Proving the Deutsch-Schorr-Waite Algorithm using Path Properties

Ronny Zakhejm
Semester Project

Supervised by Bernd Schoeller
Chair of Software Engineering
ETH Zürich

26th June 2005

Abstract

The goal of this semester project is to prove the correctness of the Deutsch-Schorr-Waite graph marking algorithm [SchorrWaite67] with a syntax-based method, i.e. the proof will be done with a hoare-style logic [Hoare69] based on Path Properties [Schoeller05]. Furthermore this proof should enable us rating the quality and the simple usage of Path Properties.

The Schorr-Waite algorithm is the first mountain that any formalism for pointer aliasing should climb — Richard Bornat

1 Introduction

The Deutsch-Schorr-Waite graph marking algorithm is a classical playground for people dealing with pointer aliasing and proving programm correctness, since it is a short, straightforward, non-recursive but nevertheless suprisingly powerful algorithm. Richard Bornat [Bornat00, page 121] even stated that proving its correctness is the first mountain that any formalism for pointer aliasing should climb, in order to show its power. Since Bernd Schoeller developed a new formalism for pointer aliasing based on his path properties, this formalism shall now climb that first mountain and show it's simplicity and expression power.

In [HubertMarché05] past approaches of proving the Deutsch-Schorr-Waite algorithm are listed and discussed.

The reader is assumed to be familiar with the Path Properties developed by Bernd Schoeller. For further details see [Schoeller05]. You can find there as well an overview and references over past approaches to formalising pointer aliasing.

The following work is very much inspired by Joseph M. Morris' proof of the Deutsch-Schorr-Deutsch algorithm [Morris82], where I have taken most of my definitions, rules and proof ideas from. Since the reader is as well assumed to

be familiar with the Deutsch-Schorr-Waite algorithm, the way it works and the terminology used (e.g. stack), the reader is again referred to [Morris82] for a comprehensive description of the algorithm.

2 The Deutsch-Schorr-Waite Algorithm

I have chosen the following implementation of the Deutsch-Schorr-Waite algorithm to work with:

```
{INIT}
/* SETUP */
Void.l = Void; Void.c = FALSE;
p := Void; t := root;
do
   until
      p = Void and t.m
   loop
      {INV}
      if not t.m then
         /* PUSH */
         q := p; p := t; t := t.l; p.l := q;
         p.m := TRUE; p.c := FALSE;
      else if t.m and not p.c
         /* SWING */
         q := t; t := p.r; p.r := p.l; p.l := q; p.c := TRUE;
      else
         /* POP */
         q := t; t := p; p := p.r; t.r := q;
      end
   end
end
Void.l = root;
{POST}
```

The node data type has the following properties:

```
Type Node: {l, r, l0, r0, k: Node; c, m: Boolean;}
```

The initial setting of all nodes is: $l \equiv l_0$, $r \equiv r_0$, m is FALSE and c doesn't matter. In order to be able to quantify over all nodes, a special property k is introduced. Starting at a special root symbol K, k links up all nodes into a linked list. With n chosen as $K.k^n \equiv Void$ and $Void.k \equiv Void$, this enables us to talk about all nodes introducing a set $AllNodes$ as follows:

$$AllNodes \quad =_{def} \quad \{x \mid x \equiv K.k^i,\ i \geq 0\}$$

A second very useful set is the set containing all nodes initially reachable from node *root* on a path using only l and r attributes. This set $AllRootReachable$ or simply ARR is defined as follows:

$$AllRootReachable = ARR \quad =_{def} \quad \{x \mid x \equiv root.\{l_0, r_0\}^i,\ i \geq 0\}$$

Note that $ARR \in AllNodes$.

We assume for simplicity reasons that no $\{r,l\}$-path from *root* reaches node $Void$, formally $\forall x \in ARR: \ x \not\equiv Void$. This assumption does not limit generality since we can always introduce a special void node *spezvoid*. $Void$ is thus $\in AllNodes$, but not in ARR. At the beginning $Void.l \equiv Void.r \equiv root$, $Void.c$ is set to (say) FALSE.

