Analyse und Verifikation
(185.276, VU 2.0, ECTS 3.0)

Übungsblatt 5

Bernhard Urban
Matr.Nr.: 0725771 KNZ: 067 937
lewurm@gmail.com

Thomas Reinbacher
Matr.Nr.: 0828472 KNZ: 786 881
treinbacher@ecs.tuwien.ac.at

24.05.2011

Aufgabe 1:

Es ist zu zeigen, dass "Simple Constants" ein monotones, aber kein distributives DFA-Problem ist.

monoton : Das DFA-Funktional $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$ heißt monoton genau dann wenn $\forall e \in E.\ [\![e]\!]$ monoton ist (vgl. Folie 295). D.h. wir müssen für jede Kante zeigen, dass das Funktional monoton ist; eine Funktion $f : \mathcal{C} \to \mathcal{C}$ heißt monoton genau dann wenn $\forall c, c' \in \mathcal{C}.\ c \sqsubseteq c' \succ f(c) \sqsubseteq f(c')$ (vgl. Folie 293).
Für "Simple Constants" ist das DFA-Funktional $[\![\ ]\!]_{sc} : E \to (\Sigma \to \Sigma)$ definiert durch (Folie 309)

$$\forall e \in E.\ [\![e]\!]_{sc} =_{df} \theta_e$$

Die Zustandstransformationsfunktion (Folie 317) $\theta_e : \Sigma \to \Sigma$ (und $e \equiv x := t$) ist definiert durch

$$\forall \sigma \in \Sigma.\ \forall y \in \mathbf{V}.\ \theta_e(\sigma)(y) =_{df} \begin{cases} \mathcal{E}(t)(\sigma) & \text{falls } y = x \\ \sigma(y) & \text{sonst} \end{cases}$$

Σ stellt dabei die Menge der Zustände dar (Folie 316)

$$\Sigma =_{df} \{\sigma | \sigma : \mathbf{V} \to \mathbf{D}\}$$

Weiters gilt:

$$\begin{array}{rll} \sigma_\bot \in \Sigma: & \forall v \in \mathbf{V}.\ \sigma_\bot(v) & = \bot \\ \sigma_c \in \Sigma: & \exists v \in \mathbf{V}.\ \sigma_c(v) & \neq \bot \neq \top \\ \sigma_\top \in \Sigma: & \forall v \in \mathbf{V}.\ \sigma_\top(v) & = \top \end{array}$$

Zwei Mengen von Zuständen stehen in "Simple Constants" in Relation wenn:

$$\forall c, c' \in \Sigma : c \sqsubseteq c' \quad \text{iff } \forall x \in \mathbf{V} : c(x) \sqsubseteq c'(x)$$

Um zu zeigen dass θ_e monoton ist, müssen wir daher folgende Kombinationen zeigen:

#	c	c'	$\forall e \in E$
(1)	$\sigma_\perp$	$\sigma_\perp$	$\sigma_\perp \sqsubseteq \sigma_\perp \succ [\![e]\!]_{sc}(\sigma_\perp) \sqsubseteq [\![e]\!]_{sc}(\sigma_\perp)$
(2)	$\sigma_\perp$	σ_c	$\sigma_\perp \sqsubseteq \sigma_c \succ [\![e]\!]_{sc}(\sigma_\perp) \sqsubseteq [\![e]\!]_{sc}(\sigma_c)$
(3)	$\sigma_\perp$	$\sigma_\top$	$\sigma_\perp \sqsubseteq \sigma_\top \succ [\![e]\!]_{sc}(\sigma_\perp) \sqsubseteq [\![e]\!]_{sc}(\sigma_\top)$
(4)	σ_c	σ_d	$\sigma_c \sqsubseteq \sigma_d \succ [\![e]\!]_{sc}(\sigma_c) \sqsubseteq [\![e]\!]_{sc}(\sigma_d)$
(5)	σ_c	$\sigma_\top$	$\sigma_c \sqsubseteq \sigma_\top \succ [\![e]\!]_{sc}(\sigma_c) \sqsubseteq [\![e]\!]_{sc}(\sigma_\top)$
(6)	$\sigma_\top$	$\sigma_\top$	$\sigma_\top \sqsubseteq \sigma_\top \succ [\![e]\!]_{sc}(\sigma_\top) \sqsubseteq [\![e]\!]_{sc}(\sigma_\top)$

