

```

1  ┌────────────────────────── MODULE PieceInclusionProof ───────────────────────────┐
2  EXTENDS Integers, TLC, Sequences, FiniteSets

4  SIZE  $\triangleq$  256      Must be a power of 2. TODO: Make CONSTANT and add ASSUME .
5  HEIGHT  $\triangleq$  9    TODO: Calculate this from SIZE, as  $\log_2(\text{SIZE}) + 1$ .

7  --algorithm PieceInclusionProof

9  variables
10     HashCounter = - 1 ;
11     HashRecord =  $\langle \rangle$  ;

13  macro RepCompress(a, b, height, var) begin
14     if ( $\langle a, b, height \rangle \in \text{DOMAIN } HashRecord$ ) then
15         var := HashRecord[ $\langle a, b, height \rangle$ ] ;
16     else
17         HashCounter := HashCounter - 1 ;
18         HashRecord := ( $\langle a, b, height \rangle \rightarrow HashCounter$ ) @@ HashRecord ;
19         var := HashCounter
20     end if ;
21 end macro ;

23 process test_hash = "test hash"
24 variables h1, h2, h3 ;

26 begin
27     L1:
28         RepCompress(1, 2, 0, h1) ;
29     L2:
30         RepCompress(1, 2, 0, h2) ;
31     L3:
32         RepCompress(2, 1, 0, h3) ;

34         assert h1 = h2 ;
35         assert h1  $\neq$  h3 ;
36 end process ;

38 fair process merkle_tree = "merkle tree"
39 variables h,
40     input, row, rowSize, nextRow, index, proof_element, root, challenge,
41     cursor_index, cursor_row, cursor_element, proof_index, challenge_path_acc, place_acc,
42     rows =  $\langle \rangle$  ;
43     height = - 1 ;

45     proof_path =  $\langle \rangle$  ;
46     proof_elements =  $\langle \rangle$  ;
47 begin
48     BuildTree:

```

```

49   input := [i ∈ 1 .. SIZE ↦ i];
50   row := input;
51   rows := ⟨⟩ ;
52   RowLoop:
53     height := height + 1;
54     rows := Append(rows, row);
55     It would be nice to make this assert an invariant, but how do we make an invariant
56     over a process variable?
57     assert height > 1 ⇒ Len(rows[height - 1]) = 2 * Len(rows[height]);

59     nextRow := ⟨⟩ ;
60     index := 1;
61     rowSize := Cardinality(DOMAIN row);
62     if rowSize > 1 then
63       HashRow:
64         RepCompress(row[index], row[index + 1], height, h);
65         nextRow := Append(nextRow, h);
66         Advance: index := index + 2;
67         if index < Cardinality(DOMAIN row) then
68           goto HashRow;
69         else
70           row := nextRow;
71         end if ;
72         Repeat: goto RowLoop;
73       else
74         assert Len(rows) = HEIGHT;
75       end if ;
76   Proofs:
77     challenge := 1;
78   MakeProof:
79     cursor_index := challenge;
80     cursor_row := 1;
81     cursor_element := rows[cursor_row][cursor_index];
82     proof_path := ⟨⟩ ;
83     proof_elements := ⟨⟩ ;
84   S1:
85     if cursor_index%2 = 1 then
86       proof_path := Append(proof_path, FALSE);
87       proof_element := rows[cursor_row][cursor_index + 1];
88       RepCompress(rows[cursor_row][cursor_index],
89                   proof_element,
90                   cursor_row - 1,
91                   cursor_element);
92     else
93       proof_path := Append(proof_path, TRUE);

