7.1 Iteratieve canonieke vormen

- Interpretatie van programma's
- Nogmaals canonieke vormen
- Expressiviteit van PGA
- Geavanceerde constructies en instructies

7.1 Iteratieve canonieke vormen

Voor een tweede canonieke vorm

$$u_1; \ldots; u_m; (v_1; \ldots; v_k)^\omega$$

definiëren we de bijbehorende gedragsequivalente *iteratieve canonieke vorm* als

$$(u_1; \ldots; u_m; \tilde{v}_1; \ldots; \tilde{v}_k; \#m+2; \#m+2)^\omega$$

$$\text{met } \tilde{v}_i = \begin{cases} \#n + m + 2 & \text{als } v_i = \#n \text{ en } i + n > k, \\ v_i & \text{anders.} \end{cases}$$

7.1 Iteratieve canonieke vormen

Een voorbeeld:

$$|+a; \#6; +b; \#5; c; (+d; \#3; !; -f)^\omega| \;=$$
$$|(+a; \#6; +b; \#5; c; +d; \#10; !; -f; \#7; \#7)^\omega|.$$

Beide expressies definiëren het volgende gedrag P:

$$\begin{aligned} P &= S \trianglelefteq a \trianglerighteq (P_f \trianglelefteq b \trianglerighteq c \circ P_d), \\ P_f &= P_d \trianglelefteq f \trianglerighteq P_d, \\ P_d &= P_d \trianglelefteq d \trianglerighteq S. \end{aligned}$$

Merk op: het gebruik van de *tweede* canonieke vorm is noodzakelijk

$$|\#5; (a; +b)^\omega| \neq |(\#5; a; +b; \#3; \#3)^\omega|.$$

7.2 Expressiviteit van PGA

Drie karakteristieken van het ontwerp van PGA:

(1) Negatieve testinstructies zijn *overbodig* in termen van expressiviteit.

Transformeer een iteratieve canonieke vorm $(u_1; \ldots; u_k)^\omega$ als volgt:

Verhoog elke jump *over* een $-a$ in $u_1; \ldots; u_k$ met 1, en vervang $-a$ dan door $+a; \#2$.

7.2 Expressiviteit van PGA

Een voorbeeld:

$$(-a; -b; c; +d; \#2)^\omega$$

$$\Downarrow$$

$$(+a; \#2; -b; c; +d; \#3)^\omega$$

$$\Downarrow$$

$$(+a; \#3; +b; \#2; c; +d; \#3)^\omega.$$

7.2 Expressiviteit van PGA

(2) Kleine jumptellers zijn *overbodig* in termen van expressiviteit.

Transformeer een iteratieve canonieke vorm $(u_1; \ldots; u_k)^\omega$ als volgt:

Vervang elke jump $\#n$ in $u_1; \ldots; u_k$ m.b.v. PGA1,4 en PGA7 door $\#n + k$.

Merk op: We kunnen het gedrag van elk programma programmeren zonder gebruik te maken van jumps met teller kleiner dan bijvoorbeeld 1073.

7.2 Expressiviteit van PGA

(3) Onbeperkt grote jumptellers zijn $noodzakelijk$ in termen van expressiviteit.

Preciezer: Zij PGA$_n$ de verzameling PGA-expressies zonder jumps met teller groter dan n.

Stelling. *Voor elk natuurlijk getal $n \geq 2$ bestaat er een gedrag dat niet in PGA$_n$ is uit te drukken en wel in PGA.*

7.3 PGA-programma's en regulier gedrag

- In PGA kan oneindig gedrag uitsluitend worden uitgedrukt indien het *periodiek* is (ook wel *regulier* genoemd).
- Voorbeeld van een oneindig gedrag dat niet periodiek is:

$$a; b; a; b^2; a; b^3; a; b^4; \ldots$$

We kunnen dit gedrag beschrijven als $(P_n)_{n\in\mathbb{N}}$ met $P_0 = D$, $P_1 = a \circ D$ en voor $k \geq 1$,

$$P_{k+1} = \begin{cases} \alpha \circ a \circ D & \text{als } P_k = \alpha \circ D \text{ en } k+1 = \frac{1}{2}n(n+1) \text{ voor zekere } n, \\ \alpha \circ b \circ D & \text{als } P_k = \alpha \circ D \text{ en } \forall n(k+1 \neq \frac{1}{2}n(n+1)). \end{cases}$$

7.3 PGA-programma's en regulier gedrag

$\{E_i = t_i \mid i \in I\}$ is een *lineair* stelsel gedragsvergelijkingen indien $t_i \in \{S, D\}$ of $t_i = E_j \trianglelefteq a_i \trianglerighteq E_k$ met $j, k \in I$ en $a_i \in A$.