3 Connectivity

3.1 Connectivity Definitions

In order to be able to do the proof, we need some expressive predicates and rules. The main predicate involved is connectivity. Connectivity means that a certain node can be reached from an other specific node on a path using certain attributes. Three types of connectivity will be defined:

1. Connectivity by a single attribute, called l-connectivity
2. Connectivity by several attributes, called set connectivity
3. General connectivity

3.1.1 l-Connectivity

Definition 1 (l-Connectivity)

$$x \xrightarrow{l} y \quad =_{def} \quad \exists n \geq 0: \ (x.l^n \equiv y) \wedge (\forall a,b \leq n, \ a \neq b: \ x.l^a \not\equiv x.l^b) \quad (1)$$

Alternatively, we can define l-connectivity as follows

Definition 2 (l-Connectivity)

$$x \xrightarrow[n]{l} y \quad =_{def} \quad (x.l^n \equiv y) \quad (2)$$

$$x \xrightarrow{l} y \quad =_{def} \quad \exists n \geq 0: \ (x \xrightarrow[n]{l} y) \ \wedge \ (\forall a,b \leq n, \ a \neq b: \ x.l^a \not\equiv x.l^b) \quad (3)$$

In words: A path x is l-connected to path y means there exists an $n \geq 0$ for which $x.l^n \equiv y$, i.e. y is reachable from x by following only l-labeled attributes, and since we are looking for the smallest n, we may assume that there are no l- cycles. If there were, we could shortcut them and we'll get a smaller n. Now since we haven't got a cycle, we may assume $\forall a,b \leq n, \ a \neq b: \ x.l^a \not\equiv x.l^b$, means all Nodes on the l-path from x to y are distinct.

The negated form, means x is not l-connected to y, is defined as follows:

Definition 3 (Not Connected)

$$x \not\xrightarrow{l} y \quad =_{def} \quad \forall n \geq 0: \ (x.l^n \not\equiv y)$$

or

Definition 4 (Not Connected)

$$x \overset{l}{\not\longrightarrow}_{n} y \quad =_{def} \quad x.l^n \not\equiv y$$
$$x \overset{l}{\not\longrightarrow} y \quad =_{def} \quad \forall n \geq 0 : \ (x \overset{l}{\not\longrightarrow}_{n} y)$$

A star can be introduced to indicate an arbitrary non-negative integer:

Definition 5 (Not Connected, Starnotation)

$$x \overset{l}{\not\longrightarrow} y \quad =_{def} \quad x.l^* \not\equiv y$$

3.1.2 Set Connectivity

A generalised form of connectivity is a connectivity over a set K of attribute labels:

Definition 6 (Set Connectivity)

$$x \xrightarrow{K} y \quad =_{def} \quad \exists n \geq 0 : \ (x.K^n \equiv y) \wedge (\forall a, b \leq n, \ a \neq b : \ x.K^a \not\equiv x.K^b) \quad (4)$$

Again, there is an alternative definition:

Definition 7 (Set Connectivity)

$$x \xrightarrow[n]{K} y \quad =_{def} \quad (x.K^n \equiv y) \quad (5)$$
$$x \xrightarrow{K} y \quad =_{def} \quad \exists n \geq 0 : \ x \xrightarrow[n]{K} y) \ \wedge \ (\forall a, b \leq n, \ a \neq b : \ x.K^a \not\equiv x.K^b) \quad (6)$$

$x.K$ is a path from x along one attribute with a label from set K, $x.K^n$ is along n (not necessarily distinct) attribute labels from set K, while $x \xrightarrow{K} y$ means a path from x to y using attribute labels from the set K.

3.1.3 General Connectivity

Finally, general connectivity is defined as set connectivity with a set containing all attribute labels. For notational reasons, we'll use a special notation:

Definition 8 (General Connectivity)

$$x \xrightarrow[n]{*} y \quad =_{def} \quad x \xrightarrow[n]{K} y \quad \textit{with } K \textit{ containing all labels} \quad (7)$$
$$x \xrightarrow{*} y \quad =_{def} \quad \exists n \geq 0 : \ (x \xrightarrow[n]{*} y) \ \wedge \ (\forall a, b \leq n, \ a \neq b : \ x.*^a \not\equiv x.*^b) \quad (8)$$
$$x \longrightarrow y \quad =_{def} \quad x \xrightarrow{*} y \quad (9)$$

$x.*$ means a path from x following any attribute of x.