Für weitere Erklärungen nehmen wir $[\![e]\!]_{sc} = f$ an. Die Fälle **(1)** und **(6)** sind trivial, da $c = c' \succ f(c) = f(c')$.
Bei Fall **(2)** kann c höchstens durch eine Zuweisung von einer Konstante zu $f(c) \neq \sigma_\perp$ werden. Im schlimmsten Fall ändert sich c' durch die Zuweisung der Konstante gar nicht (d.h. $c' = f(c')$) und es gilt $f(c) = f(c')$, ansonsten $f(c) \sqsubseteq f(c')$.
Eine andere Möglichkeit wäre dass c' durch die die Funktionsanwendung zu $\sigma_\perp$ wird (vgl. strikte Interpretation auf Folie 315); c ändert sich durch die Funktionsapplikation nicht. Die Ordnung bleibt in beiden Fällen erhalten.
Fall **(3)**: trivial.
Fall **(4)**: $\sigma_c \sqsubseteq \sigma_d$ kann nur dann gelten, wenn $c = d$, daher gilt $\sigma_c = \sigma_d$.
Fall **(5)**: trivial.
(Im Appendix findet sich eine alternative Argumentation).

nicht distributiv : Eine Funktion $f : \mathcal{C} \to \mathcal{C}$ heißt distributiv genau dann wenn $\forall C' \subseteq \mathcal{C}.\ f(\bigsqcap C') = \bigsqcap\{f(c) \mid c \in C'\}$
Durch ein Gegenbeispiel soll gezeigt werden, dass die Zustandstransformation von "Simple Constants" nicht distributiv ist. Wir nehmen an, dass $\{x \mapsto -1, x \mapsto 1\} = C' \subseteq \Sigma$ und $e \equiv y := x * x$; gilt.

$$\theta_{y:=x*x}(\{x \mapsto -1\} \sqcap \{x \mapsto 1\}) \neq \theta_{y:=x*x}(\{x \mapsto -1\}) \sqcap \theta_{y:=x*x}(\{x \mapsto 1\})$$
$$\theta_{y:=x*x}(\{x \mapsto \perp\}) \neq \{y \mapsto 1\} \sqcap \{y \mapsto 1\}$$
$$\{y \mapsto \perp\} \neq \{y \mapsto 1\}$$

nicht distributiv per Koinzidenztheorem : Für ein distributives Framwork gilt laut dem Koinzidenztheorem (Folie 299) stets:

$$\forall c_s \in \mathcal{C}\ \forall n \in N.\ \mathsf{MaxFP}_{c_s}(n) = \mathsf{MOP}_{c_s}(n)$$

Wir werden nun ein Gegenbeispiel in Form eines knotenbenannten Flussgraphen konstruieren das eine Verletzung des Koinzidenztheorems für ein "Simple Constants" Problem

