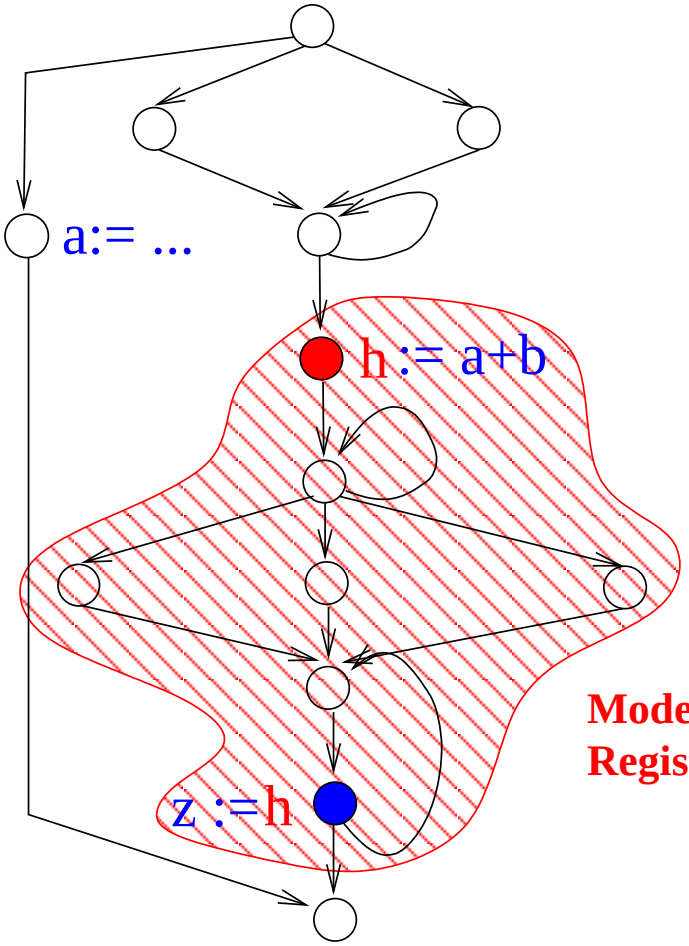
There is more than speed!

...z.B. Platz!



Moderate Register Pressure!

Der Weltmarkt für Mikroprozessoren 1999

Chip-Kategorie	Verkaufte Anzahl
Eingebettet 4-bit	2000 Millionen
Eingebettet 8-bit	4700 Millionen
Eingebettet 16-bit	700 Millionen
Eingebettet 32-bit	400 Millionen
DSP	600 Millionen
Desktop 32/64-bit	150 Millionen

... David Tennenhouse (Intel Director of Research). Hauptvortrag auf dem *20th IEEE Real-Time Systems Symposium* (*RTSS'99*), Phoenix, Arizona, Dezember 1999.

Der Weltmarkt für Mikroprozessoren 1999

Chip-Kategorie	Verkaufte Anzahl
Eingebettet 4-bit	2000 Millionen
Eingebettet 8-bit	4700 Millionen
Eingebettet 16-bit	700 Millionen
Eingebettet 32-bit	400 Millionen
DSP	600 Millionen
Desktop 32/64-bit	150 Millionen

$\sim$ **2%**

... David Tennenhouse (Intel Director of Research). Hauptvortrag auf dem *20th IEEE Real-Time Systems Symposium* (*RTSS'99*), Phoenix, Arizona, Dezember 1999.

Man denke an...

... domänenspezifische Prozessoren eingesetzt in eingebetteten Systemen

- **Telekommunikation**
  - Mobiltelefone, Pager, ...
- **Heimelektronik**
  - MP3-Spieler, Kameras, Spielekonsolen, ...
- **Automobilbereich**
  - GPS-Navigation, Airbags, ...
- ...

Code für eingebettete Systeme

Anforderungen...

- Performanz (oft Echtzeitanforderungen)
- Codegröße (system-on-chip, on-chip RAM/ROM)
- ...

Für eingebettete Systeme...

...Codegröße ist oft eine kritischere Größe als Geschwindigkeit!

Code für eingebettete Systeme (Fortsetzung)

Anforderungen (und wie sie häufig adressiert werden...):

- Assemblerprogrammierung
- Händische Postoptimierung

Schwächen...

- Fehlerträchtig
- Verzögerte Marktreife/-eintritt

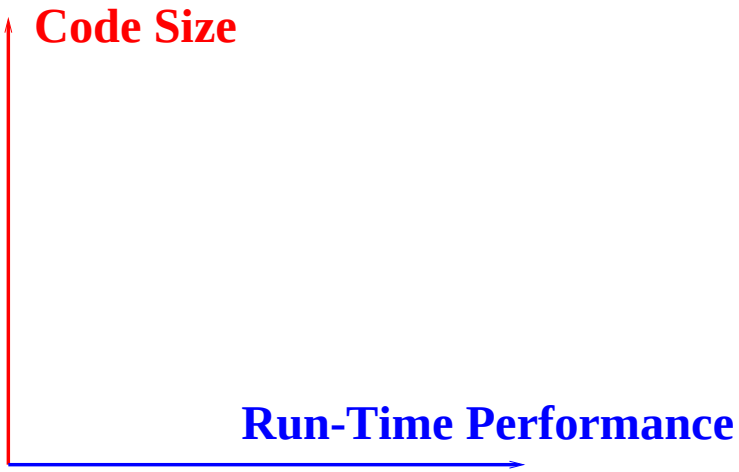
...die Probleme werden zunehmend größer mit wachsender Komplexität.

Generell beobachtet man...

...einen Trend hin zu Hochsprachenprogrammierung (C/C++)

Angesichts dieses Trends...

...wie unterstützen traditionelle Übersetzer- und Optimierertechnologien das spezielle Anforderungsprofil von Code für eingebettete Systeme?



Leider nur wenig.

Unbestritten...

Traditionelle Optimierungen...

- ...sind nahezu ausschließlich auf Performanzoptimierung getrimmt
- ...sind nicht codegrößensensitiv und bieten i.a. keinerlei Kontrolle über ihren Einfluss auf die Codegröße

In besonderer Weise...

...gilt dies für Optimierungen, die auf Codeverschiebung beruhen

Dazu gehören insbesondere

- Partial redundancy elimination
- Partial dead-code elimination
- Partial redundant-assignment elimination
- Strength reduction
- ...

Erinnerung am Beispiel von PRE

PRE kann konzeptuell als zweistufiger Prozess gesehen werden...

1. Ausdrucksvorziehen
   ...vorziehen von Ausdrücken an "frühere" sichere Berechnungspunkte

2. Totale Redundanzelimination
   ...eliminieren von Berechnungen, die durch das Vorziehen total redundant geworden sind

Erinnerung am Beispiel von LCM

LCM kann konzeptuell als Ergebnis eines zweistufigen Prozesses gesehen werden...

1. Ausdrucksvorziehen (hoisting expressions)
   ...zu ihren "frühesten" sicheren Berechnungspunkten

2. Ausdrucksverzögern (sinking expressions)
   ...zu ihren "spätesten" sicheren und berechnungsoptimalen Berechnungspunkten

Auf dem Weg zu codegrößensensitiver PRE...

- **Hintergrund: Klassische PRE**

  ⇝ Busy CM (BCM) / Lazy CM (LCM) (Knoop et al., PLDI'92)

  – Ausgezeichnet mit dem *ACM SIGPLAN Most Influential PLDI Paper Award 2002* (*for 1992*)

  – Ausgewählt für *"20 Years of the ACM SIGPLAN PLDI: A Selection"* (60 Artikel aus ca. 600 Artikeln)

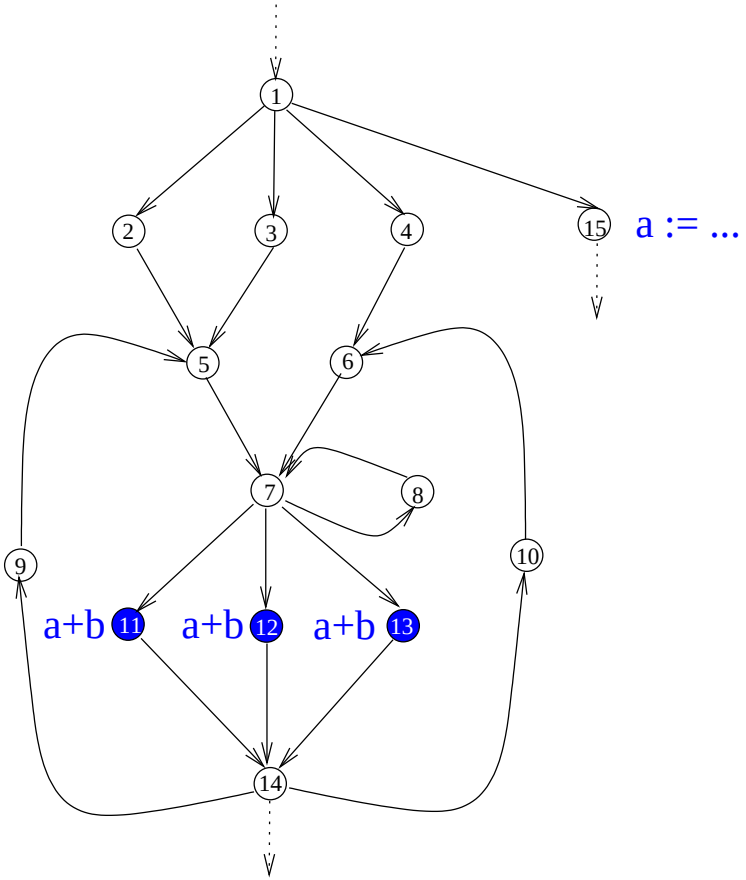
- **Codegrößensensitive PRE** (Knoop et al., POPL'00)

