Implementation of Deletion Algorithms on Lists and Binary Trees in *Theorema*

Apr 2020

Isabela Dramnesc, Tudor Jebelean

Technical Report 20-04
RISC-Linz, JKU

Abstract:

We demonstrate the implementation of various deletion algorithms on lists and binary trees in the *Theorema* system.

References:

[1] W. Windsteiger. *Theorema 2.0: A System for Mathematical Theory Exploration*. ICMS'2014, LNCS 8592, pp. 49-52.

[2] B. Buchberger, T. Jebelean, T. Kutsia, A. Maletzky, W. Windsteiger. *Theorema 2. Computer-Assisted Natural-Style Mathematics*. Journal of Formalized Reasoning, vol. 9 (2016), no. 1, pp. 149-185.

[3] www.risc.jku.at/research/theorema/software/

Lists

Lists are formed using the constructors "⟨⟩" (empty list) and "⌣" ("**a⌣U**" represents the list having "**a**" as the first element).

Basic functions: concatenation, append, head, Tail, Front, last, isEmpty

These basic functions constitute the underlying computing environment for the algorithms on lists. Some of them are implemented directly in Mathematica for increased efficiency. The symbol "⩓" prefixes Mathematica commands which are executed by the Theorema system.

Algorithm (BasicList)

Concatenation of two lists

In[10]:= $\forall_{U} ((\langle\rangle \asymp U) == U)$ (Conc – SymL – 1)

In[13]:= $\forall_{U} ((U \asymp \langle\rangle) == U)$ (ConcSymL – 2)

In[14]:= $\forall_{a,U,b,V} ((a \smile U \asymp b \smile V) == ⩓\text{ReplaceAll}[a \smile U, ⩓\text{Rule}[\langle\rangle, b \smile V]])$ (ConcSymL – 3)

In[15]:= (4)

Appending an element at the end of a list

In[16]:= $\forall_{U,a} (U \frown a == ⩓\text{ReplaceAll}[U, ⩓\text{Rule}[\langle\rangle, (a \smile \langle\rangle)]])$ (AppSymL)

The first element of a list

In[17]:= $\forall_{U,a} (\text{head}[a \smile U] == a)$ (head)

The rest of the elements, without the first one

In[18]:= $\forall_{U,a} (\text{Tail}[a \smile U] == U)$ (Tail)

The first elements of a list without the last one, does not apply to the empty list

In[19]:= $\forall_{U,a} (\text{Front}[a \smile U] == ⩓\text{ReplaceAll}[a \smile U, ⩓\text{Rule}[(⩓\text{Blank}[] \smile \langle\rangle), \langle\rangle]])$ (Front)

The last element of a list, does not apply to the empty list

In[20]:= $\forall_{U,a} (\text{last}[a \smile U] == ⩓\text{First}[⩓\text{First}[⩓\text{Cases}[a \smile U, (⩓\text{Blank}[] \smile \langle\rangle), ⩓\text{Infinity}]]])$ (Last)

Testing emptiness

```
In[21]:=  isEmpty[⟨⟩] == True                    (isEmptyTrue)
```

```
In[22]:=  ( ∀_{a,U} isEmpty[a ⌣ U] == False )     (isEmptyFalse)
```

Computation

```
In[76]:=  ⟨⟩ ≍ (1 ⌣ (2 ⌣ (5 ⌣ (4 ⌣ ⟨⟩))))
Out[76]=  1 ⌣ (2 ⌣ (5 ⌣ (4 ⌣ ⟨⟩)))

In[77]:=  (1 ⌣ (2 ⌣ (5 ⌣ (4 ⌣ ⟨⟩)))) ≍ ⟨⟩
Out[77]=  1 ⌣ (2 ⌣ (5 ⌣ (4 ⌣ ⟨⟩)))

In[70]:=  (1 ⌣ ⟨⟩) ≍ (1 ⌣ (2 ⌣ (5 ⌣ (4 ⌣ ⟨⟩))))
Out[70]=  1 ⌣ (1 ⌣ (2 ⌣ (5 ⌣ (4 ⌣ ⟨⟩))))

In[78]:=  (3 ⌣ (4 ⌣ ⟨⟩)) ≍ (1 ⌣ (2 ⌣ (3 ⌣ (4 ⌣ ⟨⟩))))
Out[78]=  3 ⌣ (4 ⌣ (1 ⌣ (2 ⌣ (3 ⌣ (4 ⌣ ⟨⟩)))))

In[79]:=  ⟨⟩ ⌢ 1
Out[79]=  1 ⌣ ⟨⟩

In[80]:=  1 ⌣ (2 ⌣ (3 ⌣ (4 ⌣ ⟨⟩))) ⌢ 1
Out[80]=  1 ⌣ (2 ⌣ (3 ⌣ (4 ⌣ (1 ⌣ ⟨⟩))))

In[81]:=  Front[⟨⟩]
Out[81]=  Front[⟨⟩]

In[82]:=  Front[(1 ⌣ (2 ⌣ (3 ⌣ (4 ⌣ ⟨⟩))))]
Out[82]=  1 ⌣ (2 ⌣ (3 ⌣ ⟨⟩))

In[83]:=  last[⟨⟩]
Out[83]=  last[⟨⟩]
```

