Script generated by TTT

Title: Seidl: Programmoptimierung (05.11.2015)

Date: Thu Nov 05 08:35:01 CET 2015

Duration: 89:40 min

Pages: 61

... end of background on: Complete Lattices

Final Question

Why is a (or the least) solution of the constraint system useful ???

For a complete lattice $\mathbb{D}$, consider systems:

$$\begin{array}{lll} \mathcal{I}[start] & \sqsupseteq & d_0 \\ \mathcal{I}[v] & \sqsupseteq & [\![k]\!]^\sharp\,(\mathcal{I}[u]) \qquad k = (u,\_,v) \quad \text{edge} \end{array}$$

where $d_0 \in \mathbb{D}$ and all $[\![k]\!]^\sharp : \mathbb{D} \to \mathbb{D}$ are monotonic ...

... end of background on: Complete Lattices

Final Question

Why is a (or the least) solution of the constraint system useful ???

For a complete lattice $\mathbb{D}$, consider systems:

$$\begin{array}{lll} \mathcal{I}[start] & \sqsupseteq & d_0 \\ \mathcal{I}[v] & \sqsupseteq & [\![k]\!]^\sharp\,(\mathcal{I}[u]) \qquad k = (u,\_,v) \quad \text{edge} \end{array}$$

where $d_0 \in \mathbb{D}$ and all $[\![k]\!]^\sharp : \mathbb{D} \to \mathbb{D}$ are monotonic ...

$\Longrightarrow$ Monotonic Analysis Framework

Wanted: MOP (Merge Over all Paths)

$$\mathcal{I}^*[v] = \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \to^* v\}$$

... end of background on: Complete Lattices

Final Question

Why is a (or the least) solution of the constraint system useful ???

For a complete lattice $\mathbb{D}$, consider systems:

$$\begin{array}{lll} \mathcal{I}[start] & \sqsupseteq & d_0 \\ \mathcal{I}[v] & \sqsupseteq & [\![k]\!]^\sharp\,(\mathcal{I}[u]) \qquad k = (u, \_, v) \quad \text{edge} \end{array}$$

where $d_0 \in \mathbb{D}$ and all $[\![k]\!]^\sharp : \mathbb{D} \to \mathbb{D}$ are monotonic ...

$\Longrightarrow$ Monotonic Analysis Framework

Wanted: MOP (Merge Over all Paths)

$$\mathcal{I}^*[v] = \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \to^* v\}$$

Wanted: MOP (Merge Over all Paths)

$$\mathcal{I}^*[v] = \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \to^* v\}$$

Theorem

Kam, Ullman 1975

Assume $\mathcal{I}$ is a solution of the constraint system. Then:

$$\mathcal{I}[v] \sqsupseteq \mathcal{I}^*[v] \qquad \text{for every } v$$

Proof: Induction on the length of π.

Proof: Induction on the length of π.

Proof: Induction on the length of π.

Foundation: $\pi = \epsilon$ (empty path)

Then:

$$[\![\pi]\!]^\sharp \, d_0 = [\![\epsilon]\!]^\sharp \, d_0 = d_0 \sqsubseteq \mathcal{I}[start]$$

Proof: Induction on the length of π.

Foundation: $\pi = \epsilon$ (empty path)

Then:

$$[\![\pi]\!]^\sharp \, d_0 = [\![\epsilon]\!]^\sharp \, d_0 = d_0 \sqsubseteq \mathcal{I}[start]$$

Step: $\pi = \pi' k$ for $k = (u, \_, v)$ edge.

Proof: Induction on the length of π.

Foundation: $\pi = \epsilon$ (empty path)

Then:

$$[\![\pi]\!]^\sharp \, d_0 = [\![\epsilon]\!]^\sharp \, d_0 = d_0 \sqsubseteq \mathcal{I}[start]$$

Step: $\pi = \pi' k$ for $k = (u, \_, v)$ edge.

Then:

$$\begin{array}{llll} & [\![\pi']\!]^\sharp \, d_0 & \sqsubseteq & \mathcal{I}[u] & \text{by I.H. for } \pi' \\ \Longrightarrow & [\![\pi]\!]^\sharp \, d_0 & = & [\![k]\!]^\sharp \, ([\![\pi']\!]^\sharp \, d_0) & \\ & & \sqsubseteq & \boxed{[\![k]\!]^\sharp \, (\mathcal{I}[u])} & \text{since } [\![k]\!]^\sharp \text{ monotonic} \\ & & \sqsubseteq & \boxed{\mathcal{I}[v]} & \text{since } \mathcal{I} \text{ solution} \end{array}$$

Disappointment

Are solutions of the constraint system just upper bounds ???