  ⇝ ...modulare Erweiterung von BCM/LCM

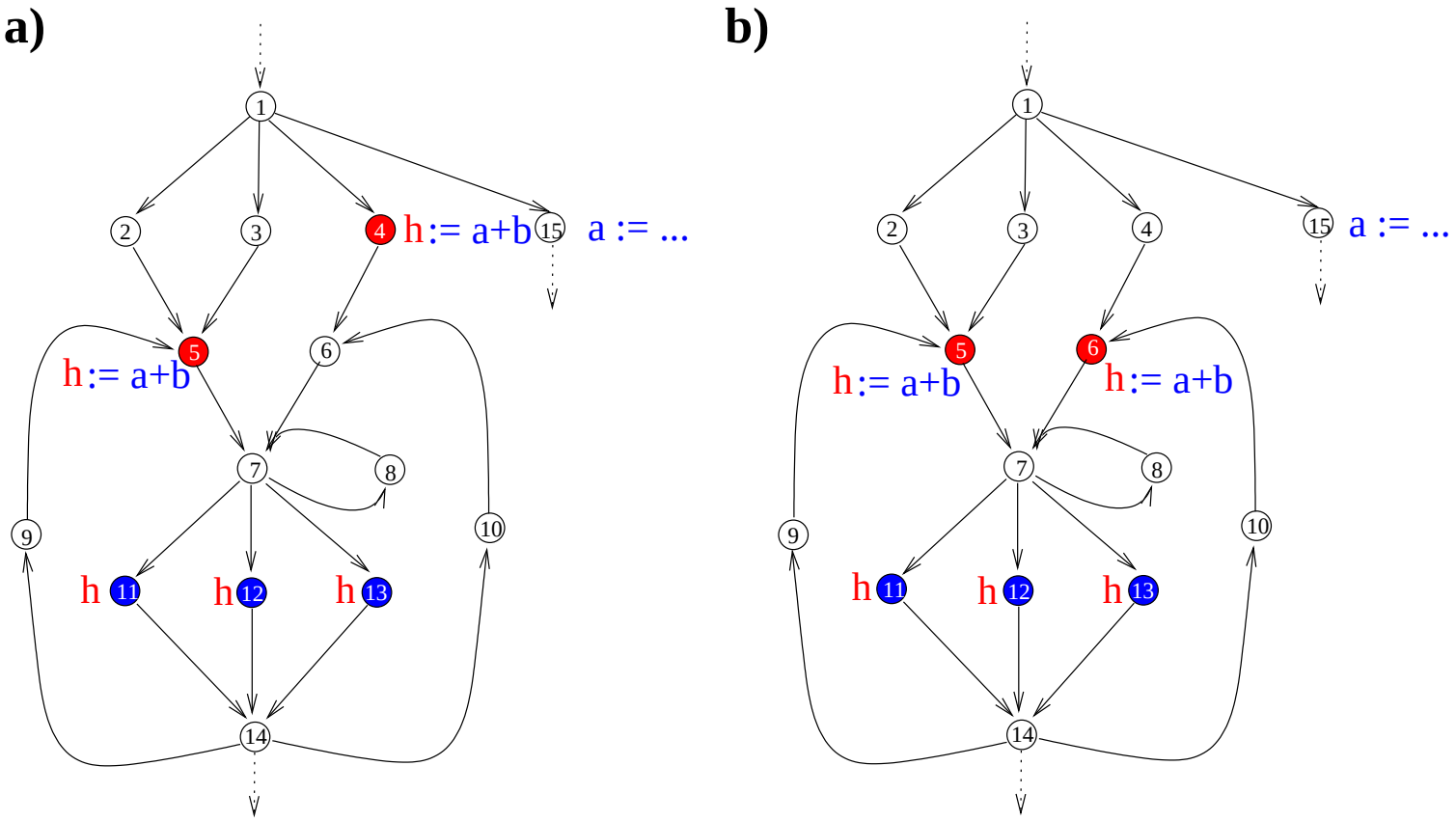
  * Problemmodellierung und -lösung
    ...basiert auf graphtheoretischen Hilfsmitteln

  * Hauptergebnisse
    ...**Korrektheit, Optimalität**

Laufendes Beispiel



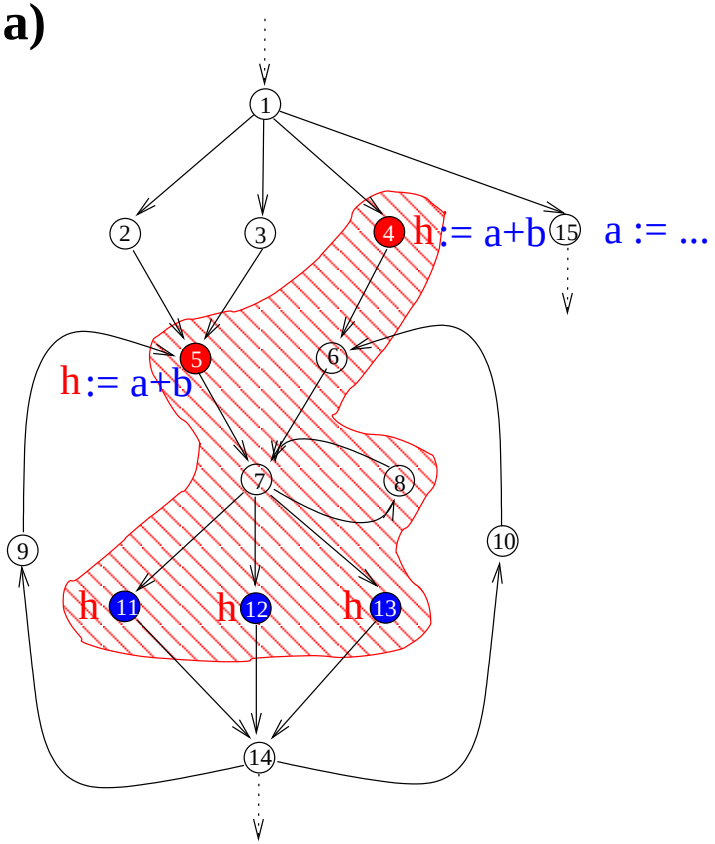
Laufendes Beispiel (Fortsetzung)



Two Code–size Optimal Programs

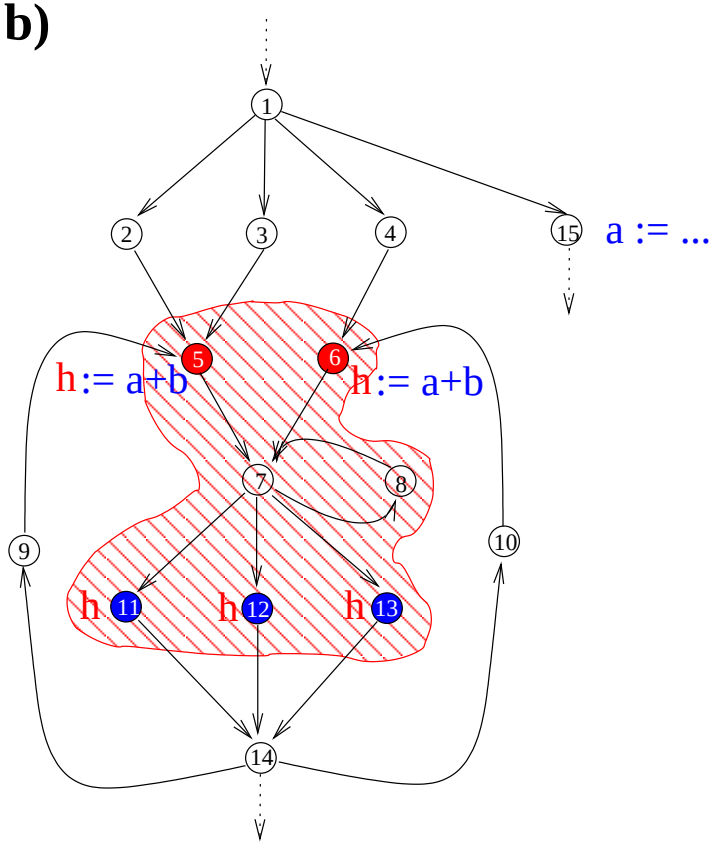
Laufendes Beispiel (Fortsetzung)

a)



SQ > CQ > LQ

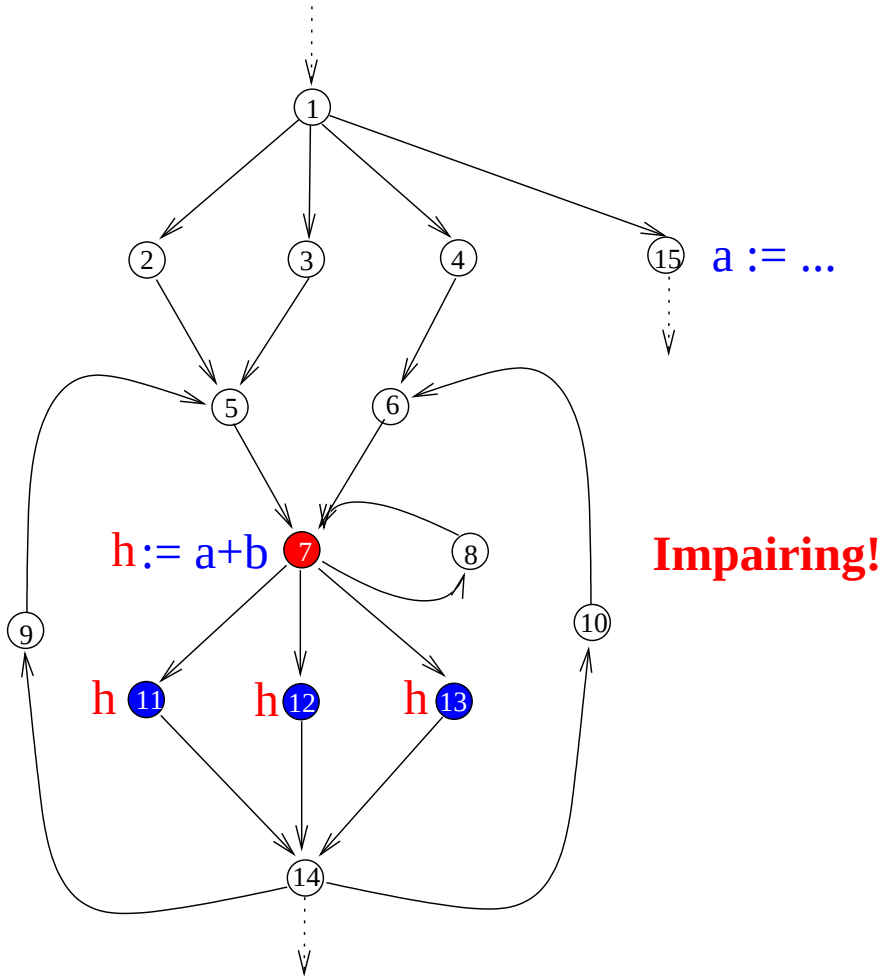
b)



SQ > LQ > CQ

Laufendes Beispiel (Fortsetzung)

Beachte: Folgende Transformation ist unerwünscht!