In[84]:=
```
last[(1 ⌣ (2 ⌣ (3 ⌣ (4 ⌣ ⟨⟩))))]
```
Out[84]=
```
4
```
In[85]:=
```
isEmpty[⟨⟩]
```
Out[85]=
```
True
```
In[86]:=
```
isEmpty[(3 ⌣ (4 ⌣ ⟨⟩))]
```
Out[86]=
```
False
```

DelFOL: Delete the first occurrence of an element from a list

Pattern matching algorithm

ALGORITHM (DELFOL)

In[23]:=
$$\forall_{a} \left(\text{DelFOL}[a, \langle\rangle] = \langle\rangle\right) \qquad (\text{DelFOL} - 0)$$

In[24]:=
$$\forall_{a,b,U} \left(\text{DelFOL}[a, b \smile U] = \begin{cases} U & \Leftarrow\ a == b \\ b \smile \text{DelFOL}[a, U] & \Leftarrow\ a \neq b \end{cases}\right) \qquad (\text{DelFOL} - 1)$$

Computation

In[87]:=
```
DelFOL[3,  ⟨⟩]
```
Out[87]=
```
⟨⟩
```
In[88]:=
```
DelFOL[5,  1 ⌣ (3 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]
```
Out[88]=
```
1 ⌣ (3 ⌣ (6 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))
```
In[89]:=
```
DelFOL[20,  1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]
```
Out[89]=
```
1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))
```
In[90]:=
```
DelFOL[2,  1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]
```
Out[90]=
```
1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))
```

Functional algorithm

ALGORITHM (DELFOLF)

In[25]:=

$$\underset{a,U}{\forall}\left(\text{DelFOLf}[a, U] == \left(\begin{cases} \langle\rangle & \Leftarrow \text{isEmpty}[U] \\ \text{Tail}[U] & \Leftarrow a == \text{head}[U] \\ \text{head}[U] \smile \text{DelFOLf}[a, \text{Tail}[U]] & \Leftarrow \text{True} \end{cases}\right)\right)$$

(DelFOLf)

Computation

In[91]:= `DelFOLf[3, ⟨⟩]`

Out[91]= `⟨⟩`

In[92]:= `DelFOLf[5, 1 ⌣ (3 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]`

Out[92]= `1 ⌣ (3 ⌣ (6 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))`

In[94]:= `DelFOLf[2, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]`

Out[94]= `1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))`

In[95]:= `DelFOLf[20, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]`

Out[95]= `1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))`

Tail recursive algorithm

ALGORITHM (DELFOLTR)

In[26]:=

$$\underset{a,U}{\forall}\left(\text{DelFOL}[a, U] == \text{DelFOLtr}[U, a, \langle\rangle]\right)$$

(DelFolNewTR)

In[27]:=

$$\underset{a,V}{\forall}\left(\text{DelFOLtr}[\langle\rangle, a, V] == V\right)$$

(DelFOLtr – 0)

In[28]:=

$$\underset{a,b,U,V}{\forall}\left(\text{DelFOLtr}[b \smile U, a, V] == \left(\begin{cases} V \asymp U & \Leftarrow a == b \\ \text{DelFOLtr}[U, a, V \frown b] & \Leftarrow a \neq b \end{cases}\right)\right)$$

(DelFOLtr – 1)

Computation

In[96]:= `DelFOL[3, ⟨⟩]`

Out[96]= `⟨⟩`

```
In[97]:= DelFOL[5, 1 ⌣ (3 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))))]
Out[97]= 1 ⌣ (3 ⌣ (6 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))

In[98]:= DelFOL[20, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))))]
Out[98]= 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))

In[99]:= DelFOL[2, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))))]
Out[99]= 1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))

In[100]:= DelFOLtr[⟨⟩, 3, ⟨⟩]
Out[100]= ⟨⟩

In[101]:= DelFOLtr[(5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))), 3, ⟨⟩]
Out[101]= 5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))

In[102]:= DelFOLtr[1 ⌣ (3 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))), 5, ⟨⟩]
Out[102]= 1 ⌣ (3 ⌣ (6 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))

In[103]:= DelFOLtr[1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))), 20, ⟨⟩]
Out[103]= 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))

In[104]:= DelFOLtr[1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))), 2, ⟨⟩]
Out[104]= 1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))
```

Functional tail recursive algorithm

ALGORITHM (DELFOLTRF)

```
In[29]:= ( ∀_{a,U} (DelFOL[a, U] == DelFOLtrf[U, a, ⟨⟩]) )        (DelFOLnewTRF)

In[30]:= ( ∀_{a,U,V} (DelFOLtrf[U, a, V] ==
              V                                  ⇐ isEmpty[U]
              V ≍ Tail[U]                        ⇐ (a == head[U])
              DelFOLtrf[Tail[U], a, V ⌢ head[U]] ⇐ (True) ) )   (DelFOLtrf)
```