```

```

94         proof_element := rows[cursor_row][cursor_index - 1];
95         RepCompress(proof_element,
96                     rows[cursor_row][cursor_index],
97                     cursor_row - 1,
98                     cursor_element);
99     end if ;

101     proof_elements := Append(proof_elements, proof_element);
102 ProofLoop:
103     cursor_row := cursor_row + 1;
104     cursor_index := (cursor_index + 1) ÷ 2;
105     if cursor_row < Len(rows) then
106         goto S1;
107     end if ;
108 FinishProof:
109     root := rows[Len(rows)][1];

111 CheckProof:
112     proof_index := 1;
113     height := 0;
114     cursor_index := challenge;
115     cursor_element := rows[height + 1][cursor_index];
116     challenge_path_acc := 0;
117     place_acc := 1;
118 ProofCheckLoop:
119     if proof_path[proof_index] then
120         RepCompress(proof_elements[proof_index],
121                     cursor_element,
122                     height,
123                     cursor_element);
124         challenge_path_acc := challenge_path_acc + place_acc;
125     else
126         RepCompress(cursor_element,
127                     proof_elements[proof_index],
128                     height,
129                     cursor_element);
130     end if ;
131     place_acc := place_acc * 2;

133     proof_index := proof_index + 1;
134     height := height + 1;
135     if height < Len(proof_elements) then
136         goto ProofCheckLoop;
137     end if ;

139 CheckRoot:

```

```

140   assert cursor_element = root ;
141   assert challenge_path_acc = challenge - 1 ;   challenges are 1-indexed because TLA+.
142
143   IncrementChallenge:
144     challenge := challenge + 1 ;
145     if challenge ≤ Len(input) then
146       goto MakeProof ;
147     end if ;
148 end process ;
149
150 end algorithm ;

```

152 **BEGIN TRANSLATION**

153 **CONSTANT** *defaultInitValue*

154 **VARIABLES** *HashCounter*, *HashRecord*, *pc*, *h1*, *h2*, *h3*, *h*, *input*, *row*, *rowSize*,
155 *nextRow*, *index*, *proof_element*, *root*, *challenge*, *cursor_index*,
156 *cursor_row*, *cursor_element*, *proof_index*, *challenge_path_acc*,
157 *place_acc*, *rows*, *height*, *proof_path*, *proof_elements*

159 *vars* \triangleq \langle *HashCounter*, *HashRecord*, *pc*, *h1*, *h2*, *h3*, *h*, *input*, *row*, *rowSize*,
160 *nextRow*, *index*, *proof_element*, *root*, *challenge*, *cursor_index*,
161 *cursor_row*, *cursor_element*, *proof_index*, *challenge_path_acc*,
162 *place_acc*, *rows*, *height*, *proof_path*, *proof_elements* \rangle