Codegrößensensitive PRE

$\rightsquigarrow$ Das Problem

...wir erhalten wir eine codegrößenminimale Platzierung der Berechnungen, d.h. eine Platzierung, die

- zulässig (semantik- & performanzerhaltend)
- codegrößenminimal ist?

$\rightsquigarrow$ Lösung: Eine neue Sicht auf PRE

...betrachte PRE als ein Austauschproblem: Austauschen der ursprünglichen Berechnungen gegen neu eingesetzte!

$\rightsquigarrow$ Der Clou: Benutze Graphtheorie!

...führe das Austauschproblem auf die Berechnung sog. tight sets in bipartiten Graphen zurück basierend auf maximalen matchings!

Wir verschieben, aber behalten im Gedächtnis

Wir müssen beantworten...

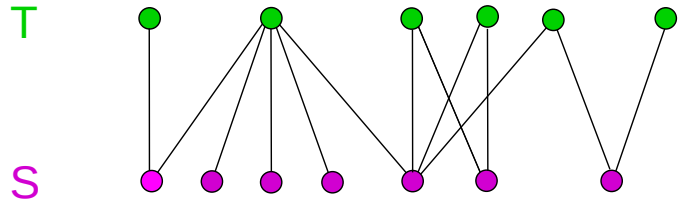
- Wo sind Initialisierungen vorzunehmen und wo sind ursprüngliche Berechnungen zu ersetzen?

...und beweisen

- Warum dies korrekt (semantikerhaltend) ist
- Wie sich dies auf die Codegröße auswirkt
- Warum dies "optimal" bezüglich einer vorgegebenen Priorisierung von Zielen ist?

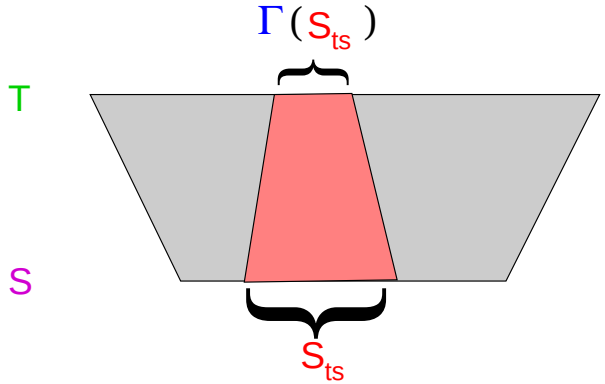
Für jede dieser Fragen werden wir ein spezielles Theorem angeben, das uns die entsprechende Antwort liefert!

Bipartite Graphen



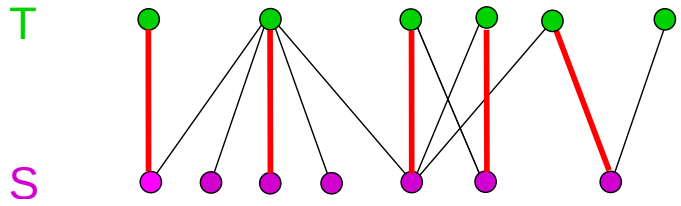
Tight Set
...eines bipartiten Graphen $(S \cup T, E)$: Teilmenge $S_{ts} \subseteq S$ mit

$$\forall\, S' \subseteq S.\ |S_{ts}| - |\Gamma(S_{ts})| \ \geq\ |S'| - |\Gamma(S')|$$



Zwei Varianten: (1) **Größte** Tight Sets (2) **Kleinste** Tight Sets

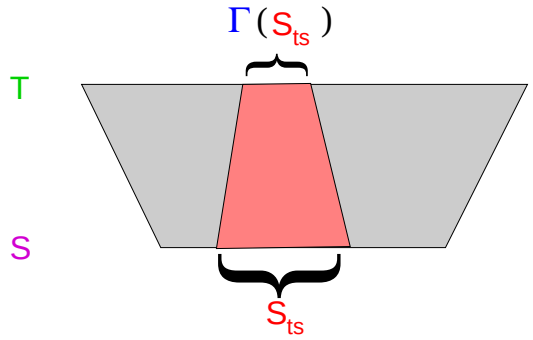
Bipartite Graphen



Tight Set

...eines bipartiten Graphen $(S \cup T, E)$: Teilmenge $S_{ts} \subseteq S$ mit

$$\forall\, S' \subseteq S.\ |S_{ts}| - |\Gamma(S_{ts})| \ \geq\ |S'| - |\Gamma(S')|$$



Zwei Varianten: (1) **Größte** Tight Sets (2) **Kleinste** Tight Sets

Offensichtlich

...können wir auf vorgefertigte Standardalgorithmen aus der Graphtheorie zurückgreifen, um

- **Maximale Matchings** und
- **Tight sets**

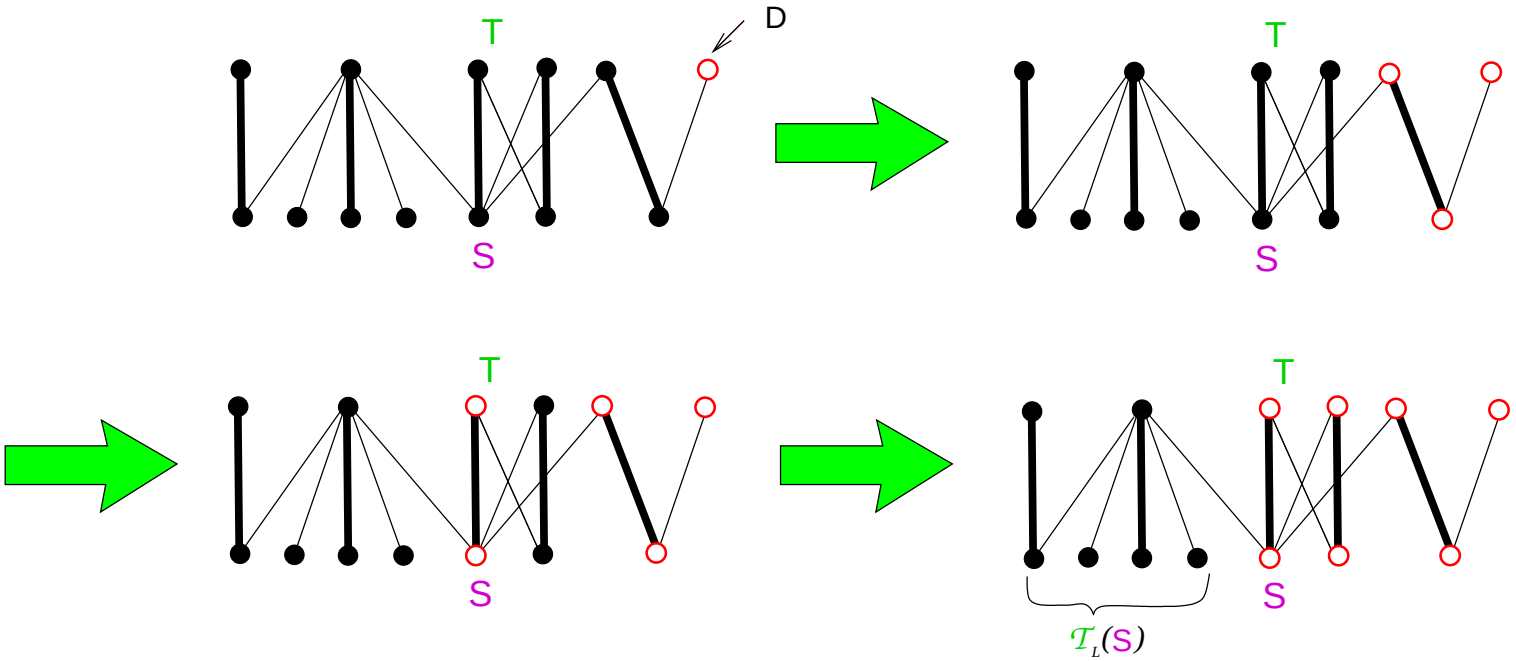
zu berechnen.

Damit reduziert sich unser PRE-Problem auf...

...die Konstruktion des bipartiten Graphen, der das Problem modelliert!

Zur Berechnung größter/kleinster Tight Sets

...auf Grundlage **maximaler Matchings**



Algorithmus LTS (Largest Tight Sets)

Eingabe: Bipartiter Graph $(S \dot\cup T, E)$, maximales Matching M.

Ausgabe: Größte tight set $\mathcal{T}_{LaTS}(S) \subseteq S$.

$S_M := S$; $D := \{t \in T \mid t \text{ is unmatched}\}$;
`WHILE` $D \neq \emptyset$ `DO`
  choose some $x \in D$; $D := D \setminus \{x\}$;
  `IF` $x \in S$
    `THEN` $S_M := S_M \setminus \{x\}$;
         $D := D \cup \{y \mid \{x, y\} \in M\}$
    `ELSE` $D := D \cup (\Gamma(x) \cap S_M)$
  `FI`
`OD`;
$\mathcal{T}_{LaTS}(S) := S_M$

Algorithmus STS (Smallest Tight Sets)

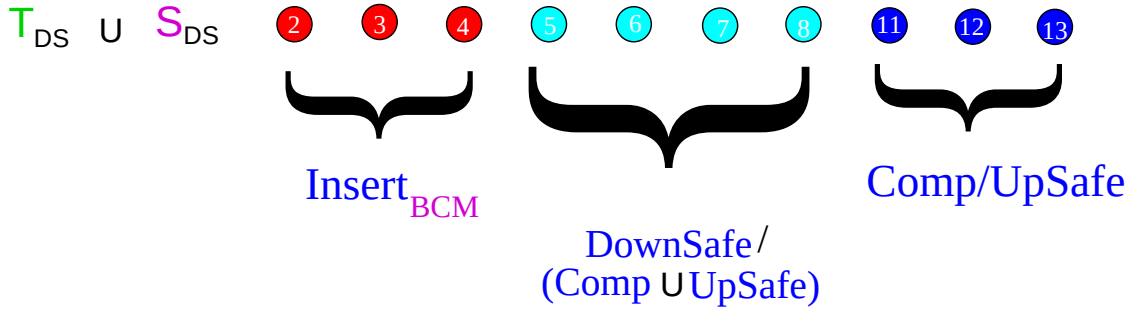
Eingabe: Bipartiter Graph $(S \dot\cup T, E)$, maximales Matching M.

Ausgabe: Kleinste tight set $\mathcal{T}_{SmTS}(S) \subseteq S$.