Disappointment

Are solutions of the constraint system just upper bounds ???

Answer

In general: yes

Disappointment

Are solutions of the constraint system just upper bounds ???

Answer

In general: yes

With the notable exception when all functions $[k]^\sharp$ are distributive ...

Gary A. Kildall (1942-1994).

Has developed the operating system CP/M and GUIs for PCs.

Gary A. Kildall (1942-1994).

Has developed the operating system CP/M and GUIs for PCs.

The function $f : \mathbb{D}_1 \to \mathbb{D}_2$ is called

- distributive, if $f\,(\bigsqcup X) = \bigsqcup\{f\,x \mid x \in X\}$ for all $\emptyset \neq X \subseteq \mathbb{D}$;
- strict, if $f\,\bot = \bot$.
- totally distributive, if f is distributive and strict.

The function $f : \mathbb{D}_1 \to \mathbb{D}_2$ is called

- distributive, if $f\,(\bigsqcup X) = \bigsqcup\{f\,x \mid x \in X\}$ for all $\emptyset \neq X \subseteq \mathbb{D}$;
- strict, if $f\,\bot = \bot$.
- totally distributive, if f is distributive and strict.

Examples

- $f\,x = x \cap a \cup b$ for $a, b \subseteq U$.

  Strictness: $f\,\emptyset = a \cap \emptyset \cup b = b = \emptyset$ whenever $b = \emptyset$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \mathrm{inc}\,x = x + 1$

The function $f : \mathbb{D}_1 \to \mathbb{D}_2$ is called

$f(x_1 \cap x_2) = a \cap x_1 \cap x_2 \cup b$

- distributive, if $f\,(\bigsqcup X) = \bigsqcup\{f\,x \mid x \in X\}$ for all $\emptyset \neq X \subseteq \mathbb{D}$;
- strict, if $f\,\bot = \bot$.
- totally distributive, if f is distributive and strict.

$f(x_1) \cap f(x_2) = (a \cap x_1 \cup b) \cap (a \cap x_2 \cup b)$

Examples

$= a \cap x_1 \cap x_2 \cup a \cap x_1 \cap b \cup a \cap x_2 \cap b \cup b$

- $f\,x = x \cap a \cup b$ for $a, b \subseteq U$.

Strictness: $f\,\emptyset = a \cap \emptyset \cup b = b = \emptyset$ whenever $b = \emptyset$

Distributivity:

$$\begin{aligned} f\,(x_1 \cup x_2) &= a \cap (x_1 \cup x_2) \cup b \\ &= a \cap x_1 \cup a \cap x_2 \cup b \\ &= f\,x_1 \cup f\,x_2 \end{aligned}$$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \mathsf{inc}\,x = x + 1$

$f(x_1 \cap x_2) = a \cap x_1 \cap x_2 \cup b$

$f(x_1) \cap f(x_2) = (a \cap x_1 \cup b) \cap (a \cap x_2 \cup b)$

$= a \cap x_1 \cap x_2 \cup a \cap x_1 \cap b \cup a \cap x_2 \cap b \cup b$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \mathsf{inc}\,x = x + 1$

Strictness: $f\,\bot = \mathsf{inc}\,0 = 1 \neq \bot$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \mathsf{inc}\,x = x + 1$

Strictness: $f\,\bot = \mathsf{inc}\,0 = 1 \neq \bot$

Distributivity: $f\,(\bigsqcup X) = \bigsqcup\{x + 1 \mid x \in X\}$ for $\emptyset \neq X$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \text{inc}\, x = x + 1$

  Strictness: $f\,\bot = \text{inc}\, 0 = 1 \quad \neq \quad \bot$

  Distributivity: $f\,(\bigsqcup X) \quad = \quad \bigsqcup\{x+1 \mid x \in X\}$ for $\emptyset \neq X$

- $\mathbb{D}_1 = (\mathbb{N} \cup \{\infty\})^2$, $\quad \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad f\,(x_1, x_2) = x_1 + x_2$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \text{inc}\, x = x + 1$

  Strictness: $f\,\bot = \text{inc}\, 0 = 1 \quad \neq \quad \bot$

  Distributivity: $f\,(\bigsqcup X) \quad = \quad \bigsqcup\{x+1 \mid x \in X\}$ for $\emptyset \neq X$

- $\mathbb{D}_1 = (\mathbb{N} \cup \{\infty\})^2$, $\quad \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad f\,(x_1, x_2) = x_1 + x_2$:

  Strictness: $f\,\bot = 0 + 0 \quad = \quad 0$

- $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad \text{inc}\, x = x + 1$

  Strictness: $f\,\bot = \text{inc}\, 0 = 1 \quad \neq \quad \bot$

  Distributivity: $f\,(\bigsqcup X) \quad = \quad \bigsqcup\{x+1 \mid x \in X\}$ for $\emptyset \neq X$

- $\mathbb{D}_1 = (\mathbb{N} \cup \{\infty\})^2$, $\quad \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}$, $\quad f\,(x_1, x_2) = x_1 + x_2$:

  Strictness: $f\,\bot = 0 + 0 \quad = \quad 0$

  Distributivity:

$$\begin{array}{rcl} f\,((1,4) \sqcup (4,1)) & = & f\,(4,4) \quad = \quad 8 \\ & \neq & 5 \quad = \quad f\,(1,4) \sqcup f\,(4,1) \end{array}$$

Remark

If $\quad f : \mathbb{D}_1 \to \mathbb{D}_2 \quad$ is distributive, then also monotonic.

Remark

If $f : \mathbb{D}_1 \to \mathbb{D}_2$ is distributive, then also monotonic.

Remark

If $f : \mathbb{D}_1 \to \mathbb{D}_2$ is distributive, then also monotonic.

Obviously: $a \sqsubseteq b$ iff $a \sqcup b = b$.

Remark

If $f : \mathbb{D}_1 \to \mathbb{D}_2$ is distributive, then also monotonic.

Obviously: $a \sqsubseteq b$ iff $a \sqcup b = b$.

Remark

If $f : \mathbb{D}_1 \to \mathbb{D}_2$ is distributive, then also monotonic.

Obviously: $a \sqsubseteq b$ iff $a \sqcup b = b$.

From that follows:

$$\begin{array}{rrcl} & f\,b & = & f\,(a \sqcup b) \\ & & = & f\,a \sqcup f\,b \\ \Longrightarrow & f\,a & \sqsubseteq & f\,b \end{array}$$

Assumption: all v are reachable from *start* .

Assumption: all v are reachable from *start* .
Then:

Theorem

Kildall 1972

If all effects of edges $[\![k]\!]^\sharp$ are distributive, then: $\mathcal{I}^*[v] = \mathcal{I}[v]$
for all v .

Assumption: all v are reachable from *start* .
Then:

Theorem

Kildall 1972

If all effects of edges $[\![k]\!]^\sharp$ are distributive, then: $\mathcal{I}^*[v] = \mathcal{I}[v]$
for all v .

Proof

It suffices to prove that $\mathcal{I}^*$ is a solution!

For this, we show that $\mathcal{I}^*$ satisfies all constraints.

(1) We prove for *start* :

$$\begin{array}{lll} \mathcal{I}^*[start] & = & \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \to^* start\} \\ & \sqsupseteq & [\![\epsilon]\!]^\sharp\, d_0 \\ & \sqsupseteq & d_0 \end{array}$$

(1) We prove for *start* :

$$\begin{array}{lll} \mathcal{I}^*[start] & = & \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \to^* start\} \\ & \sqsupseteq & [\![\epsilon]\!]^\sharp\, d_0 \\ & \sqsupseteq & d_0 \end{array}$$

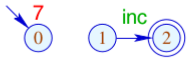
(2) For every $k = (u, \_, v)$ we prove:

$$\begin{array}{lll} \mathcal{I}^*[v] & = & \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \to^* v\} \\ & \sqsupseteq & \bigsqcup\{[\![\pi' k]\!]^\sharp\, d_0 \mid \pi' : start \to^* u\} \\ & = & \bigsqcup\{[\![k]\!]^\sharp\,([\![\pi']\!]^\sharp\, d_0) \mid \pi' : start \to^* u\} \\ & = & [\![k]\!]^\sharp\,(\bigsqcup\{[\![\pi']\!]^\sharp\, d_0 \mid \pi' : start \to^* u\}) \\ & = & [\![k]\!]^\sharp\,(\mathcal{I}^*[u]) \end{array}$$

since $\{\pi' \mid \pi' : start \to^* u\}$ is non-empty.