164 *ProcSet* \triangleq {“test hash”} \cup {“merkle tree”}

166 *Init* \triangleq **Global variables**
167 \wedge *HashCounter* = - 1
168 \wedge *HashRecord* = $\langle \rangle$
169 **Process** *test_hash*
170 \wedge *h1* = *defaultInitValue*
171 \wedge *h2* = *defaultInitValue*
172 \wedge *h3* = *defaultInitValue*
173 **Process** *merkle_tree*
174 \wedge *h* = *defaultInitValue*
175 \wedge *input* = *defaultInitValue*
176 \wedge *row* = *defaultInitValue*
177 \wedge *rowSize* = *defaultInitValue*
178 \wedge *nextRow* = *defaultInitValue*
179 \wedge *index* = *defaultInitValue*
180 \wedge *proof_element* = *defaultInitValue*
181 \wedge *root* = *defaultInitValue*
182 \wedge *challenge* = *defaultInitValue*
183 \wedge *cursor_index* = *defaultInitValue*
184 \wedge *cursor_row* = *defaultInitValue*
185 \wedge *cursor_element* = *defaultInitValue*
186 \wedge *proof_index* = *defaultInitValue*

```

187       $\wedge$  challenge_path_acc = defaultInitValue
188       $\wedge$  place_acc = defaultInitValue
189       $\wedge$  rows =  $\langle \rangle$ 
190       $\wedge$  height = -1
191       $\wedge$  proof_path =  $\langle \rangle$ 
192       $\wedge$  proof_elements =  $\langle \rangle$ 
193       $\wedge$  pc = [self  $\in$  ProcSet  $\mapsto$  CASE self = "test hash"  $\rightarrow$  "L1"
194                                $\square$  self = "merkle tree"  $\rightarrow$  "BuildTree"]]

196  L1  $\triangleq$   $\wedge$  pc["test hash"] = "L1"
197       $\wedge$  IF ( $\langle 1, 2, 0 \rangle \in$  DOMAIN HashRecord)
198          THEN  $\wedge$  h1' = HashRecord[( $\langle 1, 2, 0 \rangle$ )]
199               $\wedge$  UNCHANGED  $\langle$ HashCounter, HashRecord $\rangle$ 
200          ELSE  $\wedge$  HashCounter' = HashCounter - 1
201               $\wedge$  HashRecord' = ( $\langle 1, 2, 0 \rangle$  :> HashCounter') @@ HashRecord
202               $\wedge$  h1' = HashCounter'
203       $\wedge$  pc' = [pc EXCEPT !["test hash"] = "L2"]
204       $\wedge$  UNCHANGED  $\langle$ h2, h3, h, input, row, rowSize, nextRow, index,
205                  proof_element, root, challenge, cursor_index, cursor_row,
206                  cursor_element, proof_index, challenge_path_acc,
207                  place_acc, rows, height, proof_path, proof_elements $\rangle$ 

209  L2  $\triangleq$   $\wedge$  pc["test hash"] = "L2"
210       $\wedge$  IF ( $\langle 1, 2, 0 \rangle \in$  DOMAIN HashRecord)
211          THEN  $\wedge$  h2' = HashRecord[( $\langle 1, 2, 0 \rangle$ )]
212               $\wedge$  UNCHANGED  $\langle$ HashCounter, HashRecord $\rangle$ 
213          ELSE  $\wedge$  HashCounter' = HashCounter - 1
214               $\wedge$  HashRecord' = ( $\langle 1, 2, 0 \rangle$  :> HashCounter') @@ HashRecord
215               $\wedge$  h2' = HashCounter'
216       $\wedge$  pc' = [pc EXCEPT !["test hash"] = "L3"]
217       $\wedge$  UNCHANGED  $\langle$ h1, h3, h, input, row, rowSize, nextRow, index,
218                  proof_element, root, challenge, cursor_index, cursor_row,
219                  cursor_element, proof_index, challenge_path_acc,
220                  place_acc, rows, height, proof_path, proof_elements $\rangle$ 

222  L3  $\triangleq$   $\wedge$  pc["test hash"] = "L3"
223       $\wedge$  IF ( $\langle 2, 1, 0 \rangle \in$  DOMAIN HashRecord)
224          THEN  $\wedge$  h3' = HashRecord[( $\langle 2, 1, 0 \rangle$ )]
225               $\wedge$  UNCHANGED  $\langle$ HashCounter, HashRecord $\rangle$ 
226          ELSE  $\wedge$  HashCounter' = HashCounter - 1
227               $\wedge$  HashRecord' = ( $\langle 2, 1, 0 \rangle$  :> HashCounter') @@ HashRecord
228               $\wedge$  h3' = HashCounter'
229       $\wedge$  Assert(h1 = h2, "Failure of assertion at line 34, column 9.")
230       $\wedge$  Assert(h1  $\neq$  h3', "Failure of assertion at line 35, column 9.")
231       $\wedge$  pc' = [pc EXCEPT !["test hash"] = "Done"]
232       $\wedge$  UNCHANGED  $\langle$ h1, h2, h, input, row, rowSize, nextRow, index,

```

233 $proof_element, root, challenge, cursor_index, cursor_row,$
 234 $cursor_element, proof_index, challenge_path_acc,$
 235 $place_acc, rows, height, proof_path, proof_elements\}$

237 $test_hash \triangleq L1 \vee L2 \vee L3$

239 $BuildTree \triangleq \wedge pc["merkle\ tree"] = "BuildTree"$
 240 $\wedge input' = [i \in 1 \dots SIZE \mapsto i]$
 241 $\wedge row' = input'$
 242 $\wedge rows' = \langle \rangle$
 243 $\wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "RowLoop"]$
 244 $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord, h1, h2, h3, h, rowSize,$
 245 $nextRow, index, proof_element, root, challenge,$
 246 $cursor_index, cursor_row, cursor_element,$
 247 $proof_index, challenge_path_acc, place_acc,$
 248 $height, proof_path, proof_elements\}$