$S_M := \emptyset$; $A := \{s \in S \mid s \text{ is unmatched}\}$;
WHILE $A \neq \emptyset$ DO
  choose some $x \in A$; $A := A \setminus \{x\}$;
  IF $x \in S$
    THEN $S_M := S_M \cup \{x\}$;
         $A := A \cup (\Gamma(x) \setminus S_M)$
    ELSE $A := A \cup \{y \mid \{x, y\} \in M\}$
  FI
OD;
$\mathcal{T}_{SmTS}(S) := S_M$

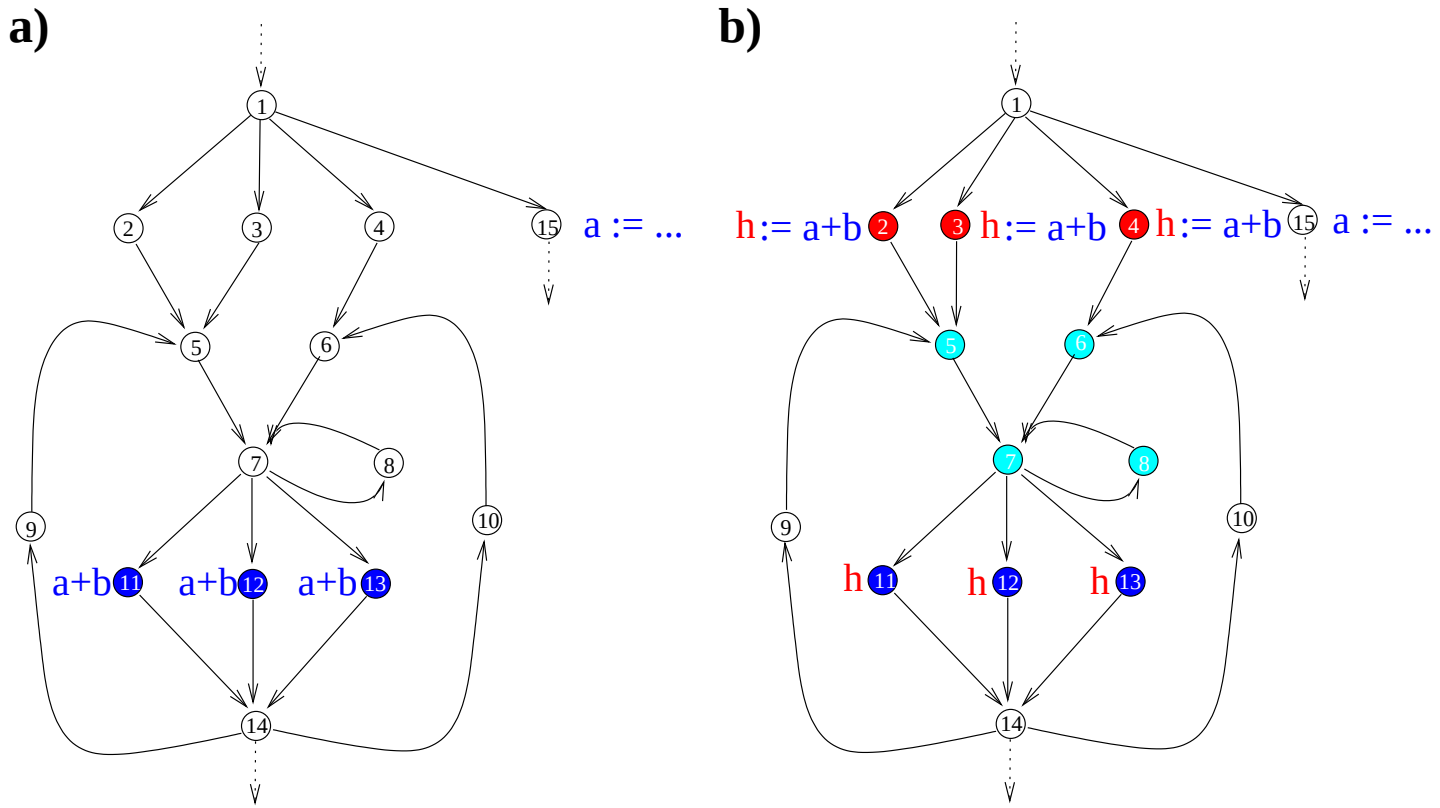
Modellierung des Austauschproblems

Die Menge der Knoten



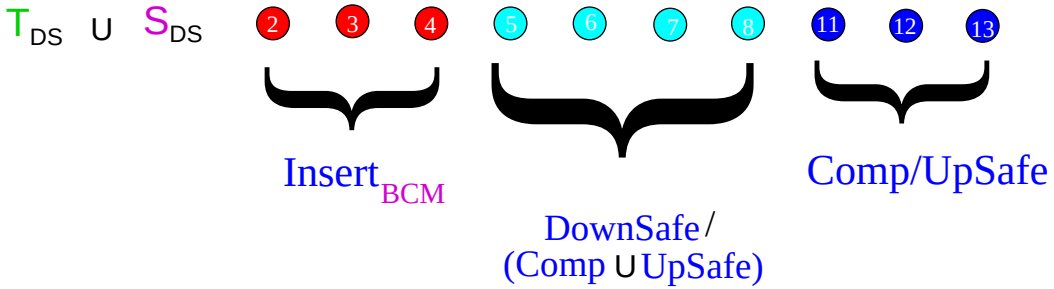
Die Menge der Kanten...

Die Menge der Knoten

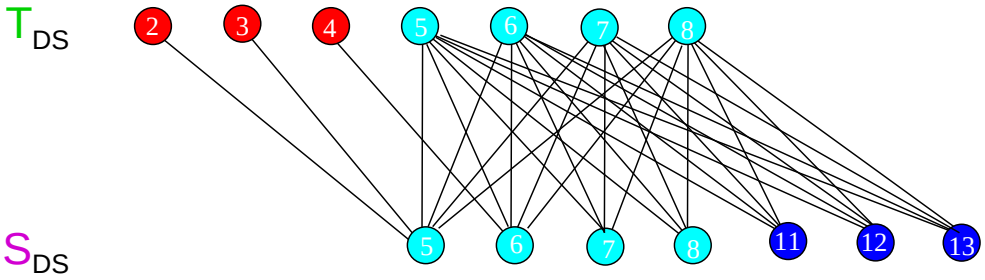


Modellierung des Austauschproblems

Die Menge der Knoten



Der bipartite Graph



Die Menge der Kanten ... $\forall\, n \in S_{DS}\ \forall\, m \in T_{DS}.$

$$\{n, m\} \in E_{DS} \Longleftrightarrow_{df} m \in \textbf{Closure}(pred(n))$$

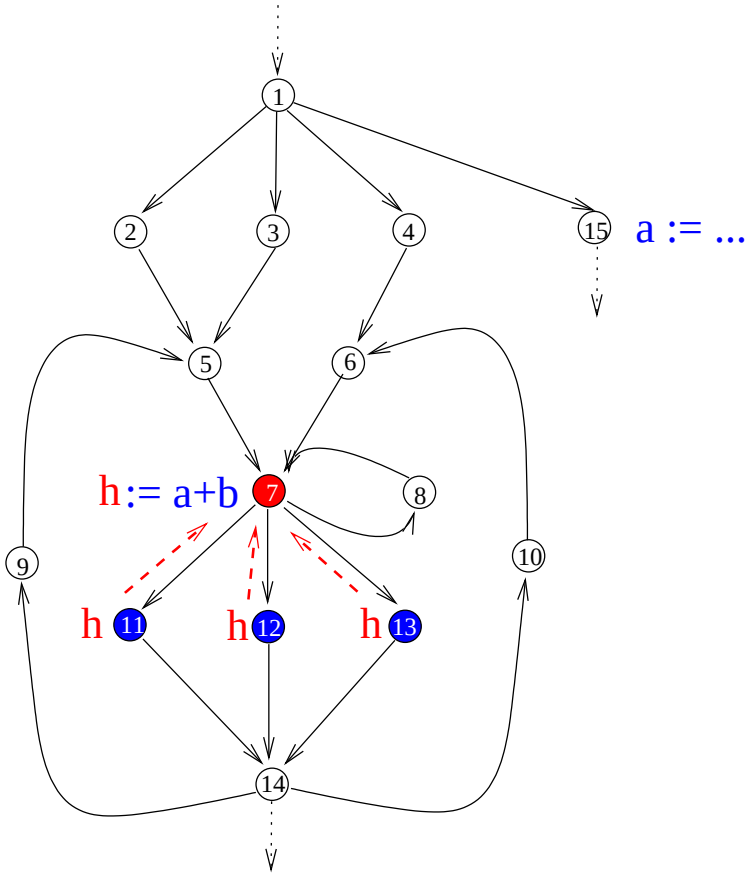
DownSafety-Hüllen

DownSafety-Hüllen

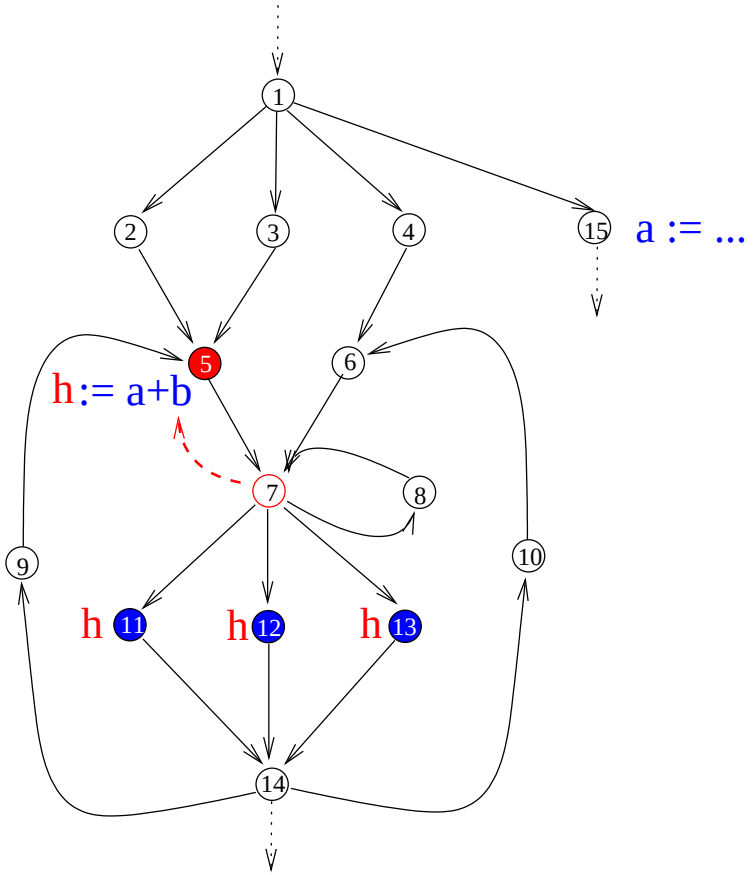
Für $n \in$ *DownSafe/Upsafe* ist die DownSafety-Hülle $Closure(n)$ die kleinste Menge von Knoten, so dass