Computation

```
In[106]:= DelFOL[3, ⟨⟩]

Out[106]= ⟨⟩

In[107]:= DelFOL[5, 1⌣(3⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))]

Out[107]= 1⌣(3⌣(6⌣(4⌣(2⌣⟨⟩))))

In[108]:= DelFOL[2, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))]

Out[108]= 1⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))

In[109]:= DelFOL[20, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))]

Out[109]= 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))

In[110]:= DelFOL[2, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))]

Out[110]= 1⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))

In[111]:= DelFOLtrf[⟨⟩, 3, ⟨⟩]

Out[111]= ⟨⟩

In[112]:= DelFOLtrf[(5⌣(4⌣(2⌣⟨⟩))), 3, ⟨⟩]

Out[112]= 5⌣(4⌣(2⌣⟨⟩))

In[113]:= DelFOLtrf[1⌣(3⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))), 5, ⟨⟩]

Out[113]= 1⌣(3⌣(6⌣(4⌣(2⌣⟨⟩))))

In[114]:= DelFOLtrf[1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))), 20, ⟨⟩]

Out[114]= 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))

In[115]:= DelFOLtrf[1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))), 2, ⟨⟩]

Out[115]= 1⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))
```

DelAOL: Delete all occurrences of an element in a list

Pattern matching algorithm

ALGORITHM (DELAOL)

In[31]:=
$$\forall_{a} \left(\mathrm{DelAOL}[a, \langle\rangle] = \langle\rangle\right) \qquad (DelAOL-0)$$

In[32]:=
$$\forall_{\substack{a,b,U\\ a=b}} \left(\mathrm{DelAOL}[a, b \smile U] = \mathrm{DelAOL}[a, U]\right) \qquad (DelAOL-1)$$

In[33]:=
$$\forall_{\substack{a,b,U\\ a\neq b}} \left(\mathrm{DelAOL}[a, b \smile U] = b \smile \mathrm{DelAOL}[a, U]\right) \qquad (DelAOL-2)$$

Computation

In[116]:= DelAOL[3, ⟨⟩]

Out[116]= ⟨⟩

In[117]:= DelAOL[5, 1 ⌣ (5 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (5 ⌣ ⟨⟩)))))]

Out[117]= 1 ⌣ (6 ⌣ (4 ⌣ ⟨⟩))

In[118]:= DelAOL[2, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]

Out[118]= 1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ ⟨⟩)))

In[119]:= DelAOL[20, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]

Out[119]= 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))

Functional algorithm

ALGORITHM (DELAOLF)

In[34]:=
$$\forall_{a,U} \left(\mathrm{DelAOLf}[a, U] = \left(\begin{cases} \langle\rangle & \Leftarrow \mathrm{isEmpty}[U] \\ \mathrm{DelAOLf}[a, \mathrm{Tail}[U]] & \Leftarrow a = \mathrm{head}[U] \\ \mathrm{head}[U] \smile \mathrm{DelAOLf}[a, \mathrm{Tail}[U]] & \Leftarrow \mathrm{True} \end{cases}\right)\right) \qquad (DelAOLf)$$

Computation

In[120]:=
```
DelAOLf[3, ⟨⟩]
```
Out[120]=
```
⟨⟩
```
In[121]:=
```
DelAOLf[5, 1⌣(5⌣(6⌣(5⌣(4⌣(5⌣⟨⟩))))))]
```
Out[121]=
```
1⌣(6⌣(4⌣⟨⟩))
```
In[122]:=
```
DelAOLf[2, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))))]
```
Out[122]=
```
1⌣(6⌣(5⌣(4⌣⟨⟩)))
```
In[123]:=
```
DelAOLf[20, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))))]
```
Out[123]=
```
1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))
```

Tail recursive algorithm

Algorithm (DelAOLtr)

In[35]:= $\forall_{a,U}$ DelAOL[a, U] == DelAOLtr[U, a, ⟨⟩] (DelAOLnew)

In[36]:= $\forall_{a,V}$ DelAOLtr[⟨⟩, a, V] == V (DelAOLtr − 0)

In[37]:= $\forall_{a,b,U,V \atop a==b}$ DelAOLtr[b⌣U, a, V] == DelAOLtr[U, a, V] (DelAOLtr − 1)

In[38]:= $\forall_{a,b,U,V \atop a\neq b}$ DelAOLtr[b⌣U, a, V] == DelAOLtr[U, a, V⌢b] (DelAOLtr − 2)

Computation

In[124]:=
```
DelAOL[3, ⟨⟩]
```
Out[124]=
```
⟨⟩
```
In[125]:=
```
DelAOL[5, 1⌣(5⌣(6⌣(5⌣(4⌣(5⌣⟨⟩))))))]
```
Out[125]=
```
1⌣(6⌣(4⌣⟨⟩))
```

```
In[126]:= DelAOL[2, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]
Out[126]= 1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ ⟨⟩)))

In[127]:= DelAOL[20, 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))]
Out[127]= 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))

In[128]:= DelAOLtr[⟨⟩, 3, ⟨⟩]
Out[128]= ⟨⟩

In[129]:= DelAOLtr[(5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))), 3, ⟨⟩]
Out[129]= 5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))

In[130]:= DelAOLtr[1 ⌣ (3 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))), 5, ⟨⟩]
Out[130]= 1 ⌣ (3 ⌣ (6 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))

In[131]:= DelAOLtr[1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))), 20, ⟨⟩]
Out[131]= 1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩)))))

In[132]:= DelAOLtr[1 ⌣ (2 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ (2 ⌣ ⟨⟩))))), 2, ⟨⟩]
Out[132]= 1 ⌣ (6 ⌣ (5 ⌣ (4 ⌣ ⟨⟩)))
```

Functional tail recursive algorithm

ALGORITHM (DELAOLTRF)

```
In[39]:= ( ∀_{a,U} (DelAOL[a, U] == DelAOLtrf[U, a, ⟨⟩]) )        (DelAOLnewTRF)

In[40]:= ( ∀_{a,U,V} (DelAOLtrf[U, a, V] ==
             {                                  V   ⇐  isEmpty[U]
               DelAOLtrf[Tail[U], a, V]             ⇐  (a == head[U])
               DelAOLtrf[Tail[U], a, V ⌢ head[U]]   ⇐  (True)          )) )   (DelAOLtrf)
```

Computation

In[142]:=
```
DelAOL[3, ⟨⟩]
```
Out[142]=
```
⟨⟩
```
In[143]:=
```
DelAOL[5, 1⌣(5⌣(6⌣(5⌣(4⌣(5⌣⟨⟩))))))]
```
Out[143]=
```
1⌣(6⌣(4⌣⟨⟩))
```
In[144]:=
```
DelAOL[2, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))))]
```
Out[144]=
```
1⌣(6⌣(5⌣(4⌣⟨⟩)))
```
In[145]:=
```
DelAOL[20, 1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))))]
```
Out[145]=
```
1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))
```
In[146]:=
```
DelAOLtrf[⟨⟩, 3, ⟨⟩]
```
Out[146]=
```
⟨⟩
```
In[147]:=
```
DelAOLtrf[(5⌣(4⌣(2⌣⟨⟩))), 3, ⟨⟩]
```
Out[147]=
```
5⌣(4⌣(2⌣⟨⟩))
```
In[148]:=
```
DelAOLtrf[1⌣(3⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))), 5, ⟨⟩]
```
Out[148]=
```
1⌣(3⌣(6⌣(4⌣(2⌣⟨⟩))))
```
In[149]:=
```
DelAOLtrf[1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))), 20, ⟨⟩]
```
Out[149]=
```
1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩)))))
```
In[150]:=
```
DelAOLtrf[1⌣(2⌣(6⌣(5⌣(4⌣(2⌣⟨⟩))))), 2, ⟨⟩]
```
Out[150]=
```
1⌣(6⌣(5⌣(4⌣⟨⟩)))
```

Trees

Trees are formed using the constructors "**ε**" (empty tree) and "**⟨...⟩**" ("**⟨L,a,R⟩**" represents the tree having "**a**" as the root, "**L**" as the left subtree, and "**R**" as the right subtree).

Basic functions: root, Left, Right, ConcT, isEmpty, occurs

These basic functions constitute the underlying computing environment for the algorithms on trees. Concatenation and emptiness test are overloaded - they have the same name as the corresponding functions for lists, but this is not a problem because the shape difference between lists and trees.

Algorithm (BasicTree)

Structural decomposition

In[41]:= $\forall_{a,L,R} (\text{Left}[\langle L, a, R\rangle] == L)$ (Left)

In[42]:= $\forall_{a,L,R} (\text{root}[\langle L, a, R\rangle] == a)$ (root)

In[43]:= $\forall_{a,L,R} (\text{Right}[\langle L, a, R\rangle] == R)$ (Right)

Concatenation of trees

In[44]:= $\forall_{S} ((\varepsilon \asymp S) == S)$ (ConcT - 0)

In[45]:= $\forall_{a,L,R,S} ((\langle L, a, R\rangle \asymp S) == \langle L, a, R \asymp S\rangle)$ (ConcT - 1)

Emptiness test

In[46]:= isEmpty[ε] == True (isEmptyTTrue)

In[47]:= $\forall_{a,L,R} (\text{isEmpty}[\langle L, a, R\rangle] == \text{False})$ (isEmptyTFalse)

Occurrence test

In[48]:= $\forall_{a} ((a \lhd \varepsilon) == \text{False})$ (occurs - 0)

In[49]:= $\forall_{a,b,L,R} \left((a \lhd \langle L, b, R\rangle) == \left(\vee \begin{cases} a \lhd L \\ a == b \\ a \lhd R \end{cases}\right)\right)$ (occurs - 1)

Computation

```
In[151]:= Left[ε]
Out[151]= Left[ε]

In[152]:= root[ε]
Out[152]= root[ε]

In[153]:= Right[ε]
Out[153]= Right[ε]

In[154]:= Left[⟨ε, 5, ⟨ε, 8, ε⟩⟩]
Out[154]= ε

In[155]:= root[⟨ε, 5, ⟨ε, 8, ε⟩⟩]
Out[155]= 5

In[156]:= Right[⟨ε, 5, ⟨ε, 8, ε⟩⟩]
Out[156]= ⟨ε, 8, ε⟩

In[157]:= Left[⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩]
Out[157]= ⟨ε, 4, ε⟩

In[158]:= root[⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩]
Out[158]= 5

In[159]:= Right[⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩]
Out[159]= ⟨ε, 8, ε⟩

In[160]:= ε ≍ ⟨ε, 5, ⟨ε, 8, ε⟩⟩
Out[160]= ⟨ε, 5, ⟨ε, 8, ε⟩⟩

In[161]:= ⟨ε, 5, ⟨ε, 8, ε⟩⟩ ≍ ε
Out[161]= ⟨ε, 5, ⟨ε, 8, ε⟩⟩

In[162]:= ⟨ε, 1, ⟨ε, 3, ε⟩⟩ ≍ ⟨ε, 5, ⟨ε, 8, ε⟩⟩
Out[162]= ⟨ε, 1, ⟨ε, 3, ⟨ε, 5, ⟨ε, 8, ε⟩⟩⟩⟩
```

In[163]:=
```
isEmpty[ε]
```
Out[163]=
```
True
```
In[164]:=
```
isEmpty[⟨ε, 8, ε⟩]
```
Out[164]=
```
False
```
In[165]:=
```
1 ⊲ ε
```
Out[165]=
```
False
```
In[166]:=
```
2 ⊲ ⟨ε, 1, ⟨ε, 3, ⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩⟩⟩
```
Out[166]=
```
False
```
In[167]:=
```
4 ⊲ ⟨ε, 1, ⟨ε, 3, ⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩⟩⟩
```
Out[167]=
```
True
```
In[168]:=
```
5 ⊲ ⟨ε, 1, ⟨ε, 3, ⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩⟩⟩
```
Out[168]=
```
True
```
In[169]:=
```
8 ⊲ ⟨ε, 1, ⟨ε, 3, ⟨⟨ε, 4, ε⟩, 5, ⟨ε, 8, ε⟩⟩⟩⟩
```
Out[169]=
```
True
```

DelFOST: Delete the first occurrence of an element from a sorted tree

Pattern matching algorithm

ALGORITHM (DELFOST)

In[50]:=
```
(∀_a (DelFOST[a, ε] == ε))
```
(DelFOST – 0)

In[51]:=
```
(∀_{a,b,L,R; a<b} (DelFOST[a, ⟨L, b, R⟩] == ⟨DelFOST[a, L], b, R⟩))
```
(DelFOST – 1)

In[52]:=
```
(∀_{a,b,L,R; a==b} (DelFOST[a, ⟨L, b, R⟩] == (L ≍ R)))
```
(DelFOST – 2)

In[53]:=
```
(∀_{a,b,L,R; a>b} (DelFOST[a, ⟨L, b, R⟩] == ⟨L, b, DelFOST[a, R]⟩))
```
(DelFOST – 3)

Computation

```
In[170]:= DelFOST[3, ε]
Out[170]= ε

In[171]:= DelFOST[3, ⟨ε, 5, ε⟩]
Out[171]= ⟨ε, 5, ε⟩

In[172]:= DelFOST[5, ⟨ε, 5, ε⟩]
Out[172]= ε

In[173]:= DelFOST[10, ⟨ε, 5, ε⟩]
Out[173]= ⟨ε, 5, ε⟩

In[174]:= DelFOST[3, ⟨⟨ε, 3, ε⟩, 5, ε⟩]
Out[174]= ⟨ε, 5, ε⟩

In[175]:= DelFOST[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
Out[175]= ⟨ε, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩

In[176]:= DelFOST[7, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
Out[176]= ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 10, ε⟩⟩

In[177]:= DelFOST[10, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
Out[177]= ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ε⟩⟩

In[178]:= DelFOST[5, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
Out[178]= ⟨ε, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩
```

.