250 $RowLoop \triangleq \wedge pc["merkle\ tree"] = "RowLoop"$
 251 $\wedge height' = height + 1$
 252 $\wedge rows' = Append(rows, row)$
 253 $\wedge Assert(height' > 1 \Rightarrow Len(rows'[height' - 1]) = 2 * Len(rows'[height']),$
 254 $"Failure\ of\ assertion\ at\ line\ 57,\ column\ 9.")$
 255 $\wedge nextRow' = \langle \rangle$
 256 $\wedge index' = 1$
 257 $\wedge rowSize' = Cardinality(DOMAIN\ row)$
 258 $\wedge \text{IF } rowSize' > 1$
 259 $\quad \text{THEN } \wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "HashRow"]$
 260 $\quad \text{ELSE } \wedge Assert(Len(rows') = HEIGHT,$
 261 $\quad \quad "Failure\ of\ assertion\ at\ line\ 74,\ column\ 13.")$
 262 $\quad \wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "Proofs"]$
 263 $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord, h1, h2, h3, h, input, row,$
 264 $proof_element, root, challenge, cursor_index,$
 265 $cursor_row, cursor_element, proof_index,$
 266 $challenge_path_acc, place_acc, proof_path,$
 267 $proof_elements\}$

269 $HashRow \triangleq \wedge pc["merkle\ tree"] = "HashRow"$
 270 $\wedge \text{IF } (\langle (row[index]), (row[index + 1]), height \rangle \in \text{DOMAIN } HashRecord)$
 271 $\quad \text{THEN } \wedge h' = HashRecord[\langle (row[index]), (row[index + 1]), height \rangle]$
 272 $\quad \wedge \text{UNCHANGED } \langle HashCounter, HashRecord \rangle$
 273 $\quad \text{ELSE } \wedge HashCounter' = HashCounter - 1$
 274 $\quad \wedge HashRecord' = (\langle (row[index]), (row[index + 1]), height \rangle :> HashCounter') @ @ Ha$
 275 $\quad \wedge h' = HashCounter'$
 276 $\wedge nextRow' = Append(nextRow, h')$
 277 $\wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "Advance"]$
 278 $\wedge \text{UNCHANGED } \langle h1, h2, h3, input, row, rowSize, index,$

```

279                                     proof_element, root, challenge, cursor_index,
280                                     cursor_row, cursor_element, proof_index,
281                                     challenge_path_acc, place_acc, rows, height,
282                                     proof_path, proof_elements)

284 Advance  $\triangleq$   $\wedge pc["merkle\ tree"] = "Advance"$ 
285                $\wedge index' = index + 2$ 
286                $\wedge IF\ index' < Cardinality(DOMAIN\ row)$ 
287                 THEN  $\wedge pc' = [pc\ EXCEPT\ !["merkle\ tree"] = "HashRow"]$ 
288                    $\wedge row' = row$ 
289                 ELSE  $\wedge row' = nextRow$ 
290                    $\wedge pc' = [pc\ EXCEPT\ !["merkle\ tree"] = "Repeat"]$ 
291                $\wedge UNCHANGED\ \langle HashCounter, HashRecord, h1, h2, h3, h, input,$ 
292                  $rowSize, nextRow, proof\_element, root, challenge,$ 
293                  $cursor\_index, cursor\_row, cursor\_element,$ 
294                  $proof\_index, challenge\_path\_acc, place\_acc, rows,$ 
295                  $height, proof\_path, proof\_elements \rangle$ 

297 Repeat  $\triangleq$   $\wedge pc["merkle\ tree"] = "Repeat"$ 
298                $\wedge pc' = [pc\ EXCEPT\ !["merkle\ tree"] = "RowLoop"]$ 
299                $\wedge UNCHANGED\ \langle HashCounter, HashRecord, h1, h2, h3, h, input, row,$ 
300                  $rowSize, nextRow, index, proof\_element, root,$ 
301                  $challenge, cursor\_index, cursor\_row, cursor\_element,$ 
302                  $proof\_index, challenge\_path\_acc, place\_acc, rows,$ 
303                  $height, proof\_path, proof\_elements \rangle$ 

305 Proofs  $\triangleq$   $\wedge pc["merkle\ tree"] = "Proofs"$ 
306                $\wedge challenge' = 1$ 
307                $\wedge pc' = [pc\ EXCEPT\ !["merkle\ tree"] = "MakeProof"]$ 
308                $\wedge UNCHANGED\ \langle HashCounter, HashRecord, h1, h2, h3, h, input, row,$ 
309                  $rowSize, nextRow, index, proof\_element, root,$ 
310                  $cursor\_index, cursor\_row, cursor\_element,$ 
311                  $proof\_index, challenge\_path\_acc, place\_acc, rows,$ 
312                  $height, proof\_path, proof\_elements \rangle$ 

314 MakeProof  $\triangleq$   $\wedge pc["merkle\ tree"] = "MakeProof"$ 
315                $\wedge cursor\_index' = challenge$ 
316                $\wedge cursor\_row' = 1$ 
317                $\wedge cursor\_element' = rows[cursor\_row'][cursor\_index']$ 
318                $\wedge proof\_path' = \langle \rangle$ 
319                $\wedge proof\_elements' = \langle \rangle$ 
320                $\wedge pc' = [pc\ EXCEPT\ !["merkle\ tree"] = "S1"]$ 
321                $\wedge UNCHANGED\ \langle HashCounter, HashRecord, h1, h2, h3, h, input,$ 
322                  $row, rowSize, nextRow, index, proof\_element, root,$ 
323                  $challenge, proof\_index, challenge\_path\_acc,$ 
324                  $place\_acc, rows, height \rangle$ 