Caveat

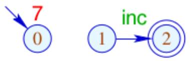
- **Reachability** of all program points cannot be abandoned! Consider:



where $\mathbb{D} = \mathbb{N} \cup \{\infty\}$

Caveat

- **Reachability** of all program points cannot be abandoned! Consider:



where $\mathbb{D} = \mathbb{N} \cup \{\infty\}$

Then:

$$\begin{array}{lllll} \mathcal{I}[2] & = & \mathsf{inc}\; 0 & = & 1 \\ \mathcal{I}^*[2] & = & \bigsqcup \emptyset & = & 0 \end{array}$$

Summary and Application

$x \cap a \cup b$

$(x \cup a) \setminus b$

→ The effects of edges of the analysis of **availability of expressions** are distributive:

$$\begin{array}{lll} (a \cup (x_1 \cap x_2)) \setminus b & = & ((a \cup x_1) \cap (a \cup x_2)) \setminus b \\ & = & ((a \cup x_1) \setminus b) \cap ((a \cup x_2) \setminus b) \end{array}$$

Summary and Application

→ The effects of edges of the analysis of availability of expressions are distributive:

$$\begin{array}{lll} (a \cup (x_1 \cap x_2)) \backslash b & = & ((a \cup x_1) \cap (a \cup x_2)) \backslash b \\ & = & ((a \cup x_1) \backslash b) \cap ((a \cup x_2) \backslash b) \end{array}$$

→ If all effects of edges are distributive, then the MOP can be computed by means of the constraint system and RR-iteration.

Summary and Application

→ The effects of edges of the analysis of availability of expressions are distributive:

$$\begin{array}{lll} (a \cup (x_1 \cap x_2)) \backslash b & = & ((a \cup x_1) \cap (a \cup x_2)) \backslash b \\ & = & ((a \cup x_1) \backslash b) \cap ((a \cup x_2) \backslash b) \end{array}$$

→ If all effects of edges are distributive, then the MOP can be computed by means of the constraint system and RR-iteration.

→ If not all effects of edges are distributive, then RR-iteration for the constraint system at least returns a safe upper bound to the MOP.

1.2 Removing Assignments to Dead Variables

Example:

$$\begin{array}{ll} 1: & x = y + 2; \\ 2: & y = 5; \\ 3: & x = y + 3; \end{array}$$

The value of x at program points 1, 2 is over-written before it can be used.

Therefore, we call the variable x dead at these program points :-)

1.2 Removing Assignments to Dead Variables

Example:

$$\begin{array}{ll} 1: & x = y + 2; \\ 2: & y = 5; \\ 3: & x = y + 3; \end{array}$$

The value of x at program points 1, 2 is over-written before it can be used.

Therefore, we call the variable x dead at these program points :-)

Note:

→ Assignments to dead variables can be removed ;-)

→ Such inefficiencies may originate from other transformations.

Note:

→ Assignments to dead variables can be removed ;-)

→ Such inefficiencies may originate from other transformations.

Formal Definition:

The variable x is called live at u along the path π starting at u relative to a set X of variables either:

if $x \in X$ and π does not contain a definition of x; or:

if π can be decomposed into: $\pi = \pi_1\, k\, \pi_2$ such that:

- k is a use of x ; and
- π_1 does not contain a definition of x.

Note:

→ Assignments to dead variables can be removed ;-)

→ Such inefficiencies may originate from other transformations.

Formal Definition:

The variable x is called live at u along the path π starting at u relative to a set X of variables either:

if $x \in X$ and π does not contain a definition of x; or:

if π can be decomposed into: $\pi = \pi_1\, k\, \pi_2$ such that:

- k is a use of x ; and
- π_1 does not contain a definition of x.

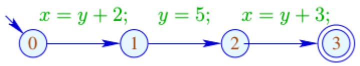
194

Thereby, the set of all defined or used variables at an edge $k = (\_, lab, \_)$ is defined by:

lab	used	defined
;	$\emptyset$	$\emptyset$
$\text{Pos}\,(e)$	$\mathit{Vars}\,(e)$	$\emptyset$
$\text{Neg}\,(e)$	$\mathit{Vars}\,(e)$	$\emptyset$
$x = e;$	$\mathit{Vars}\,(e)$	$\{x\}$
$x = M[e];$	$\mathit{Vars}\,(e)$	$\{x\}$
$M[e_1] = e_2;$	$\mathit{Vars}\,(e_1) \cup \mathit{Vars}\,(e_2)$	$\emptyset$

A variable x which is not live at u along π (relative to X) is called dead at u along π (relative to X).

Example:



where $X = \emptyset$. Then we observe:

	live	dead
0	$\{y\}$	$\{x\}$
1	$\emptyset$	$\{x, y\}$
2	$\{y\}$	$\{x\}$
3	$\emptyset$	$\{x, y\}$

The variable x is live at u (relative to X) if x is live at u along some path to the exit (relative to X). Otherwise, x is called dead at u (relative to X).

