VDM-SL-Beispiele aus dem Editor EFASS

Inhaltsverzeichnis

Der folgende Text gibt Ausschnitte aus der Spezifikation des Editors EFASS wieder, die in der Vorlesung 2002 in Tübingen als Handout ausgegeben wurden. Die vorliegende Version wurde damals manuell generiert und später aus diesem Grund nicht mehr nachgezogen. In diesem Sinn ist dieses Dokument nur von historischem Interesse, da (a) das Laden der Zeilen aus einer Datei einfacher spezifiziert werden kann und (b) spätere Errata nicht eingearbeitet wurden. Die einzigen Veränderungen für das öffentlich Release von „TeachSWT" waren die Umstellung auf ein verbessertes Layout sowie die Korrektur von Rechtschreibfehlern.

1 Übersicht

Die folgenden Abschnitte geben einen Ausschnitt aus der formalen Spezifikation der abstrakten Maschine eines kleinen Editors – EFASS – wieder. Eine umfangreichere Spezifikation und der vollständige Quelltext des Editors werden zu gegebener Zeit von der Webseite zur Vorlesung verfügbar sein.

Der Spezifikationsabschnitt[1] *Sys* beschreibt wichtige Eigenschaften von Zielsystem und -sprache, der Abschnitt *Appchars* im Wesentlichen die Beziehung zwischen Bytes (hier *c-chars* genannt) und den Zeichen, die im Editor manipuliert werden.

Beide Abschnitte sind nur zur Referenz mit abgedruckt. Der für die Vorlesung wichtige Teil der Spezifikation ist die Beschreibung einer Zeile mittels VDM-SL (Spezifikationsteil *Lines*).

2 Systemanbindung

module *sys*

imports

1.0 from *seqtools* all

exports all

functions

2.0 $newline() \; c : c\text{-}char$

.1 post $(c = c)$;

3.0 $space() \; c : c\text{-}char$

.1 post $c \neq newline()$;

4.0 $ord() \; m : c\text{-}char \xleftrightarrow{m} \mathbb{N}$

.1 post $m \neq \{\mapsto\}$;

5.0 $chr() \; m : \mathbb{N} \xleftrightarrow{m} c\text{-}char$

.1 post $m = (ord()^{-1})$;

[1] Auch wenn das entsprechende VDM-SL-Schlüsselwort "*module*" lautet, möchte ich diese Bezeichnung hier zur Vermeidung von Missverständnissen doch vermeiden: Es besteht keine 1:1 Beziehung zwischen den Modulen einer Programmiersprache und den Abschnitten der Spezifikation, die in IFAD VDM-SL als Module bezeichnet werden.

6.0 $C\text{-}CHARS\,()\ v$: token-set

.1 post $v = \text{dom}\ ord\,()$

types

7.0 $c\text{-}char = \text{token}$

.1 inv $c \triangleq c \in (C\text{-}CHARS\,())$

functions

8.0 $PrintableCHARS\,()\ v$: token-set

.1 post $v \subset C\text{-}CHARS\,() \wedge$

.2 $\neg\,((newline\,()) \in v) \wedge$

.3 $(space\,()) \in v \wedge$

.4 $v \neq \{\}$;

9.0 $isPrintable\,(c : c\text{-}char)\ b : \mathbb{B}$

.1 post $b = (c \in (PrintableCHARS\,()))$;

10.0 $isNewline\,(c : c\text{-}char)\ b : \mathbb{B}$

.1 post $b = (c = newline\,())$

types

11.0 $stream = c\text{-}char^*$

values

12.0 $prefix = seqtools`prefix[c\text{-}char]$;

13.0 $suffix = seqtools`suffix[c\text{-}char]$

functions

14.0 $write\,(s : stream, d : c\text{-}char^*)\ s' : stream$

.1 post $s' = s \frown d$;

15.0 $read\,(s : stream, n : \mathbb{N})\ s' : stream, d : c\text{-}char^*$

.1 pre $(\text{len}\ s) \geq n$

.2 post $d \frown s' = s \wedge (\text{len}\ d) = n$;