```

```

326  $S1 \triangleq \wedge pc["merkle\ tree"] = "S1"$ 
327  $\wedge \text{IF } cursor\_index \% 2 = 1$ 
328   THEN  $\wedge proof\_path' = Append(proof\_path, FALSE)$ 
329    $\wedge proof\_element' = rows[cursor\_row][cursor\_index + 1]$ 
330    $\wedge \text{IF } (\langle (rows[cursor\_row][cursor\_index]), proof\_element', (cursor\_row - 1) \rangle \in \text{DOMAIN } H)$ 
331     THEN  $\wedge cursor\_element' = HashRecord[\langle (rows[cursor\_row][cursor\_index]), proof\_element' \rangle]$ 
332      $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord \rangle$ 
333     ELSE  $\wedge HashCounter' = HashCounter - 1$ 
334      $\wedge HashRecord' = (\langle (rows[cursor\_row][cursor\_index]), proof\_element', (cursor\_row - 1) \rangle \in \text{DOMAIN } H)$ 
335      $\wedge cursor\_element' = HashCounter'$ 
336   ELSE  $\wedge proof\_path' = Append(proof\_path, TRUE)$ 
337    $\wedge proof\_element' = rows[cursor\_row][cursor\_index - 1]$ 
338    $\wedge \text{IF } (\langle proof\_element', (rows[cursor\_row][cursor\_index]), (cursor\_row - 1) \rangle \in \text{DOMAIN } H)$ 
339     THEN  $\wedge cursor\_element' = HashRecord[\langle proof\_element', (rows[cursor\_row][cursor\_index]) \rangle]$ 
340      $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord \rangle$ 
341     ELSE  $\wedge HashCounter' = HashCounter - 1$ 
342      $\wedge HashRecord' = (\langle proof\_element', (rows[cursor\_row][cursor\_index]), (cursor\_row - 1) \rangle \in \text{DOMAIN } H)$ 
343      $\wedge cursor\_element' = HashCounter'$ 
344    $\wedge proof\_elements' = Append(proof\_elements, proof\_element')$ 
345    $\wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "ProofLoop"]$ 
346    $\wedge \text{UNCHANGED } \langle h1, h2, h3, h, input, row, rowSize, nextRow, index, root,$ 
347      $challenge, cursor\_index, cursor\_row, proof\_index,$ 
348      $challenge\_path\_acc, place\_acc, rows, height \rangle$ 
349
350  $ProofLoop \triangleq \wedge pc["merkle\ tree"] = "ProofLoop"$ 
351    $\wedge cursor\_row' = cursor\_row + 1$ 
352    $\wedge cursor\_index' = ((cursor\_index + 1) \div 2)$ 
353    $\wedge \text{IF } cursor\_row' < Len(rows)$ 
354     THEN  $\wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "S1"]$ 
355     ELSE  $\wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "FinishProof"]$ 
356    $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord, h1, h2, h3, h, input,$ 
357      $row, rowSize, nextRow, index, proof\_element, root,$ 
358      $challenge, cursor\_element, proof\_index,$ 
359      $challenge\_path\_acc, place\_acc, rows, height,$ 
360      $proof\_path, proof\_elements \rangle$ 
361
362  $FinishProof \triangleq \wedge pc["merkle\ tree"] = "FinishProof"$ 
363    $\wedge root' = rows[Len(rows)][1]$ 
364    $\wedge pc' = [pc \text{ EXCEPT } !["merkle\ tree"] = "CheckProof"]$ 
365    $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord, h1, h2, h3, h, input,$ 
366      $row, rowSize, nextRow, index, proof\_element,$ 
367      $challenge, cursor\_index, cursor\_row,$ 
368      $cursor\_element, proof\_index, challenge\_path\_acc,$ 
369      $place\_acc, rows, height, proof\_path,$ 
370      $proof\_elements \rangle$ 