1. $n \in Closure(n)$
2. $\forall\, m \in Closure(n) \setminus \mathit{Comp}.\ succ(m) \subseteq Closure(n)$
3. $\forall\, m \in Closure(n).\ pred(m) \cap Closure(n) \neq \emptyset \ \Rightarrow$
   $pred(m) \setminus \mathit{UpSafe} \subseteq Closure(n)$

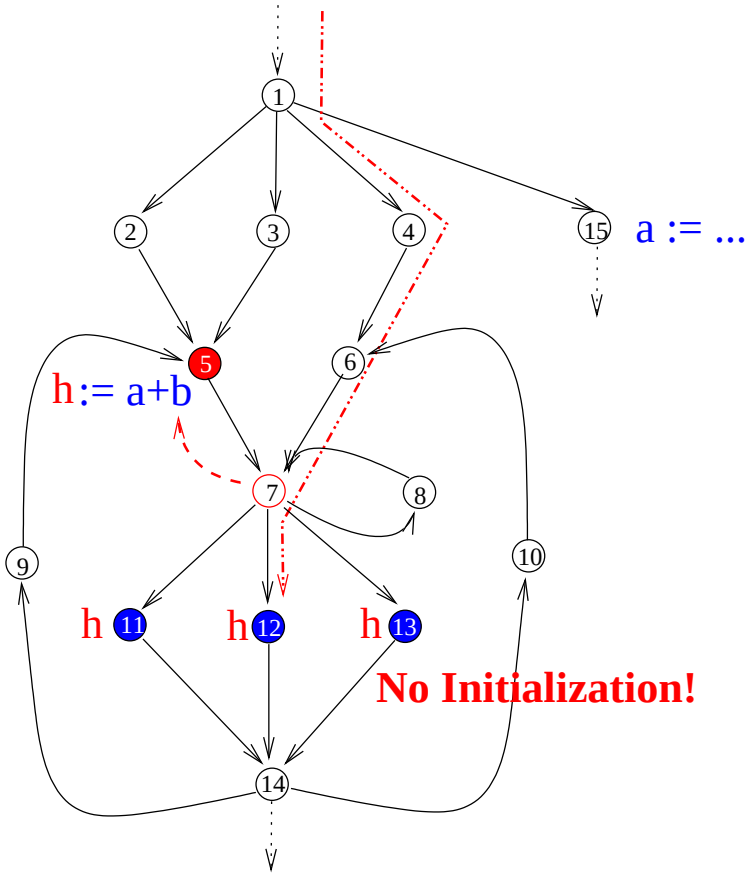
DownSafety-Hüllen – Die zentrale Idee 1(4)



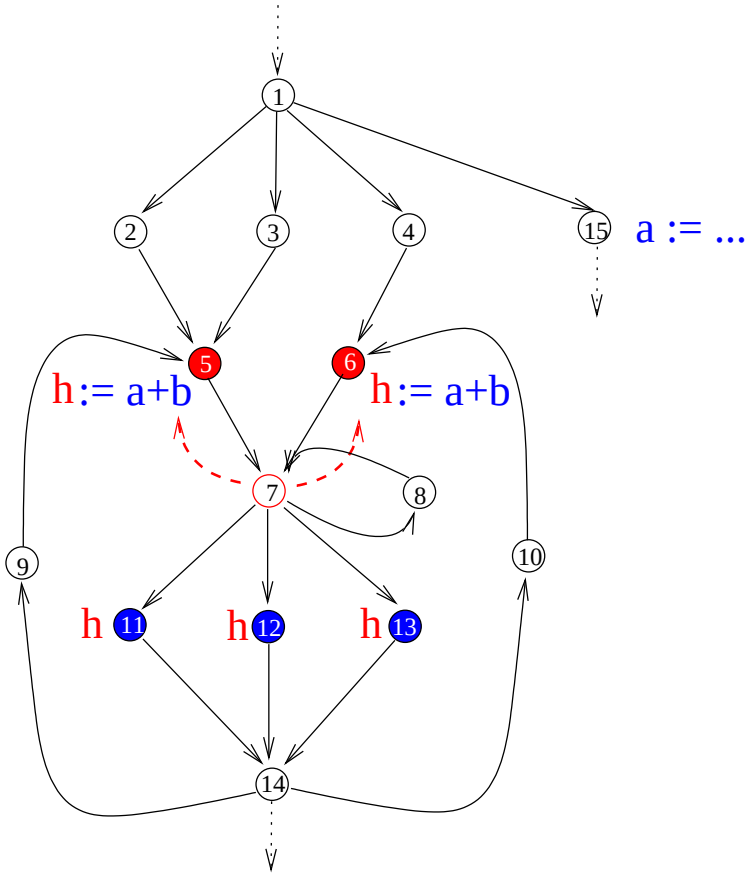
DownSafety-Hüllen – Die zentrale Idee 2(4)



DownSafety-Hüllen – Die zentrale Idee 3(4)



DownSafety-Hüllen – Die zentrale Idee 4(4)



DownSafety-Hüllen

DownSafety-Hüllen

Für $n \in$ *DownSafe/Upsafe* ist die DownSafety-Hülle $Closure(n)$ die kleinste Menge von Knoten, so dass

1. $n \in Closure(n)$
2. $\forall\, m \in Closure(n) \setminus$ *Comp*. $succ(m) \subseteq Closure(n)$
3. $\forall\, m \in Closure(n).\ pred(m) \cap Closure(n) \neq \emptyset \ \Rightarrow$
   $pred(m) \setminus$ *UpSafe* $\subseteq Closure(n)$

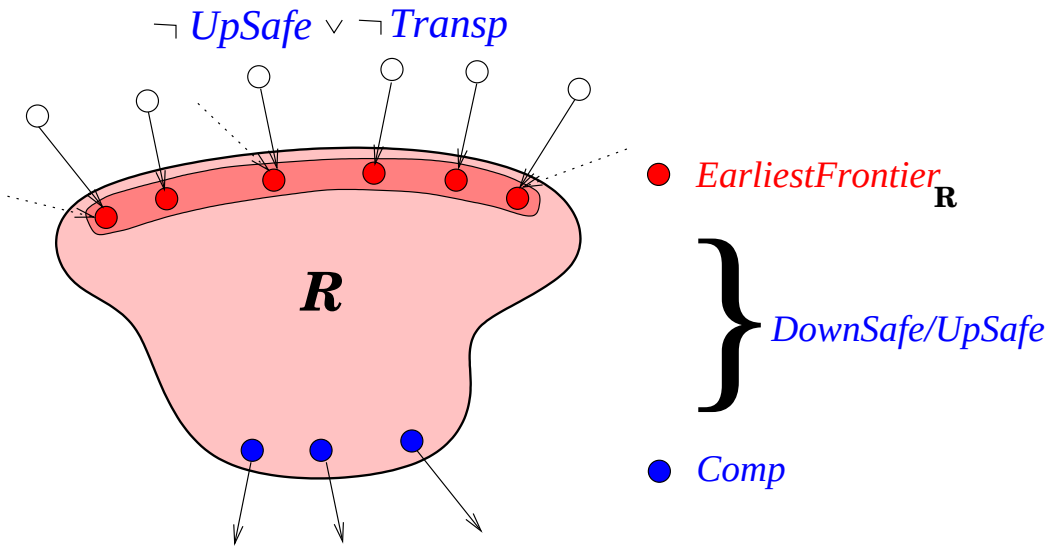
DownSafety-Regionen

Einige Teilmengen von Knoten sind in besonderer Weise ausgezeichnet. Wir nennen diese Mengen DownSafety-Regionen...

- Eine Menge $\mathcal{R} \subseteq N$ von Knoten heißt DownSafety-Region gdw
  1. $\mathit{Comp} \setminus \mathit{UpSafe} \subseteq \mathcal{R} \subseteq \mathit{DownSafe} \setminus \mathit{UpSafe}$
  2. $Closure(\mathcal{R}) = \mathcal{R}$

Fundamental...

Initialisierungstheorem
Initialisierungen zulässiger PRE-Transformationen erfolgen stets am **“frühesten Rand”** von **DownSafety-Regionen**.



...charakterisiert erstmals alle semantikerhaltenden PRE-Transformationen.

Die Schlüsselfragen

...bezüglich Korrektheit und Optimalität:

1. Wo Initialisierungen vornehmen, warum ist es korrekt?
2. Wie ist der Effekt auf die Codegröße?
3. Warum ist das Resultat optimal, d.h., codegrößenminimal?

...drei Theoreme werden jeweils eine dieser Fragen beantworten.

Hauptergebnisse / Erste Frage

1. Wo Initialisierungen vornehmen, warum ist es korrekt?

Intuitiv: am frühesten Rand der von der tight set induzierten DS-region...

Theorem 1 [Tight Sets: Initialisierungspunkte]
Sei $TS \subseteq S_{DS}$ eine tight set.
Dann ist $\mathcal{R}_{TS} =_{df} \Gamma(TS) \cup (Comp \backslash UpSafe)$
eine DownSafety-Region mit $Body_{\mathcal{R}_{TS}} = TS$

Korrektheit
...unmittelbares Korollar aus Theorem 1 und dem Initialisierungstheorem

Hauptergebnisse / Zweite Frage

2. Wie ist der Effekt auf die Codegröße?

Intuitiv: Die Differenz aus eingesetzten und ersetzten Berechnungen...