The variable x is live at u (relative to X) if x is live at u along some path to the exit (relative to X). Otherwise, x is called dead at u (relative to X).

Question:

How can the sets of all dead/live variables be computed for every u ???

The variable x is live at u (relative to X) if x is live at u along some path to the exit (relative to X). Otherwise, x is called dead at u (relative to X).

Question:

How can the sets of all dead/live variables be computed for every u ???

Idea:

For every edge $k = (u, \_, v)$, define a function $[\![k]\!]^\sharp$ which transforms the set of variables which are live at v into the set of variables which are live at u ...

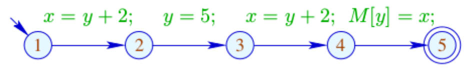
Let $\mathbb{L} = 2^{Vars}$.

For $k = (\_, lab, \_)$, define $[\![k]\!]^\sharp = [\![lab]\!]^\sharp$ by:

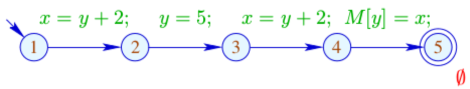
$$
\begin{array}{lcl}
[\![;]\!]^\sharp\, L & = & L \\
[\![\mathrm{Pos}(e)]\!]^\sharp\, L & = & [\![\mathrm{Neg}(e)]\!]^\sharp\, L \;=\; L \cup Vars(e) \\
[\![x = e;]\!]^\sharp\, L & = & (L \backslash \{x\}) \cup Vars(e) \\
[\![x = M[e];]\!]^\sharp\, L & = & (L \backslash \{x\}) \cup Vars(e) \\
[\![M[e_1] = e_2;]\!]^\sharp\, L & = & L \cup Vars(e_1) \cup Vars(e_2)
\end{array}
$$

$$[\![x = x+1;]\!]^\sharp(\{x, y\})$$

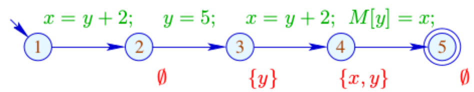
We verify that these definitions are meaningful :-)



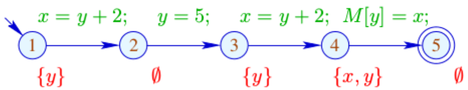
We verify that these definitions are meaningful :-)



We verify that these definitions are meaningful :-)



We verify that these definitions are meaningful :-)



The set of variables which are live at u then is given by:

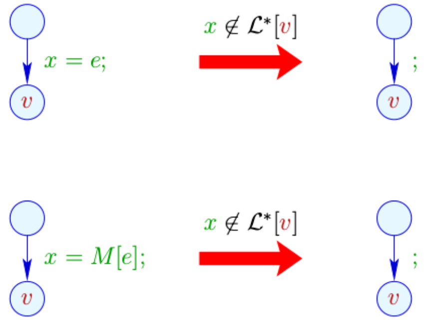
$$\mathcal{L}^*[u] \;=\; \bigcup \{ [\![\pi]\!]^\sharp \, X \mid \pi : u \to^* stop \}$$

... literally:

- The paths start in u :-)

  $\Longrightarrow$ As partial ordering for $\mathbb{L}$ we use $\sqsubseteq \, = \, \subseteq$.
- The set of variables which are live at program exit is given by the set X :-)

Transformation 2:



Correctness Proof:

→ Correctness of the effects of edges: If L is the set of variables which are live at the exit of the path π , then $[\![\pi]\!]^\sharp \, L$ is the set of variables which are live at the beginning of π :-)

→ Correctness of the transformation along a path: If the value of a variable is accessed, this variable is necessarily live. The value of dead variables thus is irrelevant :-)

→ Correctness of the transformation: In any execution of the transformed programs, the live variables always receive the same values :-))

Computation of the sets $\mathcal{L}^*[u]$:

(1) Collecting constraints:

$$\begin{array}{lll} \mathcal{L}[stop] & \supseteq & X \\ \mathcal{L}[u] & \supseteq & [\![k]\!]^\sharp(\mathcal{L}[v]) \qquad k = (u, \_, v) \quad \text{edge} \end{array}$$

(2) Solving the constraint system by means of RR iteration.

Since $\mathbb{L}$ is finite, the iteration will terminate :-)

(3) If the exit is (formally) reachable from every program point, then the smallest solution $\mathcal{L}$ of the constraint system equals $\mathcal{L}^*$ since all $[\![k]\!]^\sharp$ are distributive :-))

Transformation 2: