1 PDF properties.

1.1 Page Tree Inheritance Property.

```
Tree:

Root is unique d satisfying  d.<"Id"> =  d1.<"Pages">
where unique d1 satisfies  d1.<"Type"> = "Catalog"  .

forall d in Tree, d0 is Child(d) where d0.<"Id"> in d.<"Kids">  and  d0.<"Type"> in [ "Page", "Pages" ]   .


Attributes:

forall d in Root,
d.<is_Resources_defined> = iskeydefined( d , "Resources" ).

forall d in Root,
d.<is_MediaBox_defined> = iskeydefined( d , "MediaBox" ).

forall d1 in Tree,
forall d2 in Child(d1),
d2.<is_Resources_defined> =  d1.<is_Resources_defined> or iskeydefined(d2 , "Resources" )  .

forall d1 in Tree,
forall d2 in Child(d1),
d2.<is_MediaBox_defined> =  d1.<is_MediaBox_defined> or iskeydefined(d2, "MediaBox" )  .

Specification Conditions:

forall d in Leaves,   d.<is_Resources_defined> = True  .

forall d in Leaves,   d.<is_MediaBox_defined> = True  .
```

2 HTML properties.

2.1 'P' isn't nested.

```
Tree:

Root is unique d satisfying  d.<"Name"> = "html"  .

forall d in Tree, d0 is Child(d) where  d0.<"Id"> in d.<"Kids">  and  d0.<"Name"> !=  "html" .


Graph Property:

P is d satisfying d.<"Name"> =  "p" .

Specification Conditions:

forall d in P, isempty ( d.<"Kids"> ).
```

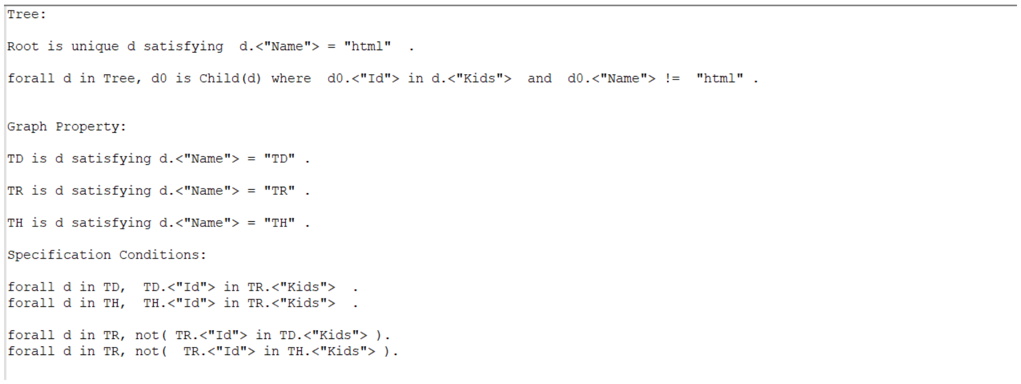
2.2 Elements inside 'head' referenced atmost once.

```
Tree:

Root is unique d satisfying  d.<"Name"> = "html"  .

forall d in Tree, d0 is Child(d) where  d0.<"Id"> in d.<"Kids">  and  d0.<"Name"> !=  "html" .


Graph Property:

Head is d satisfying d.<"Name"> = "head" .

Ref is (d1, d2) satisfying d1.<"href"> = REFSTRING(d2.<"id">)
where d1 in [ Tree ]
where d2 in [ Tree ] .

Specification Conditions:

forall d1 in Head,
forall (d2, d3) in Ref,
forall (d4, d5) in Ref,
(d3 = d5  and   d3.<"Id"> in PATH(d1.<"Id">, Tree)) implies  d2 = d4  .
```

2.3 Each element has unique 'id' value.

```
Tree:

Root is unique d satisfying  d.<"Name"> = "html"  .

forall d in Tree, d0 is Child(d) where  d0.<"Id"> in d.<"Kids">  and  d0.<"Name"> !=  "html" .


Attributes:

forall d in Leaves,
d.<id_def> = make_singleton_array( d.<"id"> ) .

forall d in Tree,
d.<id_def> =  append_all_children_attributes(d.<"Id"> , <id_def> )  union  make_singleton_array(d.<"id"> ) .

Specification Conditions:

forall d in Root, is_set( d.<id_def> ).
```

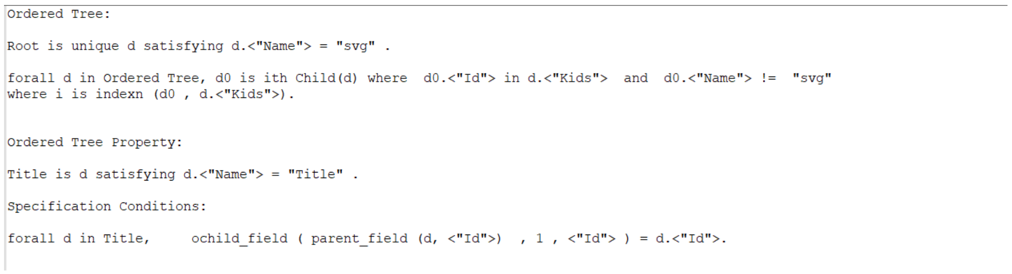
2.4 TD and TH are nested inside TR, but never vice versa.



```
Tree:

Root is unique d satisfying  d.<"Name"> = "html"  .

forall d in Tree, d0 is Child(d) where  d0.<"Id"> in d.<"Kids">  and  d0.<"Name"> !=  "html" .


Graph Property:

TD is d satisfying d.<"Name"> = "TD" .

TR is d satisfying d.<"Name"> = "TR" .

TH is d satisfying d.<"Name"> = "TH" .

Specification Conditions:

forall d in TD,  TD.<"Id"> in TR.<"Kids">  .
forall d in TH,  TH.<"Id"> in TR.<"Kids">  .

forall d in TR, not( TR.<"Id"> in TD.<"Kids"> ).
forall d in TR, not(  TR.<"Id"> in TH.<"Kids"> ).
```