Theorem 2 [DownSafety-Regionen: Platzgewinn]
Sei $\mathcal{R}$ eine DownSafety-Region
mit $Body_{\mathcal{R}} =_{df} \mathcal{R} \backslash EarliestFrontier_{\mathcal{R}}$
Dann

- **Platzgewinn aufgrund Einsetzens an $EarliestFrontier_{\mathcal{R}}$:**
  $|Comp \backslash UpSafe| - |EarliestFrontier_{\mathcal{R}}| =$
  $|Body_{\mathcal{R}}| - |\Gamma(Body_{\mathcal{R}})| \quad_{df} = defic(Body_{\mathcal{R}})$

Hauptergebnisse / Dritte Frage

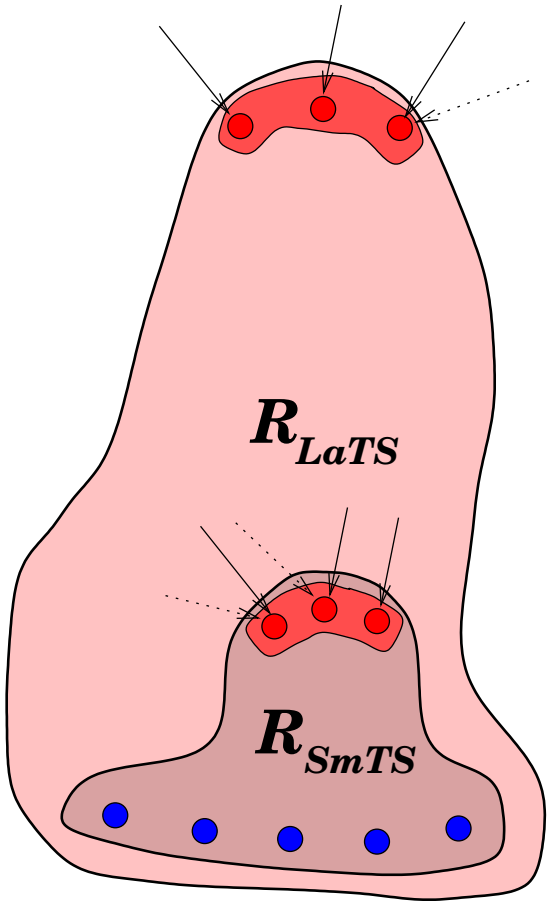
3. Warum ist das Resultat optimal, d.h., codegrößenminimal?

Aufgrund einer inhärenten Eigenschaft von tight sets (non-negative deficiency!)...

Optimalitätstheorem [Die Transformation]
Sei $TS \subseteq S_{DS}$ eine tight set.

- **Initialisierungspunkte:**
  $\mathit{Insert}_{SpCM} =_{df} \mathit{EarliestFrontier}_{\mathcal{R}_{TS}} = \mathcal{R}_{TS} \backslash TS$
- **Platzgewinn:**
  $\mathit{defic}(TS) =_{df} |TS| - |\Gamma(TS)| \geq 0$ max.

Größte vs. kleinste Tight Sets: Der Einfluss



- *EarliestFrontier*$_{R_{LaTS}}$

Largest tight sets favor

Computational Quality

→ **Earliestness Principle**

- *EarliestFrontier*$_{R_{SmTS}}$

Smallest tight sets favor

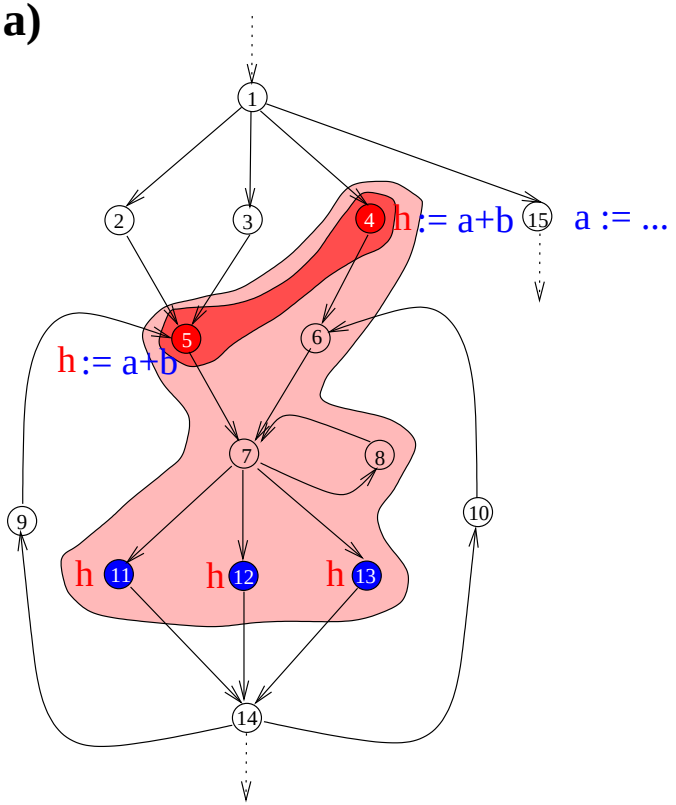
Lifetime Quality

→ **Latestness Principle**

- *Comp*

Effekt illustriert am laufenden Beispiel

a)

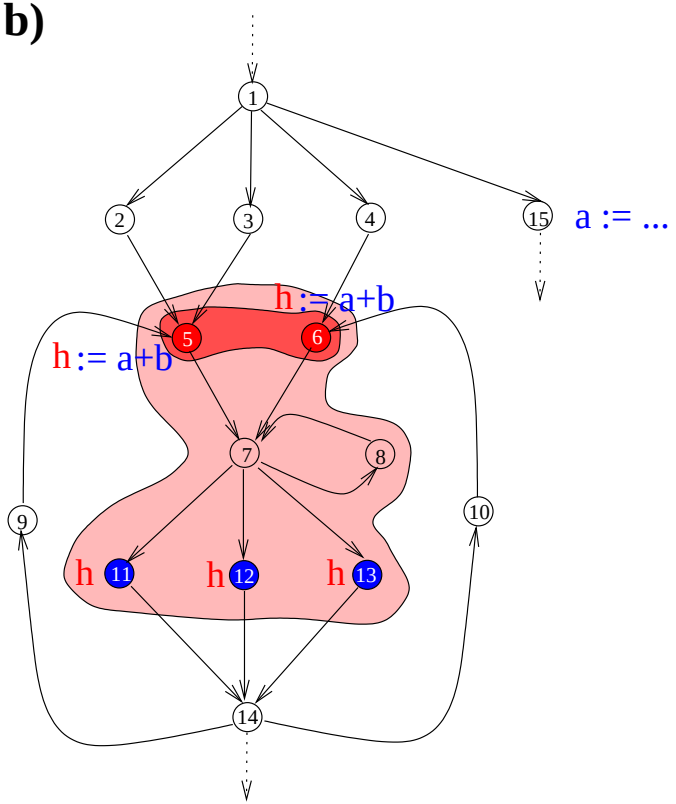


Largest Tight Set

(SQ > CQ)

Earliestness Principle

b)

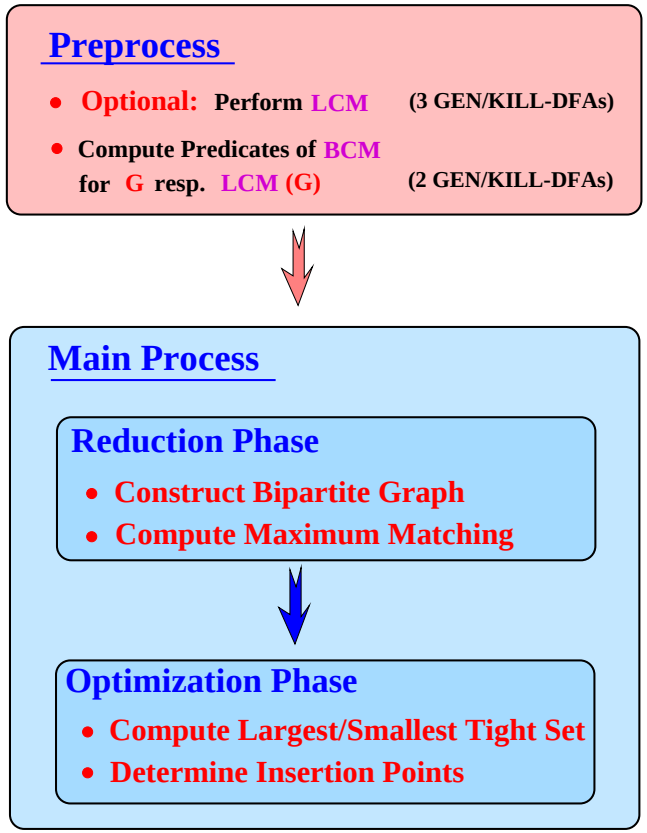


Smallest Tight Set

(SQ > LQ)

Latestness Principle

Codegrößensensitive PRE auf einen Blick 1(2)

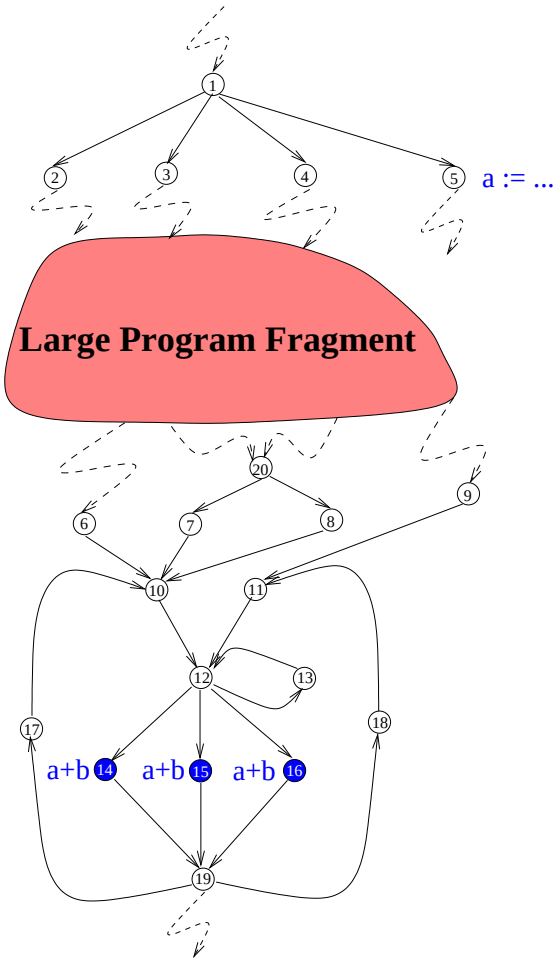


...auf einen Blick 2(2)