```



```

372 CheckProof  $\triangleq$   $\wedge pc["merkle tree"] = "CheckProof"$ 
373            $\wedge proof\_index' = 1$ 
374            $\wedge height' = 0$ 
375            $\wedge cursor\_index' = challenge$ 
376            $\wedge cursor\_element' = rows[height' + 1][cursor\_index']$ 
377            $\wedge challenge\_path\_acc' = 0$ 
378            $\wedge place\_acc' = 1$ 
379            $\wedge pc' = [pc \text{ EXCEPT } !["merkle tree"] = "ProofCheckLoop"]$ 
380            $\wedge \text{UNCHANGED } \langle HashCounter, HashRecord, h1, h2, h3, h, input,$ 
381              $row, rowSize, nextRow, index, proof\_element,$ 
382              $root, challenge, cursor\_row, rows, proof\_path,$ 
383              $proof\_elements \rangle$ 

385 ProofCheckLoop  $\triangleq$   $\wedge pc["merkle tree"] = "ProofCheckLoop"$ 
386            $\wedge \text{IF } proof\_path[proof\_index]$ 
387              $\text{THEN } \wedge \text{IF } (\langle (proof\_elements[proof\_index]), cursor\_element, height \rangle \in \text{DOMAIN } H)$ 
388                $\text{THEN } \wedge cursor\_element' = HashRecord[\langle (proof\_elements[proof\_index]$ 
389                  $\wedge \text{UNCHANGED } \langle HashCounter,$ 
390                    $HashRecord \rangle$ 
391                $\text{ELSE } \wedge HashCounter' = HashCounter - 1$ 
392                  $\wedge HashRecord' = (\langle (proof\_elements[proof\_index]), cursor\_element,$ 
393                    $\wedge cursor\_element' = HashCounter'$ 
394                  $\wedge challenge\_path\_acc' = challenge\_path\_acc + place\_acc$ 
395              $\text{ELSE } \wedge \text{IF } (\langle cursor\_element, (proof\_elements[proof\_index]), height \rangle \in \text{DOMAIN } H)$ 
396                $\text{THEN } \wedge cursor\_element' = HashRecord[\langle cursor\_element, (proof\_element,$ 
397                  $\wedge \text{UNCHANGED } \langle HashCounter,$ 
398                    $HashRecord \rangle$ 
399                $\text{ELSE } \wedge HashCounter' = HashCounter - 1$ 
400                  $\wedge HashRecord' = (\langle cursor\_element, (proof\_elements[proof\_index]$ 
401                    $\wedge cursor\_element' = HashCounter'$ 
402                  $\wedge \text{UNCHANGED } challenge\_path\_acc$ 
403                  $\wedge place\_acc' = place\_acc * 2$ 
404                  $\wedge proof\_index' = proof\_index + 1$ 
405                  $\wedge height' = height + 1$ 
406                  $\wedge \text{IF } height' < Len(proof\_elements)$ 
407                    $\text{THEN } \wedge pc' = [pc \text{ EXCEPT } !["merkle tree"] = "ProofCheckLoop"]$ 
408                    $\text{ELSE } \wedge pc' = [pc \text{ EXCEPT } !["merkle tree"] = "CheckRoot"]$ 
409                  $\wedge \text{UNCHANGED } \langle h1, h2, h3, h, input, row, rowSize, nextRow,$ 
410                    $index, proof\_element, root, challenge,$ 
411                    $cursor\_index, cursor\_row, rows, proof\_path,$ 
412                    $proof\_elements \rangle$ 

414 CheckRoot  $\triangleq$   $\wedge pc["merkle tree"] = "CheckRoot"$ 
415            $\wedge \text{Assert}(cursor\_element = root,$ 
416             "Failure of assertion at line 140, column 9.")

