UNIVERSITÀ DEGLI STUDI DI FIRENZE
Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica

Svolgimento esercizi assegnati

MODELLI DI SISTEMI SEQUENZIALI E CONCORRENTI

MARCO BURACCHI

Prof. Rosario Pugliese

Anno Accademico 2015-2016

Marco Buracchi: *MODELLI DI SISTEMI SEQUENZIALI E CONCORRENTI,* Corso di Laurea Magistrale in Informatica, Anno Accademico 2015-2016

INDICE

1 Svolgimento esercizi assegnati 1
- 1.1 Esercizio 2.13 1
- 1.2 Esercizio 3.13 4
- 1.3 Esercizio 4.6 5
  - 1.3.1 punto (a) 5
  - 1.3.2 punto (e) 5
  - 1.3.3 punto (f) 5
  - 1.3.4 punto (j) 5
- 1.4 Esercizio 5.7 7
- 1.5 Esercizio 6.7 8
- 1.6 Esercizio 7.3 9
  - 1.6.1 Sintassi 9
  - 1.6.2 Semantica operazionale 10
  - 1.6.3 Semantica denotazionale 10
  - 1.6.4 Funzione 10
- 1.7 Esercizio 10.2 11
  - 1.7.1 Operatore di ricorsione 11
  - 1.7.2 Costanti di processo 11
  - 1.7.3 Operatore di replicazione 12
- 1.8 Esercizio 11.3 13
  - 1.8.1 Caso Id 13
  - 1.8.2 Caso RUS 14
  - 1.8.3 Caso RS 14
- 1.9 Esercizio 11.8 15
- 1.10 Esercizio 12.3 16

A Svolgimento completo esercizio 5.4 17

ELENCO DELLE FIGURE

Figura 1 Un semplice interruttore 1
Figura 2 Albero di derivazione 4

1

SVOLGIMENTO ESERCIZI ASSEGNATI

1.1 ESERCIZIO 2.13

Formalizzare e dimostrare la validità dell'induzione strutturale *mutua* che consente la dimostrazione simultanea di diverse proprietà per diverse categorie sintattiche.

A volte si ha la necessità di dimostrare congiuntamente un gruppo di enunciati $S1(n), S2(n), \ldots, Sk(n)$ per induzione su n. Un gruppo di enunciati potrebbe essere dimostrato, dimostrando la congiunzione (AND logico) di tutti gli enunciati $(S1(n) \wedge S2(n) \wedge \ldots \wedge Sk(n))$. Tuttavia di solito conviene tenere separati gli enunciati e dimostrare per ciascuno la rispettiva base e il passo induttivo. Questo tipo di dimostrazione è detto *induzione mutua*.

Consideriamo ad esempio un interruttore on/off, rappresentato con il seguente automa:

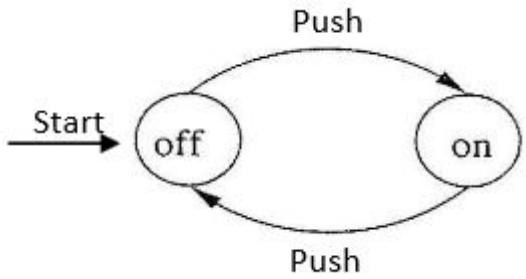


Figura 1.: Un semplice interruttore

Ad ogni pressione del pulsante lo stato cambia tra ON e OFF. Proviamo a dimostrare i seguenti due enunciati:

$S1(n)$: *l'automa si trova nello stato OFF dopo n pressioni* $\Leftrightarrow$ *n è pari.*
$S2(n)$: *l'automa si trova nello stato ON dopo n pressioni* $\Leftrightarrow$ *n è dispari.*

Sapendo che un numero n non può essere allo stesso tempo pari e dispari, si potrebbe supporre che $S1 \Rightarrow S2$, e viceversa. Questo però non è sempre vero in quanto, in generale, un automa potrebbe trovarsi contemporaneamente in più stati. Non è questo il caso dell'automa preso come esempio che si trova sempre esattamente in un solo stato, ma questo deve essere dimostrato come parte dell'induzione mutua.

Proviamo a dimostrare le precedenti proprietà:

BASE: Per il caso base scegliamo $n = 0$. Dato che ci sono due enunciati, ognuno dei quali deve essere dimostrato in entrambe le direzioni (S1 e S2 sono enunciati 'se e solo se'), in effetti ci sono quattro casi per la base e altrettanti per l'induzione:

1. $[S1(0), \Rightarrow]$ Dato che 0 è pari, dobbiamo dimostrare che dopo 0 pressioni l'automa si trova nello stato OFF. Lo stato iniziale dell'automa è proprio OFF e quindi l'automa si trova effettivamente nello stato OFF dopo 0 pressioni.

2. $[S1(0), \Leftarrow]$ L'automa si trova nello stato OFF dopo 0 pressioni, quindi dobbiamo dimostrare che 0 è pari. 0 è pari per definizione quindi non resta altro da dimostrare.

3. $[S2(0), \Rightarrow]$ L'ipotesi afferma che 0 è un numero dispari $\Rightarrow$ l'implicazione è vera.

4. $[S2(0), \Leftarrow]$ L'ipotesi afferma che l'automa si trovi nello stato ON dopo 0 pressioni. Questo è impossibile in quanto all'automa serve almeno una pressione del tasto per arrivare nello stato ON $\Rightarrow$ l'implicazione è vera.

PASSO INDUTTIVO: Supponiamo che $S1(n)$ e $S2(n)$ siano vere, e proviamo a dimostrare $S1(n+1)$ e $S2(n+1)$. Anche questa dimostrazione si divide in 4 parti:

1. $[S1(n+1), \Rightarrow]$ Per ipotesi, $n+1$ è pari. Di conseguenza n è dispari. $S2(n)$ dice che dopo n pressioni l'automa si trova nello stato ON. L'arco da ON a OFF etichettato 'Push' dice che la n+1-esima pressione farà passare l'automa nello stato OFF.

2. $[S1(n+1), \Leftarrow]$ L'ipotesi è che l'automa si trovi nello stato OFF dopo $n+1$ pressioni. Esaminando l'automa vediamo che l'unico modo di pervenire allo stato OFF è di trovarsi nello stato ON e di ricevere

in input il comando 'Push'. Perciò, se l'automa si trova nello stato OFF dopo $n+1$ pressioni, deve essersi trovato nello stato ON dopo n pressioni. Quindi da $[S2(n), \Leftarrow]$ concludiamo che n è dispari. Dunque $n+1$ è pari.

3. $[S2(n+1), \Rightarrow]$ L'ipotesi afferma che $n+1$ è dispari. Di conseguenza n è pari. $[S1(n), \Rightarrow]$ dice che dopo n pressioni l'automa si trova nello stato OFF. L'arco da OFF a ON con etichetta 'Push' dice che la $n+1$-esima pressione farà passare l'automa nello stato ON.

4. $[S2(n+1), \Leftarrow]$ L'ipotesi è che l'automa si trovi nello stato ON dopo $n+1$ pressioni. Esaminando l'automa vediamo che l'unico modo di pervenire allo stato ON è di trovarsi nello stato OFF e di ricevere in input il comando 'Push'. Perciò, se l'automa si trova nello stato ON dopo $n+1$ pressioni, deve essersi trovato nello stato OFF dopo n pressioni. Quindi da $[S1(n), \Leftarrow]$ concludiamo che n è dispari. Dunque $n+1$ è pari.

Da questo esempio possiamo ricavare il modello di tutte le induzioni mutue:

- Ogni enunciato deve essere dimostrato separatamente nella base e nel passo induttivo.
- Se si tratta di enunciati 'se e solo se', allora entrambe le direzioni di ogni enunciato devono essere dimostrate, sia nella base che nel passo induttivo.

1.2 ESERCIZIO 3.13

Fornire l'espressione, derivante dalla sintassi concreta dell'esercizio 11, che ha il seguente albero di derivazione:

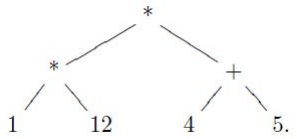


Figura 2.: Albero di derivazione

L'espressione derivante è: $(1 * 12) * (4 + 5)$

1.3 ESERCIZIO 4.6

Dimostrare che la semantica denotazionale delle espressioni regolari soddisfa le seguenti semplici proprietà:

1.3.1 *punto (a)*

$$E + (F + G) \simeq (E + F) + G$$

1.3.2 *punto (e)*

$$E(FG) \simeq (EF)G$$

1.3.3 *punto (f)*

$$E(F + G) \simeq EF + EG$$

1.3.4 *punto (j)*

$$E^* \simeq 1 + E^*E$$

La semantica denotazionale delle espressioni regolari è così definita:

- $\mathcal{L}[\![0]\!] = \emptyset$
- $\mathcal{L}[\![1]\!] = \{\varepsilon\}$
- $\mathcal{L}[\![a]\!] = \{a\}$ (per $a \in A$)
- $\mathcal{L}[\![E + F]\!] = \mathcal{L}[\![E]\!] \cup \mathcal{L}[\![F]\!]$
- $\mathcal{L}[\![E; F]\!] = \mathcal{L}[\![E]\!] \cdot \mathcal{L}[\![F]\!]$
- $\mathcal{L}[\![E^*]\!] = (\mathcal{L}[\![E]\!])^*$

Da queste equivalenze possiamo ricavare:

- punto (a):
$$E + (F + G) \simeq (E + F) + G$$

$$\begin{aligned}\mathcal{L}[\![E + (F + G)]\!] &= \mathcal{L}[\![E]\!] \cup \mathcal{L}[\![F + G]\!] \\ &= \mathcal{L}[\![E]\!] \cup \mathcal{L}[\![F]\!] \cup \mathcal{L}[\![G]\!] \\ &= \mathcal{L}[\![E + F]\!] \cup \mathcal{L}[\![G]\!] \\ &= \mathcal{L}[\![(E + F) + G]\!]\end{aligned}$$

- punto (e):

$$E(FG) \simeq (EF)G$$

$$\begin{aligned}\mathcal{L}[\![E(FG)]\!] &= \mathcal{L}[\![E]\!] \cdot \mathcal{L}[\![FG]\!] \\ &= \mathcal{L}[\![E]\!] \cdot \mathcal{L}[\![F]\!] \cdot \mathcal{L}[\![G]\!] \\ &= \mathcal{L}[\![EF]\!] \cdot \mathcal{L}[\![G]\!] \\ &= \mathcal{L}[\![(EF)G]\!]\end{aligned}$$

- punto (f):

$$E(F + G) \simeq EF + EG$$

$$\begin{aligned}\mathcal{L}[\![E(F + G)]\!] &= \mathcal{L}[\![E]\!] \cdot \mathcal{L}[\![F + G]\!] \\ &= \mathcal{L}[\![E]\!] \cdot (\mathcal{L}[\![F]\!] \cup \mathcal{L}[\![G]\!]) \\ &= (\mathcal{L}[\![E]\!] \cdot \mathcal{L}[\![F]\!]) \cup (\mathcal{L}[\![E]\!] \cdot \mathcal{L}[\![G]\!]) \\ &= \mathcal{L}[\![EF + EG]\!]\end{aligned}$$

1.4 ESERCIZIO 5.7

Siano:

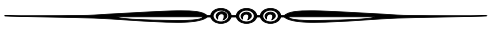
$\mathbf{S} = \lambda xyz.xz(yz)$

$\mathbf{K} = \lambda xy.x$

$\mathbf{I} = \lambda x.x$

Trovare la forma normale dei due termini:

$(\lambda y.yyy)(\mathbf{KI})(\mathbf{SS})$ e $\mathbf{SSSSSSS}$



$$\begin{aligned}(\lambda y.yyy)(\mathbf{KI}(\mathbf{SS})) &= \mathbf{KI}(\mathbf{SS})(\mathbf{KI}(\mathbf{SS}))(\mathbf{KI}(\mathbf{SS})) \\ &= (\lambda y.\mathbf{I})(\mathbf{SS})((\lambda y.\mathbf{I})(\mathbf{SS}))((\lambda y.\mathbf{I})(\mathbf{SS})) \\ &= \mathbf{III} \\ &= \mathbf{II} \\ &= \mathbf{I}\end{aligned}$$

$$\begin{aligned}\mathbf{SSSSSSS} &= (\lambda xyz.xz(yz))\mathbf{SSSSSS} \\ &= (\lambda yz.\mathbf{S}z(yz))\mathbf{SSSSS} \\ &= (\lambda z.\mathbf{S}z(\mathbf{S}z))\mathbf{SSSS} \\ &= \mathbf{SS}(\mathbf{SS})\mathbf{SSS} \\ &= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SS})\mathbf{SSS} \\ &= (\lambda z.\mathbf{S}z(\mathbf{SS})z)\mathbf{SSS} \\ &= \mathbf{SS}(\mathbf{SSS})\mathbf{SS} \\ &= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSS})\mathbf{SS} \\ &= (\lambda z.\mathbf{S}z(\mathbf{SSS}z))\mathbf{SS} \\ &= \mathbf{SS}(\mathbf{SSSS})\mathbf{S} \\ &= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSSS})\mathbf{S} \\ &\;\;\vdots \\ &= \lambda za.aa(aa(aa(\lambda b.bb(bb))))\end{aligned}$$

Lo svolgimento completo si trova in appendice A.

1.5 esercizio 6.7

Risolvere le equazioni fra linguaggi:

1. $X = \{a\} \cdot X$
2. $X = a \cup (\{b\} \cdot X)$

dopo aver scelto gli opportuni domini e verificato che $\cdot$ e $\cup$ sono operazioni continue.

$D : \{a, b\}$	l'insieme composto dai caratteri 'a' e 'b'
$\mathbb{D} = up(D)$	il dominio ottenuto tramite lifting da D
$\mathbb{D}^*$	il dominio sequenza di $\mathbb{D}$

Dato che il membro sinistro è sempre composto da un solo carattere, possiamo definire la concatenazione $\cdot$ come il costruttore $\cdot :: \cdot$ e quindi affermarne la continuità grazie al lemma 6.39. L'unione $\cdot \cup \cdot$ può essere assimilata ad una somma disgiunta di domini considerando la somma disgiunta dei linguaggi.

Le due equazioni generano i linguaggi a^+ e b^*a; sostituendo queste due espressioni nelle relative equazioni otteniamo infatti un'equivalenza. Le soluzioni delle equazioni sono dunque:

1. $X = a^+$
2. $X = b^*a$

1.6 ESERCIZIO 7.3

Introdurre in SLF un tipo di dato *lista di naturali* e scrivere una funzione che, data una lista, ne calcola la lunghezza.

1.6.1 *Sintassi*

Per introdurre il tipo di dati *lista di naturali* abbiamo sicuramente bisogno di aggiungere i nuovi simboli $\{[,]\}$ e le nuove funzioni di base **hd**(l), **tl**(l), **null**(l) e **remove**(n, l) alla sintassi di SLF. Aggiungiamo ai valori di base il tipo di dato *lista di naturali* $l := \{[n, l] \mid n \in \mathbb{NAT}, l \text{ è una lista di naturali.}\}$

Definiamo le funzioni di base che in generale si trovano associate alle liste:

- n **::** l: questo è il costruttore che aggiunge n in testa alla lista l.
- **hd**(l): questa funzione restituisce il primo elemento della lista (senza rimuoverlo).
- **tl**(l): questa funzione restituisce l'ultimo elemento della lista (senza rimuoverlo).
- **null**(l): questa funzione restituisce 0 (*true*) se la lista è vuota o un numero $n + 1$ (*false*) altrimenti.
- **remove**(n, l): questa funzione rimuove la prima occorrenza dell'elemento n dalla lista l e restituisce la lista risultante

Con queste aggiunte, la nuova sintassi di SLF diventa la seguente:

$$
\begin{aligned}
&P ::= \textbf{letrec } D \textbf{ in } T \\
&B ::= n \mid l \\
&T ::= x_i \mid B \mid b_j(T_1, \ldots, T_m) \mid f_r(T_1, \ldots, T_{\rho(r)}) \mid \textbf{if } T \textbf{ then } T1 \textbf{ else } T_2 \\
&D ::= f_1(x_1, \ldots, x_{\rho(1)}) \Leftarrow T_1, \ldots, f_n(x_1, \ldots, x_{\rho(n)}) \Leftarrow T_n \\
&\qquad\qquad\qquad\qquad\qquad\qquad \rho(j) \text{ è l'arietà di } f_j,\ 1 \leqslant j \leqslant n
\end{aligned}
$$

1.6.2 *Semantica operazionale*

Avendo aggiunto un tipo di dato di base, la semantica operazionale deve prevedere che si possano valutare funzioni che abbiano, come argomenti, anche le liste oltre ai naturali. Per questo motivo la regola di inferenza (Bas_1) diventa:

$$\frac{}{b_j(B_1, \ldots, B_m) \rightarrow_D B} \quad b_j(B_1, \ldots, B_m) = B \qquad (Bas_1)$$

1.6.3 *Semantica denotazionale*

Sicuramente la semantica denotazionale ha bisogno di qualche modifica in più, a partire dai tipi delle funzioni semantiche. Dobbiamo considerare che adesso i programmi (risp. i termini) possono restituire liste (possono essere composti da liste) e il dominio $(\mathbb{FUN}_m)^n$ rappresenterà il dominio delle funzioni continue con m argomenti che potranno essere naturali o liste. Per questi motivi i nuovi tipi saranno i seguenti:

$$\begin{aligned}
&\mathcal{P} : \mathit{Prog} \rightarrow (\mathbb{NAT} + \mathbb{LIST}) \\
&\mathcal{D} : \mathit{Decl} \rightarrow (\mathbb{FUN}_m)^n \\
&\mathcal{T} : \mathit{Term} \rightarrow (\mathbb{FUN}_m)^n \rightarrow (\mathbb{NAT} + \mathbb{LIST})^m \rightarrow (\mathbb{NAT} + \mathbb{LIST}) \\
&\mathcal{B} : \mathit{Base} \rightarrow (\mathbb{NAT} + \mathbb{LIST})
\end{aligned}$$

La semantica denotazionale, a questo punto, ha bisogno dell'aggiunta delle funzioni di interpretazione per la nuova classe sintattica B:

$$\begin{aligned}
&\mathcal{B}[\![n]\!] = n \\
&\mathcal{B}[\![l]\!] = l \\
&\mathcal{T}[\![B]\!] = \lambda \overrightarrow{F}.\lambda \overrightarrow{X}.\mathcal{B}[\![B]\!]
\end{aligned}$$

1.6.4 *Funzione*

La funzione richiesta può essere implementata come segue:

$$\mathbf{f}(l) \Leftarrow \mathbf{if}\ \mathbf{null}(l)\ \mathbf{then}\ 0\ \mathbf{else}\ 1 + \mathbf{f}(\mathbf{remove}(\mathbf{hd}(l), l))$$

1.7 ESERCIZIO 10.2

Si scriva un termine che descriva un distributore automatico in grado di offrire acqua o cioccolato un numero illimitato di volte, senza accettare monete fino a che non è stato servito l'utente precedente. Si risolva l'esercizio in tre modi: con l'operatore di ricorsione, con la definizione di costanti di processo e con l'operatore di replicazione.

1.7.1 *Operatore di ricorsione*

Utilizzando l'operatore di ricorsione il termine richiesto può essere così definito:

$$\mathrm{rec}X.\mathrm{coin}.(\overline{\mathrm{choc}}.X + \overline{\mathrm{water}}.X)$$

Si può verificare che le possibili computazioni non accettano monete prima che il prodotto venga effettivamente ritirato. Una delle computazioni possibili, infatti, è la seguente:

$$\mathrm{rec}X.\mathrm{coin}.(\overline{\mathrm{choc}}.X \ + \ \overline{\mathrm{water}}.X) \xrightarrow{\mathrm{coin}}$$

$$\begin{array}{l} \overline{\mathrm{choc}}.\mathrm{rec}X.\mathrm{coin}.(\overline{\mathrm{choc}}.X + \overline{\mathrm{water}}.X) \\ \qquad + \\ \overline{\mathrm{water}}.\mathrm{rec}X.\mathrm{coin}.(\overline{\mathrm{choc}}.X + \overline{\mathrm{water}}.X) \end{array} \quad \left\{ \begin{array}{l} \text{in questo momento} \\ \text{non accetta una nuova} \\ \text{moneta finché non} \\ \text{viene richiesta l'acqua} \\ \text{o il cioccolato} \end{array} \right.$$

1.7.2 *Costanti di processo*

Se vogliamo risolvere il problema utilizzando costanti di processo, la logica resta la stessa. Definendo la costante D come

$$D \triangleq \mathrm{coin}.(\overline{\mathrm{choc}}.D + \overline{\mathrm{water}}.D)$$

il risultato ottenuto è identico. La computazione risultante infatti è :

$coin.(\overline{choc}.D + \overline{water}.D)$ $\xrightarrow{coin}$

$(\overline{choc}.D + \overline{water}.D)$ (come sopra)

1.7.3 *Operatore di replicazione*

Similare è anche la soluzione che prevede l'utilizzo dell'operatore di replicazione. Definiamo D come

$$(\overline{a} \mid !a.coin.(\overline{choc}.\overline{a} + \overline{water}.\overline{a}))\backslash a$$

e vediamo ad esempio una possibile computazione:

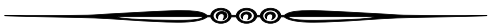
$(\overline{a} \mid !a.coin.(\overline{choc}.\overline{a} + \overline{water}.\overline{a}))\backslash a$ $\xrightarrow{\tau}$

$(coin.(\overline{choc}.\overline{a} + \overline{water}.\overline{a}) \mid !a.coin.(\overline{choc}.\overline{a} + \overline{water}.\overline{a}))\backslash a$ $\xrightarrow{coin}$

$((\overline{choc}.\overline{a} + \overline{water}.\overline{a}) \mid !a.coin.(\overline{choc}.\overline{a} + \overline{water}.\overline{a}))\backslash a$ (come sopra)

1.8 esercizio 11.3

Utilizzando la caratterizzazione delle simulazioni forti vista nell'Esercizio 11.1, si provi che Id è una simulazione e che se R ed S sono simulazioni allora $R \cup S$ ed RS sono simulazioni.



Svolgimento. . . . . .

1.9 esercizio 11.8

Dimostrare che l'unione di tutte le bisimulazioni di branching è una bisimulazione di branching e che essa è un'equivalenza.

Svolgimento. . . . . .

1.10 ESERCIZIO 12.3

Relativamente alla definizione di insieme saturato (Definizione 12.57), si dimostri che se $\mathcal{L}$ è saturato,allora valgono le seguenti proprietà :

1. se $L_1, L_2 \in \mathcal{L}$ allora:
   - $L_1 \cup L_2 \in \mathcal{L}$
   - se $L_1 \subseteq K \subseteq L_2$ allora $K \in \mathcal{L}$
2. $Act(\mathcal{L}) \in \mathcal{L}$

Svolgimento. . . . . .

A

SVOLGIMENTO COMPLETO ESERCIZIO 5.4

$$
\begin{aligned}
\mathbf{SSSSSSS} &= (\lambda xyz.xz(yz))\mathbf{SSSSSS} \\
&= (\lambda yz.\mathbf{S}z(yz))\mathbf{SSSSS} \\
&= (\lambda z.\mathbf{S}z(\mathbf{S}z))\mathbf{SSSS} \\
&= \mathbf{SS}(\mathbf{SS})\mathbf{SSS} \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SS})\mathbf{SSS} \\
&= (\lambda z.\mathbf{S}z(\mathbf{SS})z)\mathbf{SSS} \\
&= \mathbf{SS}(\mathbf{SSS})\mathbf{SS} \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSS})\mathbf{SS} \\
&= (\lambda z.\mathbf{S}z(\mathbf{SSS}z))\mathbf{SS} \\
&= \mathbf{SS}(\mathbf{SSSS})\mathbf{S} \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSSS})\mathbf{S} \\
&= (\lambda z.\mathbf{S}z(\mathbf{SSSS}z))\mathbf{S} \\
&= \mathbf{SS}(\mathbf{SSSSS}) \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSSSS}) \\
&= \lambda z.\mathbf{S}z(\mathbf{SSSSS}z) \\
&= \lambda z.(\lambda ya.a(ya))(\mathbf{SSSSS}z) \\
&= \lambda za.aa(\mathbf{SSSSS}aa) \\
&= \lambda za.aa((\lambda yb.\mathbf{S}b(yb))\mathbf{SSS}zz) \\
&= \lambda za.aa((\lambda b.\mathbf{S}b(\mathbf{S}b))\mathbf{SS}zz) \\
&= \lambda za.aa(\mathbf{SS}(\mathbf{SS})\mathbf{S}aa) \\
&= \lambda za.aa((\lambda yb.\mathbf{S}b(yb))\mathbf{SSS}zz) \\
&= \lambda za.aa((\lambda b.\mathbf{S}b(\mathbf{S}b))\mathbf{SS}zz) \\
&= \lambda za.aa(\mathbf{SS}(\mathbf{SS})\mathbf{S}aa)
\end{aligned}
$$

$$= \lambda z a.aa((\lambda y b.\mathbf{S}b(yb))(\mathbf{SS})\mathbf{S}zz)$$
$$= \lambda z a.aa((\lambda b.\mathbf{S}b(\mathbf{SS}b))\mathbf{S}zz)$$
$$= \lambda z a.aa(\mathbf{SS}(\mathbf{SSS})aa)$$
$$= \lambda z a.aa((\lambda y b.\mathbf{S}b(yb))(\mathbf{SSS})zz)$$
$$= \lambda z a.aa((\lambda b.\mathbf{S}b(\mathbf{SSS}b))zz)$$
$$= \lambda z a.aa(\mathbf{S}a(\mathbf{SSS}a)a)$$
$$= \lambda z a.aa((\lambda y b.bb(yb))(\mathbf{SSS}z)$$
$$= \lambda z a.aa((\lambda b.bb(\mathbf{SSS}bb))z)$$
$$= \lambda z a.aa(aa(\mathbf{SSS}a))$$
$$= \lambda z a.aa(aa((\lambda y b.\mathbf{S}b(yb))\mathbf{S}zz))$$
$$= \lambda z a.aa(aa(\mathbf{S}a(\mathbf{S}a)a))$$
$$= \lambda z a.aa(a((\lambda y b.bb(yb))(\mathbf{S}z)z))$$
$$= \lambda z a.aa(aa((\lambda b.bb(\mathbf{S}bb))z))$$
$$= \lambda z a.aa(aa(aa(\mathbf{S}aa)))$$
$$= \lambda z a.aa(aa(aa((\lambda y b.bb(yb))z)))$$
$$= \lambda z a.aa(aa(aa(\lambda b.bb(bb))))$$