Voorbeeld:

$$\begin{aligned} E_1 &= E_2 \trianglelefteq a \trianglerighteq E_3, \\ E_2 &= D, \\ E_3 &= E_4 \trianglelefteq b \trianglerighteq E_2, \\ E_4 &= E_5 \trianglelefteq c \trianglerighteq E_6, \\ E_5 &= S, \\ E_6 &= E_1 \trianglelefteq d \trianglerighteq E_1 (= d \circ E_1). \end{aligned}$$

P heet *regulier* indien er een *eindig* lineair stelsel $\{E_i = t_i \mid i \in I\}$ bestaat met $P = E_j$ voor zekere $j \in I$.

Voorbeeld: $P = |(+a; \#2; -b; \#0; +c; !; d)^\omega|$ is regulier, want $P = E_1$.

7.3 PGA-programma's en regulier gedrag

Elk PGA-programma definieert een regulier gedrag:

Voorbeeld $X = (+a; \#2; -b; \#0; +c; !; d)^\omega$:

$$
\begin{array}{lcll}
E_1 & = & |+a; \#2; -b; \#0; +c; !; d; X| & = E_2 \trianglelefteq a \trianglerighteq E_3, \\
E_2 & = & |\#2; -b; \#0; +c; !; d; X| & = E_4, \\
E_3 & = & |-b; \#0; +c; !; d; X| & = E_5 \trianglelefteq b \trianglerighteq E_4, \\
E_4 & = & |\#0; +c; !; d; X| & = D, \\
E_5 & = & |+c; !; d; X| & = E_6 \trianglelefteq c \trianglerighteq E_7, \\
E_6 & = & |!; d; X| & = S, \\
E_7 & = & |d; X| & = d \circ E_1.
\end{array}
$$

Door $E_2 = E_4$ te verwijderen en de overige E_4 te vervangen door E_2 ontstaat een lineair stelsel met $|X| = E_1$.

7.3 PGA-programma's en regulier gedrag

Andersom kan elk regulier gedrag in PGA worden geprogrammeerd:

Elke gedragsvergelijking vertaalt (notatie $\mapsto$) naar een PGLDg-programmafragment:

$$\begin{aligned} E_i = E_k \trianglelefteq a_i \trianglerighteq E_l \quad &\mapsto \quad \pounds i; +a_i; \#\#\pounds k; \#\#\pounds l, \\ E_i = S \quad &\mapsto \quad \pounds i; !, \\ E_i = D \quad &\mapsto \quad \pounds i; \#\#\pounds i. \end{aligned}$$

Aaneenschakeling van de vertaalde programmafragmenten levert een PGLDg-programma dat het gedrag E_1 direct beschrijft; projectie naar PGA levert vervolgens een gedragsequivalent PGA-programma.

7.4 De eenheidsoperator en conditionele keuze en repetitie

PGAu is de uitbreiding van PGA met de *eenheidsoperator* $u(X)$ die een PGA-programma X tot een eenheid van lengte 1 maakt.

Voorbeeld :

$$+a; u(b^\omega); c^\omega.$$

Gedragsextractie is eenvoudig te definiëren door aan de vergelijkingen voor gedragsextractie de volgende toe te voegen:

$$|u(X); Y| = |X; Y| \text{ en één van } |X| = |X; (\#0)^\omega|,\ |u(X)| = |X|.$$

Merk op: in plaats van een projectie van PGAu naar PGA te definiëren, kunnen we de eenheidsoperator direct *interpreteren*.