16.0 $EOF\,()\ s' : stream$

.1 post $s' = []$

end *sys*

3 Zeichen in der Applikation

module *appchars*

imports

17.0 from *seqtools* all ,

18.0 from *sys* all

exports all

definitions
functions

19.0 $encoding\,()\ v : \text{token} \xleftrightarrow{m} sys\text{`}c\text{-}char$

.1 post $(\text{rng}\ v) = sys\text{`}PrintableCHARS\,()$;

20.0 $APPCHARS\,()\ s : \text{token-set}$

.1 post $s = (\text{dom}\ encoding\,())$

types

21.0 $appchar = \text{token}$

.1 inv $a \triangleq a \in (APPCHARS\,())$

functions

22.0 $encode\,(s : appchar^*)\ s' : sys\text{`}c\text{-}char^*$

.1 post $s' = seqtools\text{`}mapped[appchar, sys\text{`}c\text{-}char](encoding\,(), s)$;

23.0 $decode\,(s : sys\text{`}c\text{-}char^*)\ s' : appchar^*$

.1 pre $(\text{elems}\ s) \subseteq (\text{rng}\ encoding\,())$

.2 post $s = encode\,(s')$;

24.0 $space\,()\ c : appchar$

.1 post $encoding\,()\,(c) = sys\text{`}space\,()$

end *appchars*

4 Datentyp “Zeile”

module *lines*

imports

25.0 from *appchars* all ,

26.0 from *seqtools* all ,

27.0 from *sys* all

exports all

definitions

4.1 Darstellung einer Zeile

types

28.0 $line = appchars\text{`}appchar^*$

4.2 Nützliche Funktionen

values

29.0 $prefix = seqtools\text{`}prefix[appchars\text{`}appchar];$

30.0 $suffix = seqtools\text{`}suffix[appchars\text{`}appchar]$

4.3 Konstruktoren/Initialisierung

functions

31.0 $empty : () \to line$
.1 $empty() \triangleq$
.2 $[];$

4.4 Attribute

32.0 $length : line \to \mathbb{N}$
.1 $length(l) \triangleq$
.2 $(\mathsf{len}\ l)$

4.5 Laden und Speichern

values

33.0 $decodeable = \lambda s : sys\text{`}c\text{-}char^* \cdot$
.1 $\mathsf{elems}\ s \subseteq sys\text{`}PrintableCHARS()$

functions

34.0 $completely\text{-}parsed : sys\text{`}stream \times sys\text{`}stream \times sys\text{`}c\text{-}char^* \to \mathbb{B}$
.1 $completely\text{-}parsed(s, s', d) \triangleq$
.2 $\forall c \in (\mathsf{elems}\ d) \cdot$
.3 $\neg\, sys\text{`}isNewline(c) \wedge$
.4 $\mathsf{let}\ n = \mathsf{len}\ d\ \mathsf{in}$
.5 $(s' = sys\text{`}EOF() \wedge sys\text{`}\mathsf{post}\text{-}read(s, n, \mathsf{mk}\text{-}(sys\text{`}EOF(), d)) \vee$
.6 $sys\text{`}\mathsf{post}\text{-}read(s, n, \mathsf{mk}\text{-}(s', d^{\curvearrowright}[sys\text{`}newline()])));$

35.0 $could\text{-}be\text{-}parsed\text{-}from : sys\text{`}stream \times sys\text{`}c\text{-}char^* \to \mathbb{B}$
.1 $could\text{-}be\text{-}parsed\text{-}from(s, d) \triangleq$
.2 $\exists s' : sys\text{`}stream \cdot$
.3 $completely\text{-}parsed(s, s', d);$

36.0 $next\text{-}stream\text{-}section : sys`stream \to sys`c\text{-}char^*$

.1 $next\text{-}stream\text{-}section(s) \triangleq$

.2 $\quad \iota p : sys`c\text{-}char^* \cdot could\text{-}be\text{-}parsed\text{-}from(s,p);$

37.0 $load(s : sys`stream)\ s' : sys`stream, l : line$

.1 pre $decodeable(next\text{-}stream\text{-}section(s))$

.2 post let $d = (appchars`encode(l))$ in

.3 $\quad completely\text{-}parsed(s,s',d)$;