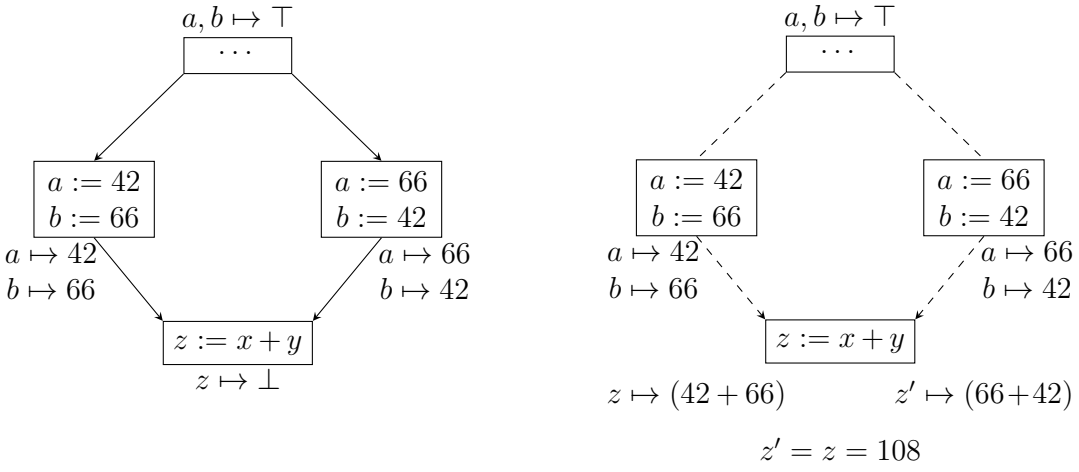


Abbildung 1: Iterative, MaxFP Lösung links und MOP Lösung rechts

zeigt. Wir betrachten dabei das Programm π in Abbildung 1. Die Anwendung des iterativen Fixpunktalgorithmus aus Folie 290 weist der Programvariable z am Ende des if-else Konstrukts das $\bot$ Element zu. Die MOP Lösung hingegen erkennt das der Ausdruck $a+b$ auf den beiden Pfaden den gleichen Wert 108 ergibt. Somit ist für dieses Beispiel das Koinzidenztheorem nicht mehr gültig und daher die nicht-distributivität von "Simple Constants" gezeigt.

Aufgabe 2:

Wir wollen zeigen dass $\forall c_s \in \mathcal{C}\ \forall n \in N.\ \mathsf{MaxFP}_{c_\mathbf{s}}(n) = \mathsf{MOP}_{c_\mathbf{s}}(n)$ falls das DFA-funktional $[\![\ ]\!]$ distributiv ist. Es gilt die folgenden Inklusionen zu zeigen:

$$\begin{aligned} &\forall c_s \in \mathcal{C}\ \forall n \in N: \\ &\qquad (1)\quad \mathsf{MaxFP}_{c_\mathbf{s}}(n) \sqsubseteq \mathsf{MOP}_{c_\mathbf{s}}(n) \quad \bigwedge \\ &\qquad (2)\quad \mathsf{MaxFP}_{c_\mathbf{s}}(n) \sqsupseteq \mathsf{MOP}_{c_\mathbf{s}}(n) \end{aligned}$$

Definitionen, Feststellungen: $f : \mathcal{C} \to \mathcal{C}$ heißt distributiv genau dann wenn $\forall C' \subseteq \mathcal{C}.\ f(\bigsqcap C') = \bigsqcap\{f(c) \mid c \in C'\}$, d.h., $f(c_1 \sqcap c_2) = f(c_1) \sqcap f(c_2)$. n bezeichnet die Länge der Pfade für die $\mathsf{MOP}_{c_\mathbf{s}}(n)$ berechnet wird, wobei wir annehmen, dass $n \geq 0$. $\mathsf{MOP}_{c_\mathbf{s}}(n)$ ist auf Folie 284 definiert. $\mathsf{MaxFP}_{c_\mathbf{s}}(n)$ definieren wir aus Notationsgründen anders:

$$\mathsf{MaxFP}_{c_\mathbf{s}}(n) = \bigsqcap\{[\![(m,n)]\!](\mathsf{MaxFP}_{c_\mathbf{s}}(m)) \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E}$$

$$c^n_{\mathbf{s}_E} = \begin{cases} c_\mathbf{s} & \text{if } n = \mathbf{s} \\ \top & \text{if } n \neq \mathbf{s} \end{cases}$$

(1) Die erste Inklusion wird per Induktion über die Pfadlänge gezeigt. Die Hypothese lautet:

$$\forall n : \mathsf{MaxFP}_{c_\mathbf{s}}(n) \sqsubseteq \mathsf{MOP}_{c_\mathbf{s}}(n)$$

Der Basisfall $n = 0$, $\mathsf{MaxFP}_{c_\mathbf{s}}(0) \sqsubseteq \mathsf{MOP}_{c_\mathbf{s}}(0)$, ist trivialerweise erfüllt. Der Induktionsschritt ist wie folgt:

$$
\begin{aligned}
\mathsf{MaxFP}_{c_\mathbf{s}}(n) &= \bigsqcap\{[\![(m,n)]\!](\mathsf{MaxFP}_{c_\mathbf{s}}(m)) \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E} \\
\text{Hypothese einsetzen} \quad &\sqsubseteq \bigsqcap\{[\![(m,n)]\!](\mathsf{MOP}_{c_\mathbf{s}}(m)) \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E} \\
&= \bigsqcap\{[\![(m,n)]\!](\bigsqcap\{[\![p]\!](c_\mathbf{s}) \mid p \in \mathbf{P}[\mathbf{s},m]\}) \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E} \\
&\sqsubseteq \bigsqcap\{\bigsqcap\{[\![(m,n)]\!]([\![p]\!](c_\mathbf{s})) \mid p \in \mathbf{P}[\mathbf{s},m]\} \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E} \\
&= \bigsqcap\{[\![p]\!](c_\mathbf{s}) \mid p \in \mathbf{P}[\mathbf{s},n]\} \\
&= \mathsf{MOP}_{c_\mathbf{s}}(n)
\end{aligned}
$$

(2) $\mathsf{MOP}_{c_\mathbf{s}}(n)$ ist lt. Folie 284 wie folgt definiert:

$$\mathsf{MOP}_{c_\mathbf{s}}(n) = \bigsqcap\{[\![p]\!](c_s) \mid p \in \mathbf{P}[\mathbf{s},n]\}$$

Intuition: In einem distributiven Framework verliert man durch das "vorzeitige" Anwenden von $\bigsqcap$ keine Genauigkeit. $\mathsf{MaxFP}_{c_\mathbf{s}}$ berechnet die größte Lösung der Datenflussgleichungen. Es gilt nun zu zeigen, dass $\mathsf{MOP}_{c_\mathbf{s}}$ ebenfalls eine Lösung der Datenflussgleichung ist. Danach kann mit dem Resultat des Fixpunktsatzes von Knaster/Tarski, Kleene auf Folie 124 die Inklusion gezeigt werden.

Aus der Distributivität folgt:

$$
\begin{aligned}
\forall C' \subseteq \mathcal{C}.\ f(\bigsqcap C') &= \bigsqcap\{f(c) \mid c \in C'\} \\
\forall n \geq 0.\ [\![(m,n)]\!](\bigsqcap\{[\![p]\!](c_\mathbf{s}) \mid p \in \mathbf{P}[\mathbf{s},m]\}) &= \bigsqcap\{[\![(m,n)]\!]([\![p]\!](c_\mathbf{s}) \mid p \in \mathbf{P}[\mathbf{s},m])\} \\
&= \bigsqcap\{[\![p]\!](c_\mathbf{s}) \mid p \in \mathbf{P}[\mathbf{s},n]\}
\end{aligned}
$$

Angewendet auf die Definition von $\mathsf{MOP}_{c_\mathbf{s}}$:

$$
\begin{aligned}
\mathsf{MOP}_{c_\mathbf{s}}(n) &= \bigsqcap\{[\![p]\!](c_\mathbf{s}) \mid p \in \mathbf{P}[\mathbf{s},n]\} \\
&= \bigsqcap(\{[\![(m,n)]\!]([\![p]\!](c_\mathbf{s})) \mid p \in \mathbf{P}[\mathbf{s},m]\ ,\ m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E}) \\
&= \bigsqcap(\{[\![(m,n)]\!](\underbrace{\bigsqcap\{[\![p]\!](c_\mathbf{s})) \mid p \in \mathbf{P}[\mathbf{s},m]\}}_{\mathsf{MOP}_{c_\mathbf{s}}(m)} \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E}) \\
&= \bigsqcap(\{[\![(m,n))]\!](\mathsf{MOP}_{c_\mathbf{s}}(m)) \mid m \in \mathrm{pred}(n)\} \sqcap c^n_{\mathbf{s}_E}
\end{aligned}
$$