Choice of Priority	Apply	To	Using	Yields	Auxiliary Information Required
$\mathcal{LQ}$	**Not meaningful: The identity, i.e., G itself is optimal!**				
$\mathcal{SQ}$	**Subsumed by $\mathcal{SQ} > \mathcal{CQ}$ and $\mathcal{SQ} > \mathcal{LQ}$!**				
$\mathcal{CQ}$	**BCM**	G			UpSafe(G), DownSafe(G)
$\mathcal{CQ} > \mathcal{LQ}$	**LCM**	G		**LCM**(G)	UpSafe(G), DownSafe(G), Delay(G)
$\mathcal{SQ} > \mathcal{CQ}$	**SpCM**	G	**Largest tight set**	**SpCM**$_{LTS}$(G)	UpSafe(G), DownSafe(G)
$\mathcal{SQ} > \mathcal{LQ}$	**SpCM**	G	**Smallest tight set**		UpSafe(G), DownSafe(G)
$\mathcal{CQ} > \mathcal{SQ}$	**SpCM**	**LCM**(G)	**Largest tight set**		UpSafe(G), DownSafe(G), Delay(G) UpSafe(**LCM**(G)), DownSafe(**LCM**(G))
$\mathcal{CQ} > \mathcal{SQ} > \mathcal{LQ}$	**SpCM**	**LCM**(G)	**Smallest tight set**		UpSafe(G), DownSafe(G), Delay(G) UpSafe(**LCM**(G)), DownSafe(**LCM**(G))
$\mathcal{SQ} > \mathcal{CQ} > \mathcal{LQ}$	**SpCM**	**DL**(**SpCM**$_{LTS}$(G))	**Smallest tight set**		UpSafe(G), DownSafe(G), Delay(**SpCM**$_{LTS}$(G)), UpSafe(**DL**(**SpCM**$_{LTS}$(G))), DownSafe(**DL**(**SpCM**$_{LTS}$(G)))

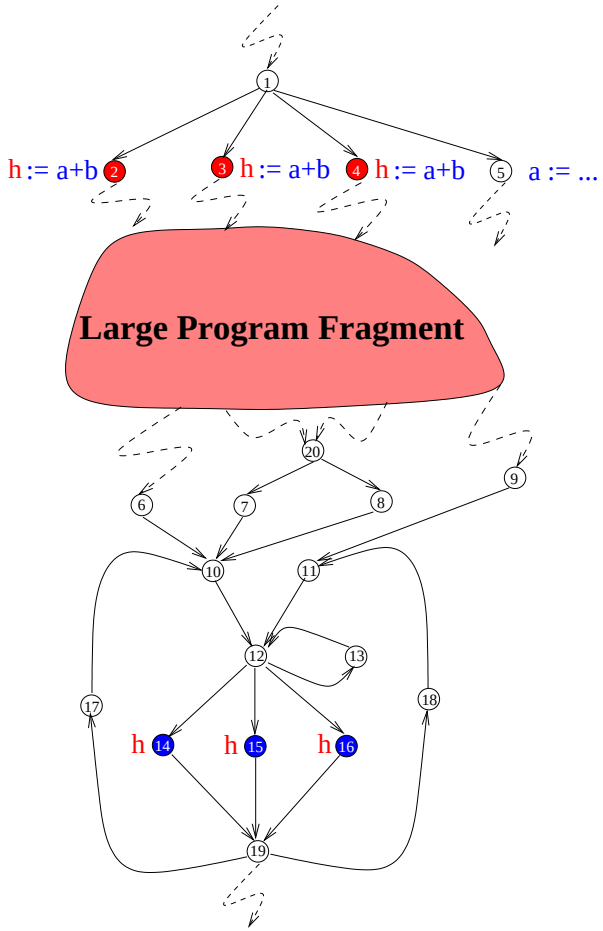
Flexibilität (1)

Das Ausgangsprogramm ...



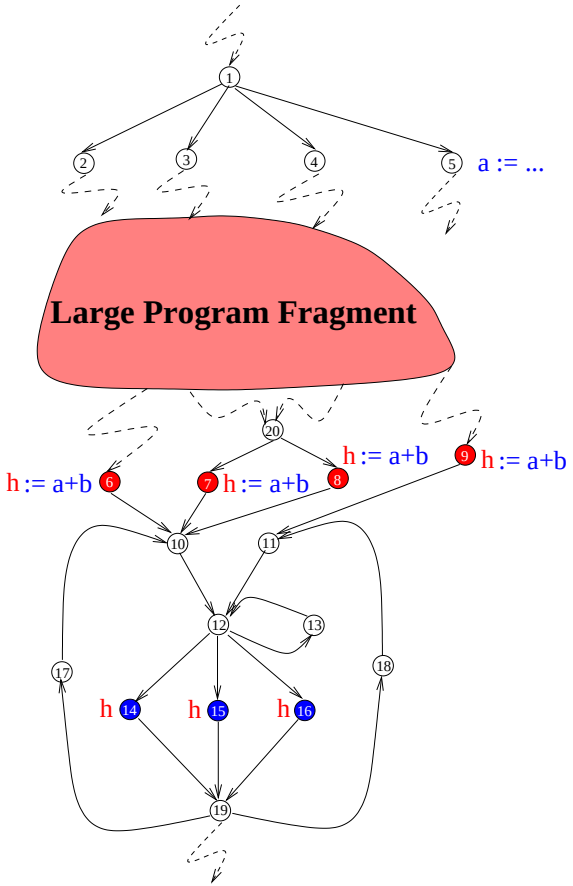
Flexibilität (2)

BCM ... Ein berechnungsoptimales Programm ($\mathcal{CQ}$)



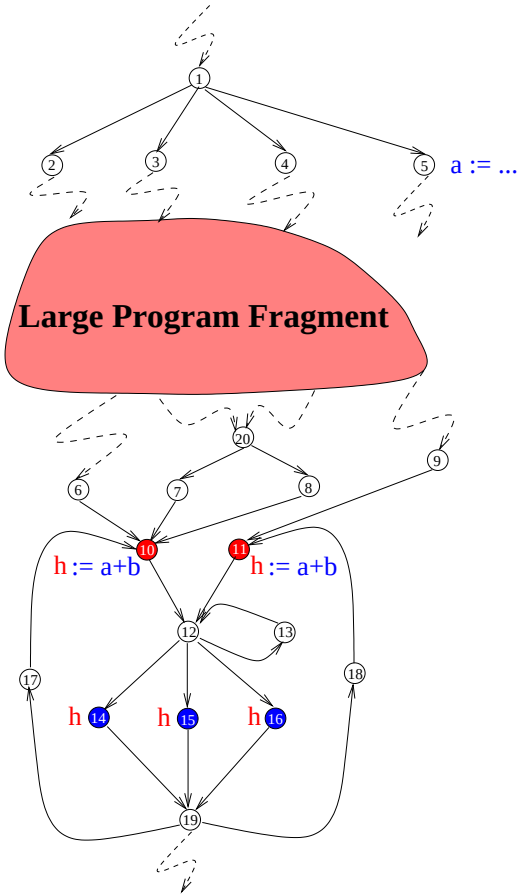
Flexibilität (3)

LCM ... A Computationally & Lifetime Opt. Program ($\mathcal{CQ} > \mathcal{LQ}$)



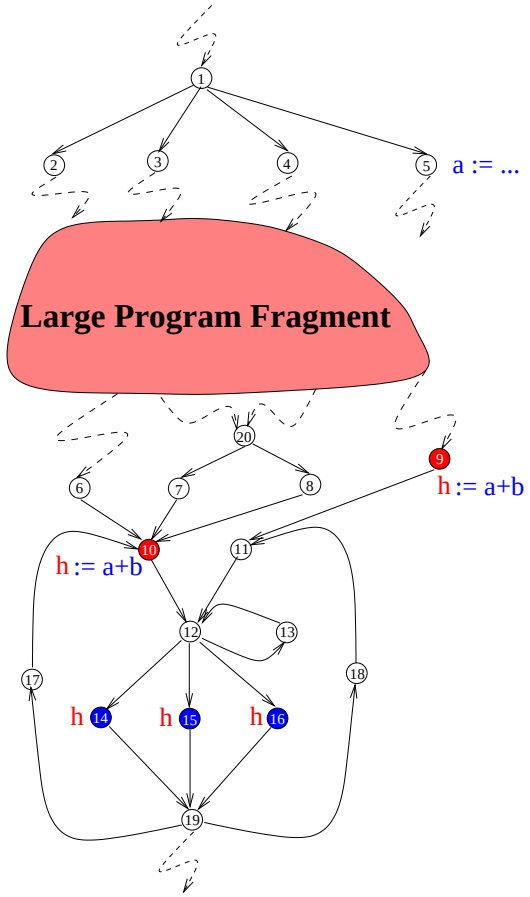
Flexibilität (4)

SpCM ... A Code-Size & Lifetime Optimal Program ($\mathcal{SQ} > \mathcal{LQ}$)



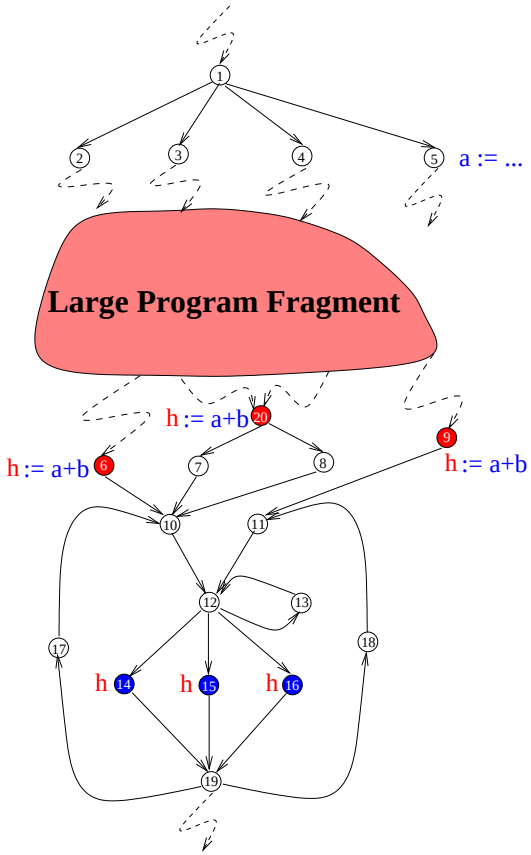
Flexibilität (5)

SpCM ... A Computationally and Lifetime Best Code-Size Optimal Program ($\mathcal{SQ} > \mathcal{CQ} > \mathcal{LQ}$)



Flexibilität (6)

SpCM ... A Code-Size and Lifetime Best Computationally Optimal Program ($\mathcal{CQ} > \mathcal{SQ} > \mathcal{LQ}$)



Ein Rückblick

...auf die Entwicklung von PRE:

- 1958: *...first glimpse of PRE*

  $\rightsquigarrow$ Ershov's work on *On Programming of Arithmetic Operations*.

