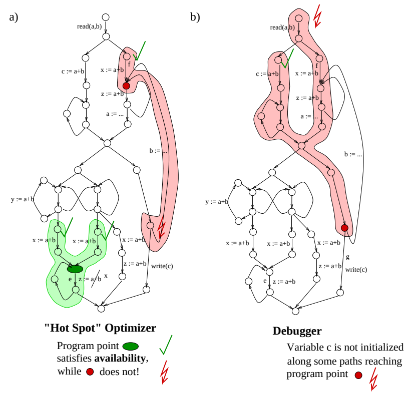
Bsp: Verfügbarkeit an einem Punkt (2)



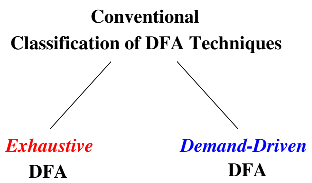
Anwendung: Einfacher Optimierer (1)



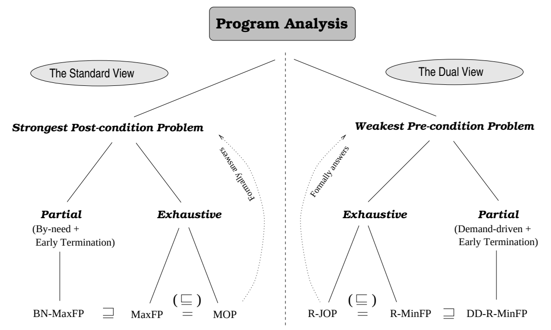
Anwendung: “Hot Spot” Optimierer und Debugger (2)



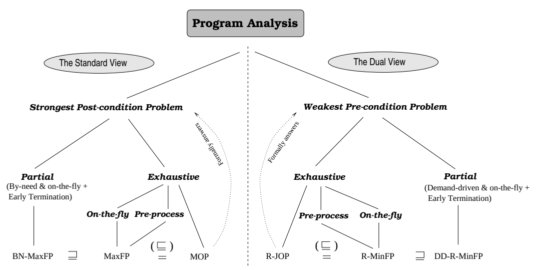
Erschöpfende vs. anforderungsgetriebene DFA (1)



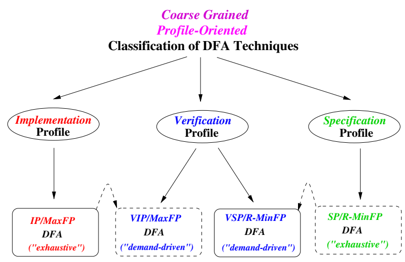
Erschöpfende vs. anforderungsgetriebene DFA (2)



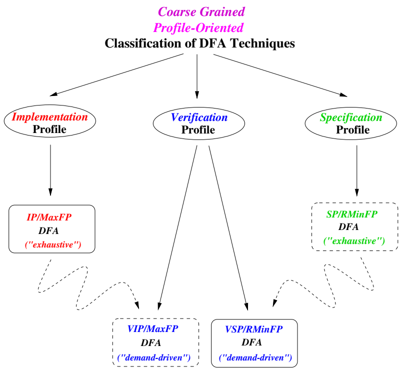
Erschöpfende vs. anforderungsgetriebene DFA (3)



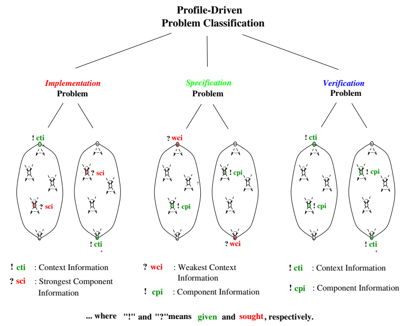
Eine andere Sicht (1)



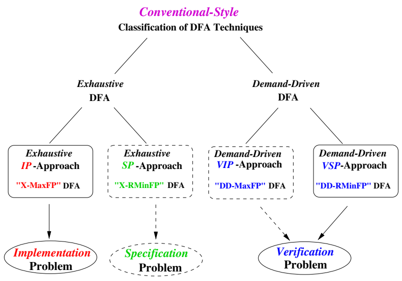
Eine andere Sicht (2)



Im Überblick



Zum Abschluss: Algorithmenorientiert (1)



Zum Abschluss: Problemorientiert (2)