7.4 De eenheidsoperator en conditionele keuze en repetitie

PGAcw is de uitbreiding van PGA met eenvoudige vormen van de conditionele constructie en de while-loop:

Positieve conditionele keuze. $\texttt{if} + a$

Negatieve conditionele keuze. $\texttt{if} - a$

Positieve conditionele repetitie. $\texttt{while} + a$

Negatieve conditionele repetitie. $\texttt{while} - a$

7.4 De eenheidsoperator en conditionele keuze en repetitie

Interpretatie voor nieuwe instructies:

$$
\begin{aligned}
|\texttt{if} + a; u; v; X| &= |-a; \#3; u; \#2; v; X|,\\
|\texttt{if} - a; u; v; X| &= |+a; \#3; u; \#2; v; X|,\\
|\texttt{while} + a; u; X| &= |-a; \#4; u; \texttt{while} + a; u; X|,\\
|\texttt{while} - a; u; X| &= |+a; \#4; u; \texttt{while} - a; u; X|.
\end{aligned}
$$

Combinatie met de eenheidsoperator leidt tot de gebruikelijke constructies:

$$\texttt{if} + a; u(X); u(Y)$$

$$\texttt{while} + a; u(X)$$

7.4 De eenheidsoperator en conditionele keuze en repetitie

Merk op: Aaneenschakeling van `while`-instructies leidt tot *niet-regulier* gedrag.

Voorbeeld: $X = \texttt{while} + a; \texttt{while} + b; c$

$$
\begin{aligned}
E_1 &= |X| \\
&= |\texttt{while} + b; X| \trianglelefteq a \trianglerighteq |c| \\
&= E_2 \trianglelefteq a \trianglerighteq |c| \\
E_2 &= |\texttt{while} + a; \texttt{while} + b; X| \trianglelefteq b \trianglerighteq |\texttt{while} + b; c| \\
&= E_3 \trianglelefteq b \trianglerighteq |\texttt{while} + b; c| \\
E_3 &= |\texttt{while} + b; \texttt{while} + a; \texttt{while} + b; X| \trianglelefteq a \trianglerighteq E_1 \\
&\vdots
\end{aligned}
$$

7.5 Macro's, procedures en recursie

PGAm is de uitbreiding van PGA met een *macrodefinitie-omgeving*

$$E_k^m = \{\mathtt{p}\,n = P_n \mid n < k \text{ en } P_n \text{ een PGA-programma}\}$$

en *macro-instructies* $m(\mathtt{p}\,n)$ met interpretatie

$$|m(\mathtt{p}\,n); X| = |P_n; X|.$$

De projectie naar PGAu is voor de hand liggend: vervang iedere macro-instructie $m(\mathtt{p}\,n)$ door $u(P_n)$.

Merk op: Macro-instructies laten niet toe dat andere macro's kunnen worden gebruikt in de definiërende PGA-programma's.

7.5 Macro's, procedures en recursie

PGAp is de uitbreiding van PGA met een *procedurefinitie-omgeving*

$$E_k^p = \{\mathtt{p}\,n = P_n \mid n < k \text{ en } P_n \text{ een PGAp-programma}\}$$

en *procedureaanroepen* $c(\mathtt{p}\,n)$ met interpretatie

$$|c(\mathtt{p}\,n); X| = |P_n; X|.$$

Merk op: In PGAp kan *recursie* worden gemodelleerd en dus niet-regulier gedrag.

7.5 Macro's, procedures en recursie

Voorbeeld: $c(\mathrm{p}\,0);!$ met $\mathrm{p}\,0$ in

$$E_1^p = \{\mathrm{p}\,0 = +a; c(\mathrm{p}\,0); b\}.$$

$$\begin{aligned}
|c(\mathrm{p}\,0);!| &= |+a; c(\mathrm{p}\,0); b;!|,\\
&= |c(\mathrm{p}\,0); b;!| \trianglelefteq a \trianglerighteq |b;!|,\\
|c(\mathrm{p}\,0); b;!| &= |c(\mathrm{p}\,0); b^2;!| \trianglelefteq a \trianglerighteq |b^2;!|,\\
|c(\mathrm{p}\,0); b^2;!| &= |c(\mathrm{p}\,0); b^3;!| \trianglelefteq a \trianglerighteq |b^3;!|,\\
&\vdots\\
|c(\mathrm{p}\,0); b^k;!| &= |c(\mathrm{p}\,0); b^{k+1};!| \trianglelefteq a \trianglerighteq |b^{k+1};!|,\\
&\vdots
\end{aligned}$$