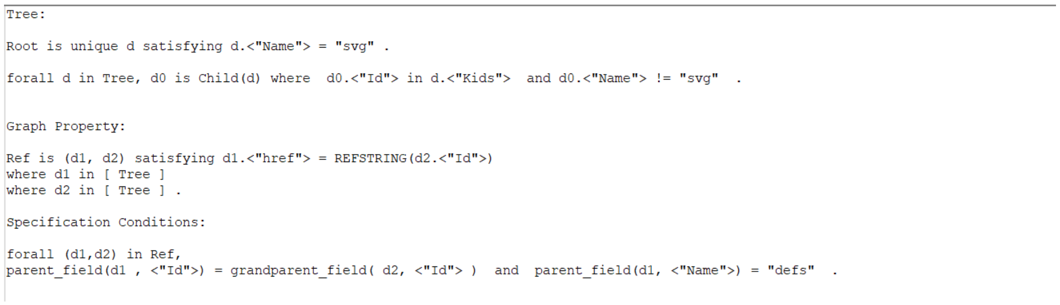
3 SVG properties.

3.1 'Title' is leftmost child of it's parent.



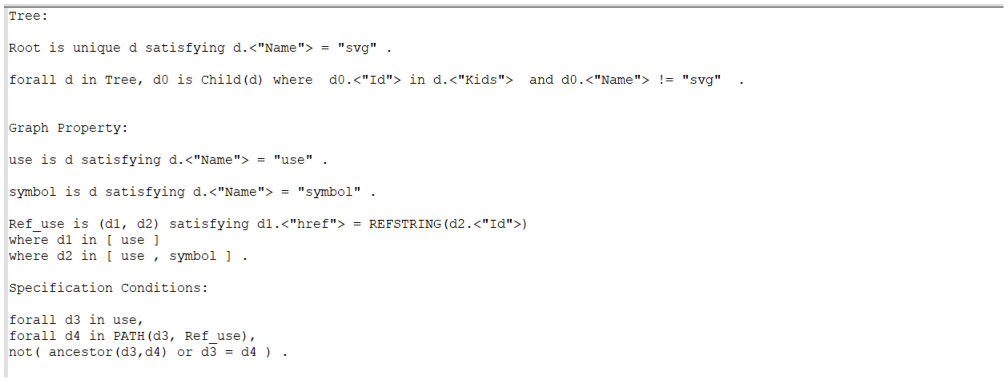
```
Ordered Tree:

Root is unique d satisfying d.<"Name"> = "svg" .

forall d in Ordered Tree, d0 is ith Child(d) where  d0.<"Id"> in d.<"Kids">  and  d0.<"Name"> !=  "svg"
where i is indexn (d0 , d.<"Kids">).


Ordered Tree Property:

Title is d satisfying d.<"Name"> = "Title" .

Specification Conditions:

forall d in Title,     ochild_field ( parent_field (d, <"Id">)  , 1 , <"Id"> ) = d.<"Id">.
```

3.2 All references inside 'defs'.



```
Tree:

Root is unique d satisfying d.<"Name"> = "svg" .

forall d in Tree, d0 is Child(d) where  d0.<"Id"> in d.<"Kids">  and d0.<"Name"> != "svg"  .


Graph Property:

Ref is (d1, d2) satisfying d1.<"href"> = REFSTRING(d2.<"Id">)
where d1 in [ Tree ]
where d2 in [ Tree ] .

Specification Conditions:

forall (d1,d2) in Ref,
parent_field(d1 , <"Id">) = grandparent_field( d2, <"Id"> )  and  parent_field(d1, <"Name">) = "defs"  .
```

3.3 No 'use'-'use' cycle.



```
Tree:

Root is unique d satisfying d.<"Name"> = "svg" .

forall d in Tree, d0 is Child(d) where  d0.<"Id"> in d.<"Kids">  and d0.<"Name"> != "svg"  .


Graph Property:

use is d satisfying d.<"Name"> = "use" .

symbol is d satisfying d.<"Name"> = "symbol" .

Ref_use is (d1, d2) satisfying d1.<"href"> = REFSTRING(d2.<"Id">)
where d1 in [ use ]
where d2 in [ use , symbol ] .

Specification Conditions:

forall d3 in use,
forall d4 in PATH(d3, Ref_use),
not( ancestor(d3,d4) or d3 = d4 ) .
```