Functional algorithm

ALGORITHM (DELFOSTF)

In[54]:=

```
∀    (DelFOSTf[a, T] ==
a,T
      ε                          ⇐ isEmpty[T]
      ⟨DelFOSTf[a, Left[T]],     ⇐ a < root[T]
       root[T], Right[T]⟩
      Left[T] ≍ Right[T]         ⇐ a == root[T]
      ⟨Left[T], root[T],         ⇐ a > root[T]
       DelFOSTf[a, Right[T]]⟩ ))
```

(DelFOSTf)

Computation

In[206]:= `DelFOSTf[3, ε]`

Out[206]= ε

In[207]:= `DelFOSTf[3, ⟨ε, 5, ε⟩]`

Out[207]= ⟨ε, 5, ε⟩

In[208]:= `DelFOSTf[5, ⟨ε, 5, ε⟩]`

Out[208]= ε

In[209]:= `DelFOSTf[10, ⟨ε, 5, ε⟩]`

Out[209]= ⟨ε, 5, ε⟩

In[210]:= `DelFOSTf[3, ⟨⟨ε, 3, ε⟩, 5, ε⟩]`

Out[210]= ⟨ε, 5, ε⟩

In[211]:= `DelFOSTf[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]`

Out[211]= ⟨ε, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩

In[212]:= `DelFOSTf[7, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]`

Out[212]= ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 10, ε⟩⟩

In[213]:=
```
DelFOSTf[10, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
```
Out[213]=
```
⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ε⟩⟩
```
In[214]:=
```
DelFOSTf[5, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
```
Out[214]=
```
⟨ε, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩
```

DelFOT: Delete the first occurrence of an element from an unsorted tree

Pattern matching algorithm

Algorithm (DelFOT)

In[55]:= $\left(\forall_{a} \left(\text{DelFOT}[a, \varepsilon] == \varepsilon\right)\right)$ (DelFOT − 0)

In[56]:= $\left(\forall_{\substack{a,b,L,R \\ a==b}} \left(\text{DelFOT}[a, \langle L, b, R\rangle] == (L \asymp R)\right)\right)$ (DelFOT − 1)

In[57]:= $\left(\forall_{\substack{a,b,L,R \\ (a\neq b \wedge a \triangleleft L)}} \left(\text{DelFOT}[a, \langle L, b, R\rangle] == \langle \text{DelFOT}[a, L], b, R\rangle\right)\right)$ (DelFOT − 2)

In[58]:= $\left(\forall_{\substack{a,b,L,R \\ (a\neq b \wedge a \triangleleft R)}} \left(\text{DelFOT}[a, \langle L, b, R\rangle] == \langle L, b, \text{DelFOT}[a, R]\rangle\right)\right)$ (DelFOT − 3)

In[59]:= $\left(\forall_{\substack{a,b,L,R \\ ((a\neq b) \wedge (\neg (a \triangleleft L)) \wedge (\neg (a \triangleleft R)))}} \left(\text{DelFOT}[a, \langle L, b, R\rangle] == \langle L, b, R\rangle\right)\right)$ (DelFOT − 4)

Computation

In[215]:=
```
DelFOT[3, ε]
```
Out[215]=
```
ε
```
In[216]:=
```
DelFOT[3, ⟨ε, 5, ε⟩]
```
Out[216]=
```
⟨ε, 5, ε⟩
```
In[217]:=
```
DelFOT[5, ⟨ε, 5, ε⟩]
```
Out[217]=
```
ε
```

In[218]:=
```
DelFOT[10, ⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩]
```
Out[218]=
```
⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩
```
In[219]:=
```
DelFOT[5, ⟨⟨ε, 5, ε⟩, 3, ε⟩]
```
Out[219]=
```
⟨ε, 3, ε⟩
```
In[220]:=
```
DelFOT[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
```
Out[220]=
```
⟨ε, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩
```
In[221]:=
```
DelFOT[7, ⟨⟨ε, 5, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
```
Out[221]=
```
⟨⟨ε, 5, ε⟩, 3, ⟨ε, 1, ε⟩⟩
```
In[222]:=
```
DelFOT[10, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 10, ε⟩⟩⟩]
```
Out[222]=
```
⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ε⟩⟩
```
In[223]:=
```
DelFOT[5, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩]
```
Out[223]=
```
⟨ε, 7, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩
```

Functional algorithm

ALGORITHM (DELFOTF)

In[60]:=
```
∀ (DelFOTf[a, T] ==
a,T
    ε                              ⇐  isEmpty[T]
    Left[T] ≍ Right[T]             ⇐  a == root[T]
    ⟨DelFOTf[a, Left[T]],          ⇐  a ⊲ Left[T]
     root[T], Right[T]⟩
    ⟨Left[T], root[T],             ⇐  a ⊲ Right[T]
     DelFOTf[a, Right[T]]⟩
    T                              ⇐  True
)
```
(DelFOTf)