- $<$ **1979** ... Special Techniques

  $\rightsquigarrow$ Total Redundancy Elimination, Loop Invariant Code Motion

- **1979:** *...origin of contemporary PRE*

  $\rightsquigarrow$ Morel/Renvoise's seminal work on PRE

- **1992:** *...LCM* [Knoop et al., PLDI'92]

  $\rightsquigarrow$ ...first to achieve comp. optimality with minimum register pressure

  $\rightsquigarrow$ ...first to rigorously be proven correct and optimal

Ein Rückblick (fortgesetzt)

- **2000:** *...origin of code-size sensitive PRE* [Knoop et al., POPL 2000]

  $\rightsquigarrow$ ...first to allow prioritization of goals

  $\rightsquigarrow$ ...rigorously be proven correct and optimal

  $\rightsquigarrow$ ...first to bridge the gap between traditional compilation and compilation for embedded systems

- ca. since 1997: *...a new strand of research on PRE*

  $\rightsquigarrow$ Speculative PRE: Gupta, Horspool, Soffa, Xue, Scholz, Knoop,...

- **2005**: *...another fresh look at PRE* (*as maximum flow problem*)

  $\rightsquigarrow$ Unifying PRE and Speculative PRE [Jingling Xue and J. Knoop]

Namen sind Nachrichten

Ein anderer Rückblick...

- $<$ **1979** ... Special Techniques

  $\rightsquigarrow$ Total Redundancy Elimination, Loop Invariant Code Motion

- **1979** ... Partial Redundancy Elimination

  $\rightsquigarrow$ **Pioneering** ... Morel/Renvoise's bidirectional algorithm [1979]

  $\rightsquigarrow$ **Heuristic improvements** ... Dhamdhere [1988, 1991], Drechsler/Stadel [1988], Sorkin [1989], Dhamdhere/Rosen/Zadeck [1992], ...

- **1992** ... BCM & LCM [Knoop et al., PLDI'92]

  $\rightsquigarrow$ BCM ... first to achieve Computational Optimality: Earliestness Principle

  $\rightsquigarrow$ LCM ... first to achieve Comp. & Lifetime Optimality: Latestness Principle

  ... first to be purely unidirectional, however, not yet code-size sensitive.

- **2000/2004**: Code-Size Sensitive PRE [Knoop et al., POPL 2000, LCTES 2004]

Warum lohnt es sich, PRE zu betrachten? (1)

Es ist...

- Relevant ...weit verbreitet in der Praxis
- Generell ...eine Familie von Optimierungen denn eine einzelne Optimierung
- Wohlverstanden ...bewiesen korrekt und *optimal*
- Herausfordernd ...konzeptuell einfach, aber weist eine Reihe kopfnussaufgebender Phänomene auf

Warum lohnt es sich, PRE zu betrachten? (2)

Zu guter letzt, PRE ist...

- Wahrlich klassisch ...blickt auf eine lange Geschichte zurück
  - Morel, E. and Renvoise, C. *Global Optimization by Suppression of Partial Redundancies*. CACM 22 (2), 96 - 103, 1979.
  - Ershov, A. P. *On Programming of Arithmetic Operations*. CACM 1 (8), 3 - 6, 1958.

Exkurs: Die PRE-Formulierung von Morel/Renvoise

PRE ist untrennbar mit den Namen von E. Morel und C. Renvoise verknüpft. Ihr 1979 vorgestelltes Verfahren kann als "Urvater" aller CM-Verfahren angesehen werden und war bis in die 90er-Jahre hinein das "state of the art"-Verfahren für PRE.

Kennzeichnend für dieses Verfahren sind:

- 3 unidirektionale Bitvektoranalysen (AV, ANT, PAV)
- 1 bidirektionales Bitvektoranalyse (PP)

Die bahnbrechende PRE-Formulierung von Morel/Renvoise (1)

- Verfügbarkeit (Availability):

$$\mathbf{AVIN}(n) = \begin{cases} false & \text{falls } n = \mathbf{s} \\ \prod_{m \in pred(n)} \mathbf{AVOUT}(m) & \text{sonst} \end{cases}$$

$$\mathbf{AVOUT}(n) = \mathbf{TRANSP}(n) * (\mathbf{COMP}(n) + \mathbf{AVIN}(n))$$

Die bahnbrechende PRE-Formulierung von Morel/Renvoise (2)

- Vorziehbarkeit (Anticipability):

$$\mathbf{ANTIN}(n) \;=\; \mathbf{COMP}(n) + \mathbf{TRANSP}(n) * \mathbf{ANTOUT}(n)$$

$$\mathbf{ANTOUT}(n) \;=\; \begin{cases} false & \text{falls } n = \mathbf{e} \\ \prod\limits_{m \in succ(n)} \mathbf{ANTIN}(m) & \text{sonst} \end{cases}$$

Die bahnbrechende PRE-Formulierung von Morel/Renvoise (3)

- Partielle Verfügbarkeit (Partial Availability):

$$\mathbf{PAVIN}(n) = \begin{cases} false & \text{falls } n = \mathrm{s} \\ \sum\limits_{m \in pred(n)} \mathbf{PAVOUT}(m) & \text{sonst} \end{cases}$$

$$\mathbf{PAVOUT}(n) = \mathbf{TRANSP}(n) * (\mathbf{COMP}(n) + \mathbf{PAVIN}(n))$$

Die bahnbrechende PRE-Formulierung von Morel/Renvoise (4)

- Platzierung möglich (Placement Possible):

$$\mathbf{PPIN}(n) = \begin{cases} \mathit{false} \quad \text{falls } n = \mathbf{s} \\ \mathbf{CONST}(n) * \\ (\prod_{m \in pred(n)} (\mathbf{PPOUT}(m) + \mathbf{AVOUT}(m)) * \\ (\mathbf{COMP}(n) + \mathbf{TRANSP}(n) * \mathbf{PPOUT}(n)) \quad \text{sonst} \end{cases}$$

$$\mathbf{PPOUT}(n) = \begin{cases} \mathit{false} & \text{falls } n = \mathbf{e} \\ \prod_{m \in succ(n)} \mathbf{PPIN}(m) & \text{sonst} \end{cases}$$

wobei $\mathbf{CONST}(n) =_{df} \mathbf{ANTIN}(n) * (\mathbf{PAVIN}(n) + \neg\mathbf{COMP}(n) * \mathbf{TRANSP}(n))$

Die bahnbrechende PRE-Formulierung von Morel/Renvoise (5)

- Initialisierung:

$$\mathbf{INSIN}(n) \;=_{df}\; false$$

$$\mathbf{INSOUT}(n) \;=_{df}\; \mathbf{PPOUT}(n) * \neg\mathbf{AVOUT}(n) * (\neg\mathbf{PPIN}(n) + \neg\mathbf{TRANSP}(n))$$

- Ersetzung:

$$\mathbf{REPLACE}(n) \;=_{df}\; \mathbf{COMP}(n) * \mathbf{PPIN}(n)$$

Schwächen der PRE-Formulierung

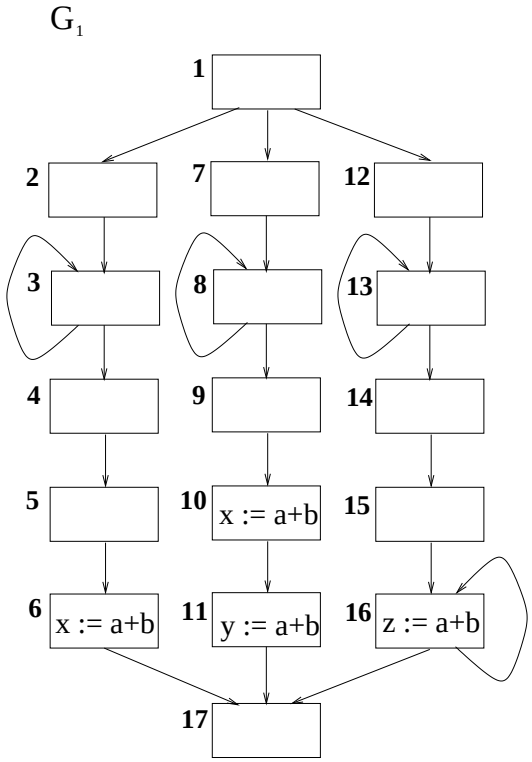
...von Morel/Renvoise:

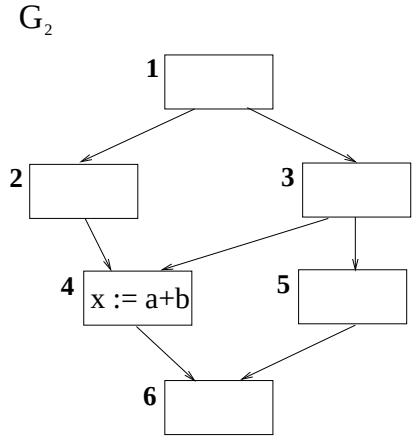
- Konzeptuell
  - Fehlende Berechnungsoptimalität
    $\leadsto$ nur aufgrund nicht gespaltener kritischer Kanten
  - Fehlende Lebenszeitoptimalität
    $\leadsto$ Heuristische Behandlung
- Technisch
  - Bidirektionalität
    $\leadsto$ konzeptuell und berechnungsmäßig komplexer

...das Transformationsergebnis liegt (nicht vorhersagbar) zwischen denen von BCM- und LCM-Transformation.

Lehrreich

...folgende zwei Beispiele mithilfe des PRE-Verfahrens von Morel/Renvoise zu optimieren:





Vorschau auf die weiteren Vorlesungstermine...

- *Mo, 17.12./24.12./31.12.2007: Keine Vorlesung(en)!* (*Ferialzeit*)
- Mo, 14.01.2008: Vorlesung von 16:15 Uhr bis 17:45 Uhr im Hörsaal 14, TU-Hauptgebäude
- Mo, 21.01.2008: Vorlesung von 16:15 Uhr bis 17:45 Uhr im Hörsaal 14, TU-Hauptgebäude

Einladung zum Kolloquiumsvortrag

Die Complang-Gruppe lädt ein zu folgendem Vortrag...

Automatic Verification of Combined Specifications

Prof. Dr. Ernst-Rüdiger Olderog

Carl v. Ossietzky Universität Oldenburg, Deutschland

ZEIT: Freitag, 13. Dezember 2007, 14:00 Uhr c.t.

ORT: TU Wien, Elektrotechnik, EI 5 Hochenegg-Hörsaal, Gußhausstr. 25-29 (Altbau), 2. Stock

MEHR INFO: `http://www.complang.tuwien.ac.at/talks/Olderog2007-12-13`

Alle Interessenten sind herzlich willkommen!