```

```

417  $\wedge \text{Assert}(\text{challenge\_path\_acc} = \text{challenge} - 1,$ 
418  $\quad \text{"Failure of assertion at line 141, column 9."})$ 
419  $\wedge pc' = [pc \text{ EXCEPT } ![\text{"merkle tree"}] = \text{"IncrementChallenge"}]$ 
420  $\wedge \text{UNCHANGED } \langle \text{HashCounter}, \text{HashRecord}, h1, h2, h3, h, \text{input},$ 
421  $\quad \text{row}, \text{rowSize}, \text{nextRow}, \text{index}, \text{proof\_element}, \text{root},$ 
422  $\quad \text{challenge}, \text{cursor\_index}, \text{cursor\_row},$ 
423  $\quad \text{cursor\_element}, \text{proof\_index}, \text{challenge\_path\_acc},$ 
424  $\quad \text{place\_acc}, \text{rows}, \text{height}, \text{proof\_path},$ 
425  $\quad \text{proof\_elements} \rangle$ 
427  $\text{IncrementChallenge} \triangleq \wedge pc[\text{"merkle tree"}] = \text{"IncrementChallenge"}$ 
428  $\wedge \text{challenge}' = \text{challenge} + 1$ 
429  $\wedge \text{IF } \text{challenge}' \leq \text{Len}(\text{input})$ 
430  $\quad \text{THEN } \wedge pc' = [pc \text{ EXCEPT } ![\text{"merkle tree"}] = \text{"MakeProof"}]$ 
431  $\quad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![\text{"merkle tree"}] = \text{"Done"}]$ 
432  $\wedge \text{UNCHANGED } \langle \text{HashCounter}, \text{HashRecord}, h1, h2, h3, h,$ 
433  $\quad \text{input}, \text{row}, \text{rowSize}, \text{nextRow}, \text{index},$ 
434  $\quad \text{proof\_element}, \text{root}, \text{cursor\_index},$ 
435  $\quad \text{cursor\_row}, \text{cursor\_element}, \text{proof\_index},$ 
436  $\quad \text{challenge\_path\_acc}, \text{place\_acc}, \text{rows},$ 
437  $\quad \text{height}, \text{proof\_path}, \text{proof\_elements} \rangle$ 
439  $\text{merkle\_tree} \triangleq \text{BuildTree} \vee \text{RowLoop} \vee \text{HashRow} \vee \text{Advance} \vee \text{Repeat}$ 
440  $\quad \vee \text{Proofs} \vee \text{MakeProof} \vee S1 \quad \vee \text{ProofLoop} \vee \text{FinishProof}$ 
441  $\quad \vee \text{CheckProof} \vee \text{ProofCheckLoop} \vee \text{CheckRoot}$ 
442  $\quad \vee \text{IncrementChallenge}$ 
444  $\text{Next} \triangleq \text{test\_hash} \vee \text{merkle\_tree}$ 
445  $\quad \vee \text{Disjunct to prevent deadlock on termination}$ 
446  $\quad ((\forall self \in \text{ProcSet} : pc[self] = \text{"Done"}) \wedge \text{UNCHANGED } vars)$ 
448  $\text{Spec} \triangleq \wedge \text{Init} \wedge \Box [Next]_{vars}$ 
449  $\quad \wedge \text{WF}_{vars}(\text{merkle\_tree})$ 
451  $\text{Termination} \triangleq \Diamond (\forall self \in \text{ProcSet} : pc[self] = \text{"Done"})$ 
453 END TRANSLATION
454

```