Computation

In[233]:=
```
DelFOTf[3, ε]
```
Out[233]=
```
ε
```

In[234]:=
```
DelFOTf[3, ⟨ε, 5, ε⟩]
```
Out[234]=
```
⟨ε, 5, ε⟩
```
In[235]:=
```
DelFOTf[5, ⟨ε, 5, ε⟩]
```
Out[235]=
```
ε
```
In[236]:=
```
DelFOTf[10, ⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩]
```
Out[236]=
```
⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩
```
In[237]:=
```
DelFOTf[5, ⟨⟨ε, 5, ε⟩, 3, ε⟩]
```
Out[237]=
```
⟨ε, 3, ε⟩
```
In[238]:=
```
DelFOTf[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
```
Out[238]=
```
⟨ε, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩
```
In[239]:=
```
DelFOTf[7, ⟨⟨ε, 5, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
```
Out[239]=
```
⟨⟨ε, 5, ε⟩, 3, ⟨ε, 1, ε⟩⟩
```
In[240]:=
```
DelFOTf[10, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 10, ε⟩⟩⟩]
```
Out[240]=
```
⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ε⟩⟩
```
In[241]:=
```
DelFOTf[5, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩]
```
Out[241]=
```
⟨ε, 7, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩
```

DelAOST: Delete all occurrences of an element from a sorted tree

Pattern matching algorithm

ALGORITHM (DELAOST)

In[61]:=
$$\forall_{a} \left(\text{DelAOST}[a, \varepsilon] = \varepsilon \right) \qquad (DelAOST - 0)$$

In[62]:=
$$\underset{a<b}{\forall_{a,b,L,R}} \left(\text{DelAOST}[a, \langle L, b, R\rangle] = \langle \text{DelAOST}[a, L], b, R\rangle \right) \qquad (DelAOST - 1)$$

In[63]:=
$$\underset{a=b}{\forall_{a,b,L,R}} \left(\text{DelAOST}[a, \langle L, b, R\rangle] = \left(\text{DelAOST}[a, L] \asymp \text{DelAOST}[a, R]\right) \right) \qquad (DelAOST - 2)$$

In[64]:=

```
( ∀_{a,b,L,R; a>b} (DelAOST[a, ⟨L, b, R⟩] == ⟨L, b, DelAOST[a, R]⟩))
```

(DelAOST − 3)

Computation

In[242]:= `DelAOST[3, ε]`

Out[242]= `ε`

In[243]:= `DelAOST[3, ⟨ε, 5, ε⟩]`

Out[243]= `⟨ε, 5, ε⟩`

In[244]:= `DelAOST[5, ⟨ε, 5, ε⟩]`

Out[244]= `ε`

In[245]:= `DelAOST[10, ⟨ε, 5, ε⟩]`

Out[245]= `⟨ε, 5, ε⟩`

In[246]:= `DelAOST[3, ⟨⟨ε, 3, ε⟩, 5, ε⟩]`

Out[246]= `⟨ε, 5, ε⟩`

In[247]:= `DelAOST[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]`

Out[247]= `⟨ε, 7, ⟨ε, 10, ε⟩⟩`

In[248]:= `DelAOST[7, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]`

Out[248]= `⟨⟨ε, 3, ε⟩, 5, ⟨ε, 10, ε⟩⟩`

In[249]:= `DelAOST[10, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]`

Out[249]= `⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ε⟩⟩`

In[250]:= `DelAOST[5, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]`

Out[250]= `⟨ε, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩`

.

Functional algorithm

ALGORITHM (DELAOSTF)

In[65]:=

$$\underset{a,T}{\forall}\left(\text{DelAOSTf}[a,T] == \left(\begin{cases}\varepsilon & \Leftarrow \text{isEmpty}[T] \\ \langle \text{DelAOSTf}[a, \text{Left}[T]], \text{root}[T], \text{Right}[T]\rangle & \Leftarrow a < \text{root}[T] \\ (\text{DelAOSTf}[a, \text{Left}[T]] \asymp \text{DelAOSTf}[a, \text{Right}[T]]) & \Leftarrow a == \text{root}[T] \\ \langle \text{Left}[T], \text{root}[T], \text{DelAOSTf}[a, \text{Right}[T]]\rangle & \Leftarrow a > \text{root}[T]\end{cases}\right)\right)$$

(*DelAOSTf*)

Computation

```
In[251]:= DelAOSTf[3, ε]
Out[251]= ε

In[252]:= DelAOSTf[3, ⟨ε, 5, ε⟩]
Out[252]= ⟨ε, 5, ε⟩

In[253]:= DelAOSTf[5, ⟨ε, 5, ε⟩]
Out[253]= ε

In[254]:= DelAOSTf[10, ⟨ε, 5, ε⟩]
Out[254]= ⟨ε, 5, ε⟩

In[255]:= DelAOSTf[3, ⟨⟨ε, 3, ε⟩, 5, ε⟩]
Out[255]= ⟨ε, 5, ε⟩

In[259]:= DelAOST[3, ⟨⟨ε, 3, ε⟩, 5, ε⟩]
Out[259]= DelAOST[3, ⟨⟨ε, 3, ε⟩, 5, ε⟩]

In[256]:= DelAOSTf[7, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
Out[256]= ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 10, ε⟩⟩
```

In[257]:=
```
DelAOSTf[10, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
```
Out[257]=
```
⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ε⟩⟩
```