Betrachten wir nun den generischen Fixpunkt Algorithmus von Folie 290, so sehen wir dass die $\mathsf{MaxFP}_{c_\mathbf{s}}$ Lösung Schritt für Schritt berechnet wird. Durch die obenstehende

Umformung haben wir gezeigt, dass wenn das DFA Problem distributiv ist, die $\mathsf{MOP}_{c_\mathbf{s}}$ Lösung ebenfalls iterativ berechnet werden kann.
Da nun $\mathsf{MOP}_{c_\mathbf{s}}$ ebenfalls eine Lösung der Datenflussgleichungen ist, und wir wissen, dass $\mathsf{MaxFP}_{c_\mathbf{s}}$ die größte Lösung der Datenflussgleichungen errechnet, dann muss folgendes gelten:

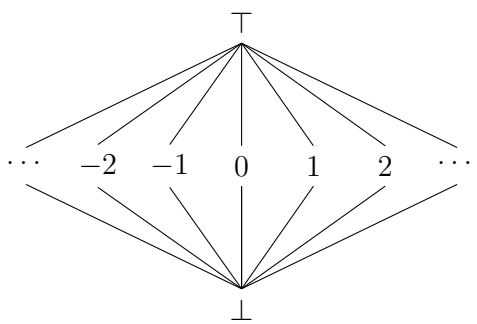
$$\forall c_s \in \mathcal{C}\ \forall n \in N: \quad \mathsf{MaxFP}_{c_\mathbf{s}}(n) \sqsupseteq \mathsf{MOP}_{c_\mathbf{s}}(n)$$

Damit wurden die beiden notwendigen Inklusionen gezeigt, und es gilt: $\forall c_s \in \mathcal{C}\ \forall n \in N : \mathsf{MaxFP}_{c_\mathbf{s}}(n) = \mathsf{MOP}_{c_\mathbf{s}}(n)$ wenn ein distributives DFA Problem vorliegt.

Appendix

Eine alternative Argumentation, warum "Simple Constraints" monoton ist. Für den Fall $y = x$ kann nun für eine konkrete Transferfunktion $q = a \bowtie b$ exemplarisch gezeigt werden, dass für jeden möglichen Input für a und b die Auswertung von $\mathcal{E}(a \bowtie b)(\sigma)$ stets kleiner als a ist mit fallenden Kombinationen für b. Das Symbol $\bowtie$ steht für die Operatoren $\{+, -, \div, \times, \dots\}$.

$a(\sigma)$	$b(\sigma)$	$\mathcal{E}(a \bowtie b)(\sigma)$
$\top$	$\top$	$\top$
$\top$	c_b	$\top$
$\top$	$\bot$	$\bot$
c_a	$\top$	$\top$
c_a	c_b	$c_a \bowtie c_b$
c_a	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	c_b	$\bot$
$\bot$	$\bot$	$\bot$



Um nun zu zeigen, dass das DFA-Funktional $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$ monoton ist, stellen wir folgendes fest:

- Für alle möglichen Input Werte für $a(\sigma)$ wird $\mathcal{E}(a \bowtie b)(\sigma)$ nicht größer wenn der Input für $b(\sigma)$ kleiner wird (kleiner und großer ist definiert über das obige Halbordnung-Diagramm).
- Bsp. für $a(\sigma) = \top$ und $b(\sigma) = \top$ ist $\mathcal{E}_0(a \bowtie b)(\sigma) = \top$. Wird nun für b ein kleinerer Wert gewählt (etwa 5), so evaluiert $\mathcal{E}_1(a \bowtie b)(\sigma) = \top$. Setzt man nun für b das bottom element $\bot$ ein so evaluiert $\mathcal{E}_2(a \bowtie b)(\sigma)$ zu $\bot$. Wir erkennen, dass

$$\mathcal{E}_2(a \bowtie b)(\sigma) \leq \mathcal{E}_1(a \bowtie b)(\sigma) \leq \mathcal{E}_0(a \bowtie b)(\sigma)$$

- Aus der obigen Tabelle ist ersichtlich, dass dies auch für alle anderen Kombinationen gültig ist, daher ist "Simple Constants" ein monotones DFA.