38.0 $save(s : sys`stream, l : line)\ s' : sys`stream$

.1 post $sys`$post-$write(s, s \frown appchars`encode(l) \frown [sys`newline()], s')$

4.6 Konventionen zur Adressierung eines Zeichens

types

39.0 $pos = \mathbb{N}$

functions

40.0 $addresses\text{-}boundary\text{-}in : line \times pos \to \mathbb{B}$

.1 $addresses\text{-}boundary\text{-}in(l,p) \triangleq$

.2 $\quad p \geq 0 \wedge p \leq (\text{len } l);$

41.0 $addresses\text{-}character\text{-}in : line \times pos \to \mathbb{B}$

.1 $addresses\text{-}character\text{-}in(l,p) \triangleq$

.2 $\quad p \geq 1 \wedge p \leq (\text{len } l);$

4.7 Operationen

42.0 $insert\text{-}char\text{-}after(l : line, c : appchars`appchar, p : pos)\ l' : line$

.1 pre $addresses\text{-}boundary\text{-}in(l,p)$

.2 post $prefix(l,p) \frown [c] \frown suffix(l, (\text{len } l) - p) = l'$;

43.0 $remove\text{-}char(l : line, p : pos)\ l' : line$

.1 pre $addresses\text{-}character\text{-}in(l,p)$

.2 post $prefix(l', p-1) \frown [l(p)] \frown suffix(l', p - (\text{len } l)) = l$;

44.0 $split\text{-}after(l : line, p : pos)\ l' : line, l'' : line$

.1 pre $addresses\text{-}boundary\text{-}in(l,p)$

.2 post $l' = prefix(l,p) \wedge$

.3 $\quad l' \frown l'' = l$;

45.0 $merge\text{-}lines(l : line, l' : line)\ l'' : line$

.1 post $l'' = l \frown l'$

end *lines*

A Funktionen auf Folgen

module *seqtools*

exports all

definitions

functions

46.0 $prefix[@item]\,(s:@item^*, n:\mathbb{N})\ p:@item^*\ \triangleq$
.1 if $n = 0$
.2 then $[]$
.3 else $[\mathsf{hd}\ s] \frown prefix[@item]\,((\mathsf{tl}\ s), n-1)$
.4 pre $(\mathsf{len}\ s) \geq n$
.5 post $(\mathsf{len}\ p) = n \wedge$
.6 $\forall i \in \mathsf{inds}\ p \cdot p(i) = s(i)$;

47.0 $suffix[@item]\,(s:@item^*, n:\mathbb{N})\ sf:@item^*\ \triangleq$
.1 if $n = (\mathsf{len}\ s)$
.2 then s
.3 else $suffix[@item]\,((\mathsf{tl}\ s), n-1)$
.4 pre $(\mathsf{len}\ s) \geq n$
.5 post $(\mathsf{len}\ sf) = n \wedge$
.6 $prefix[@item]\,(s, (\mathsf{len}\ s) - n) \frown sf = s$;

48.0 $isPrefix[@item]:@item^* \times @item^* \to \mathbb{B}$
.1 $isPrefix\,(p,s)\ \triangleq$
.2 $prefix[@item]\,(s, (\mathsf{len}\ p)) = p$;

49.0 $isSuffix[@item]:@item^* \times @item^* \to \mathbb{B}$
.1 $isSuffix\,(sf,s)\ \triangleq$
.2 $suffix[@item]\,(s, (\mathsf{len}\ sf)) = sf$;

50.0 $mapped[@item, @res]\,(m:(@item \xrightarrow{m} @res), s:@item^*)\ r:@res^*\ \triangleq$
.1 if $s = []$
.2 then $[]$
.3 else $[m\,(\mathsf{hd}\ s)] \frown (\mathsf{tl}\ s)$
.4 post $(\mathsf{len}\ r) = (\mathsf{len}\ s) \wedge$
.5 $\forall i \in \mathsf{inds}\ s \cdot r(i) = m(s(i))$;

51.0 $addresses\text{-}an\text{-}item[@item]:\mathbb{N} \times @item^* \to \mathbb{B}$
.1 $addresses\text{-}an\text{-}item\,(n,s)\ \triangleq$
.2 $(n \in (\mathsf{inds}\ s))$

end *seqtools*