In[258]:=
```
DelAOSTf[5, ⟨⟨ε, 3, ε⟩, 5, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩]
```
Out[258]=
```
⟨ε, 3, ⟨ε, 7, ⟨ε, 10, ε⟩⟩⟩
```

DelAOT: Delete all occurrences of an element from an unsorted tree

Pattern matching algorithm

Algorithm (DelAOT)

In[66]:= $\left(\forall_{a} \left(\text{DelAOT}[a, \varepsilon] == \varepsilon\right)\right)$ (DelAOT − 0)

In[67]:= $\left(\forall_{\substack{a,b,L,R \\ a\neq b}} \left(\text{DelAOT}[a, \langle L, b, R\rangle] == \langle \text{DelAOT}[a, L], b, \text{DelAOT}[a, R]\rangle\right)\right)$ (DelAOT − 1)

In[68]:= $\left(\forall_{\substack{a,b,L,R \\ a==b}} \left(\text{DelAOT}[a, \langle L, b, R\rangle] == \left(\text{DelAOT}[a, L] \asymp \text{DelAOT}[a, R]\right)\right)\right)$ (DelAOT − 2)

Computation

In[260]:=
```
DelAOT[3, ε]
```
Out[260]=
```
ε
```

In[261]:=
```
DelAOT[3, ⟨ε, 5, ε⟩]
```
Out[261]=
```
⟨ε, 5, ε⟩
```

In[262]:=
```
DelAOT[5, ⟨ε, 5, ε⟩]
```
Out[262]=
```
ε
```

In[263]:=
```
DelAOT[10, ⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩]
```
Out[263]=
```
⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩
```

In[264]:=
```
DelAOT[5, ⟨⟨ε, 5, ε⟩, 3, ε⟩]
```
Out[264]=
```
⟨ε, 3, ε⟩
```

In[265]:=
```
DelAOT[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
```
Out[265]=
```
⟨ε, 7, ⟨ε, 1, ε⟩⟩
```
In[266]:=
```
DelAOT[7, ⟨⟨ε, 5, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
```
Out[266]=
```
⟨⟨ε, 5, ε⟩, 3, ⟨ε, 1, ε⟩⟩
```
In[267]:=
```
DelAOT[10, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 10, ε⟩⟩⟩]
```
Out[267]=
```
⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ε⟩⟩
```
In[268]:=
```
DelAOT[5, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩]
```
Out[268]=
```
⟨ε, 7, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩
```
In[269]:=
```
DelAOT[5, ⟨⟨ε, 5, ε⟩, 5, ⟨ε, 5, ⟨ε, 5, ε⟩⟩⟩]
```
Out[269]=
```
ε
```

Functional algorithm

ALGORITHM (DELAOTF)

In[69]:=
```
∀ a,T (DelAOTf[a, T] == [ ε                                             ⇐ isEmpty[T]
                          ⟨DelAOTf[a, Left[T]], root[T], DelAOTf[a, Right[T]]⟩   ⇐ a ≠ root[T]
                          DelAOTf[a, Left[T]] ≍ DelAOTf[a, Right[T]]      ⇐ True ])
```
(DelAOTf)

Computation

In[270]:=
```
DelAOTf[3, ε]
```
Out[270]=
```
ε
```
In[271]:=
```
DelAOTf[3, ⟨ε, 5, ε⟩]
```
Out[271]=
```
⟨ε, 5, ε⟩
```
In[272]:=
```
DelAOTf[5, ⟨ε, 5, ε⟩]
```
Out[272]=
```
ε
```

```
In[273]:= DelAOTf[10, ⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩]
Out[273]= ⟨⟨ε, 3, ε⟩, 1, ⟨ε, 2, ⟨ε, 1, ε⟩⟩⟩

In[274]:= DelAOTf[5, ⟨⟨ε, 5, ε⟩, 3, ε⟩]
Out[274]= ⟨ε, 3, ε⟩

In[275]:= DelAOTf[3, ⟨⟨ε, 3, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
Out[275]= ⟨ε, 7, ⟨ε, 1, ε⟩⟩

In[276]:= DelAOTf[7, ⟨⟨ε, 5, ε⟩, 3, ⟨ε, 7, ⟨ε, 1, ε⟩⟩⟩]
Out[276]= ⟨⟨ε, 5, ε⟩, 3, ⟨ε, 1, ε⟩⟩

In[277]:= DelAOTf[10, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 10, ε⟩⟩⟩]
Out[277]= ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ε⟩⟩

In[278]:= DelAOTf[5, ⟨⟨ε, 7, ε⟩, 5, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩]
Out[278]= ⟨ε, 7, ⟨ε, 3, ⟨ε, 1, ε⟩⟩⟩

In[279]:= DelAOTf[5, ⟨⟨ε, 5, ε⟩, 5, ⟨ε, 5, ⟨ε, 5, ε⟩⟩⟩]
Out[279]= ε
```