Note: [C] is a list of Chars, which could also be a String.

**ADT for Text Editor**

**NAME**

Text Editor – an ADT representing a simple one-line text editor

**SETS**

T the set of Text Editors { [C], [C], [C], [C]}

C the set of characters { a..z U A..Z U 0..9 U.. …”..’..!..?..(..)..[..] }

**SYNTAX**

Create: ꓕ → T

Destroy T → ꓕ

Cut T → T

Paste T → T

Backspace T → T

BackspaceForward T → T

Copy T → T

Write [C] x T → T

HighlightCharLeft T → T

HighlightCharRight T → T

HighlightWordLeft T → T

HighlightWordRight T → T

HighlightAllLeft T → T

HighlightAllRight T → T

HighlightAll T → T

ClearHighlighted T → T

MoveCursorLeft T → T

MoveCursorRight T → T

MoveCursorWordStart T → T

MoveCursorWordEnd T → T

MoveCursorStart T → T

MoveCursorEnd T → T

MoveCursorStart T → T

MoveCursorEnd T → T

Save T x [C] → IO

Load IO → T

**SEMANTICS**

Ɐ l.l ∈ [C], Ɐ h.h ∈ [C], Ɐ r.r ∈ [C], Ɐ c.c ∈ [C]

pre-create() :: true

post-create( ; r ) :: r = ([], [], [], [])

pre-Destroy( t ) :: true

post-Destroy( (l , h, r, c) ; r ) :: r = ꓕ

pre-Cut( t ) :: true

post-Cut( (l, h, r, c) ; r) :: r = (l , [] , r, h)

pre-Paste( t ) :: true

post-Paste( (l, h, r, c) ; r) :: r = ((l+c), h , r, c)

pre-backspace ( t ) :: true

post-backspace ( (l , h, r, c) ; r ) :: r = ((init l) , h , r, c)

pre-backspaceForward ( t ) :: true

post-backspaceForward ( (l, h, r, c) ; r) :: r = (l , h , (tail r), c)

pre-copy ( t ) :: true

post-copy ( (l , h, r, c) ; r ) :: r = (l , h , r, h)

pre-write ( x, t ) :: true

post-write ( x, (l, h, r, c) ; r) :: r = ((l++x) , h , r, c)

pre-highlightCharLeft ( t ) :: true

post-highlightCharLeft ( (l , h, r, c) ; r ) :: r = (init l , ([last l]++h) , r, c)

pre-highlightCharRight ( t ) :: true

post-highlightCharRight ( (l, h, r, c) ; r) :: r = (l , (h++[head r]) , (tail r), c)

pre-highlightWordLeft ( t ) :: true

post-highlightWordLeft ( (l , h, r, c) ; r ) :: r = selectLeft ((init l), ([last l]++h), r, c)

where

selectLeft ([], h, r, c) = ([], h, r, c)

selectLeft (l, h, r, c) = if (last l) = ‘ ‘

then (l, h, r, c)

else selectLeft ((init l), ([last l]++h), r, c)

pre-highlightWordRight ( t ) :: true

post-highlightWordRight ( (l, h, r, c) ; r) :: r = selectRight (l, (h++ (head r)), (tail r), c)

where

selectRight (l, h, [], c) = (l, h, [], c)

selectRight (l, h, r, c) = if (head r) = ‘ ‘

then (l, h, r, c)

else selectRight (l, (h++ (head r)), (tail r), c)

pre-highlightAllLeft ( t ) :: true

post-highlightAllLeft ( (l , h, r, c) ; r ) :: r = ([] , (l++h), r, c)

pre-highlightAllRight ( t ) :: true

post-highlightAllRight ( (l, h, r, c) ; r) :: r = (l , (h++r) , [], c)

pre-highlightAll ( t ) :: true

post-highlightAll ( (l , h, r, c) ; r ) :: r = ([] , (l++h++r) , [], c)

pre-clearHighlighted ( t ) :: true

post-clearHighlighted ( (l, h, r, c) ; r) :: r = ((l++h) , [] , r, c)

pre-moveCursorLeft ( t ) :: true

post-moveCursorLeft ( (l, h, r, c) ; r) :: r = ((init l) , [] , ([last l]++h++r), c)

pre-moveCursorRight ( t ) :: true

post-moveCursorRight ( (l , h, r, c) ; r ) :: r = ((l++h++[head r]), [], (tail r), c)

pre-moveCursorWordStart ( t ) :: true

post-moveCursorWordStart ( (l, h, r, c) ; r) :: r = moveToStart ((init l), [], ([last l]++r), c))

where

moveToStart ([], h, r, c) = ([], h, r, c)

moveToStart (l, h, r, c) = if (last l) = ‘ ‘

then (l, h, r, c)

else moveToStart ((init l), [], ([last l]++r), c))

pre-moveCursorWordEnd ( t ) :: true

post-moveCursorWordEnd ( (l , h, r, c) ; r ) :: r = moveToEnd (l++[head r]), [], (tail r), c)

where

moveToEnd (l, h, [], c) = (l, h, [], c)

moveToEnd (l, h, r, c) = if (head r) = ‘ ‘

then (l, h, r, c)

else moveToEnd (l++[head r]), [], (tail r), c)

pre-moveCursorStart ( t ) :: true

post-moveCursorStart ( (l, h, r, c) ; r) :: r = ((l++h++r), [], [], c)

pre-moveCursorEnd ( t ) :: true

post-moveCursorEnd ( (l , h, r, c) ; r ) :: r = ([], [], (l++h++r), c)

pre-save ( t, s ) :: true

post-save ( (l, h, r, c), s ; r) :: r = writeFile s (l++h++r)

pre-load ( s ) :: true

post-load (s ; r) :: r = (loadStringFromFile s , [] , [], [])

**AUTHOR**

Lawrence Taylor(16601608@students.lincoln.ac.uk). April 2018