3.2 Rules for Connectivity

In order to be able to work with connectivity, we need to state and prove some rules concerning connectivity.

Rule 1

$$\{x \xrightarrow{l} y \wedge\ q \not\succ x.l^n \wedge\ q \not\succ y\}$$
$$\text{q} := \text{t};$$
$$\{x \xrightarrow{l} y\}$$

with n in $q \not\succ x.l^n$ equals the n in $\exists n \geq 0$... in the definition of $x \xrightarrow{l} y$, i.e. the pathlength of $x \xrightarrow{l} y$.

Proof: This rule can be directly derived from the definition of connectivity and from the axiom 4.1a.

The same rule applies to $x \not\xrightarrow{l} y$ (with $q \not\succ x.l^*$ instead).

Rule 2

$$\{x \xrightarrow{l} y\ \wedge\ y.l \not\succ y \wedge\ y.l \not\succ x\}$$
$$\text{y.l} := \text{t};$$
$$\{x \xrightarrow{l} y\}$$

Proof: It has to be shown, that $y.l \not\succ x.l^n$ with n the pathlength of $x \xrightarrow{l} y$ hold in the precondition. $y.l \not\succ x.l^n \triangleq (y.l \not\succ x.l^n) \wedge (y.l \not\succ x.l^{n-1}) \wedge ... \wedge (y.l \not\succ x.l) \wedge (y.l \not\succ x)$. The last term is already in the precondition. The other terms are $y.l \not\succ x.l^i$ for $1 \leq i \leq n$, so we have to show that $y \not\equiv x.l^i$ for $0 \leq i \leq n-1$. This follows from $x \xrightarrow{l} y \implies y \equiv x.l^n$ and $\forall a, b \leq n,\ a \neq b :\ x.l^a \not\equiv x.l^b$.

Rule 3

$$x \xrightarrow{l} y \iff x \equiv y \vee\ x.l \xrightarrow{l} y$$

Proof: " $\Rightarrow$ ": $x \xrightarrow{l} y \implies \exists n \geq 0 :\ (x.l^n \equiv y)$. If $n = 0$, so $x \equiv y$. Else, if $n > 0$, $x.l.l^{n-1} \equiv y$ which means that $x.l \xrightarrow[n-1]{l} y$. Hence $x.l \xrightarrow{l} y$.

" $\Leftarrow$ ": If $x \equiv y$, then $x.l^0 \equiv y$, thus $\exists n \geq 0 :\ (x.l^n \equiv y)$ hence $x \xrightarrow{l} y$. If $x.l \xrightarrow{l} y$, this means that $\exists n \geq 0 :\ (x.l.l^n \equiv y)$, therefore $\exists n \geq 0 :\ x.l^n \equiv y$, hence x is l-connected to y. If this path isn't the shortest possible, i.e. it has got cycles in it, so simply shortcut the cycle. Thus $x \xrightarrow{l} y$.

Note that the connection path in $x \xrightarrow{l} y$ might be shorter than in $x.l \xrightarrow{l} y$, exactly then when $x \equiv y$.

Rule 4

$$\{x.l \xrightarrow{l} y\ \wedge\ x.l \not\succ y\ \wedge\ x.l \not\succ q\ \wedge\ x.l \equiv q\}$$
$$\text{x.l} := \text{t};$$
$$\{q \xrightarrow{l} y\}$$

Proof: $x.l \xrightarrow{l} y$ means that $\exists n \geq 0 : (x.l.l^n \equiv y) \wedge (\forall a, b \leq n,\ a \neq b : x.l.l^a \not\equiv x.l.l^b)$, and from $x.l \equiv q$ follows that $(q.l^n \equiv y) \wedge (\forall a, b \leq n,\ a \neq b : q.l^a \not\equiv q.l^b)$ for that n, hence $q \xrightarrow{l} y$ with pathlength n in the Precondition. If we are able to derive $x.l \not\succ q.l^n$ from the precondition, we are done, since we can apply rule 1. $x.l \not\succ q.l^n \triangleq (x.l \not\succ q.l^n) \wedge (x.l \not\succ q.l^{n-1}) \wedge ... \wedge (x.l \not\succ q.l) \wedge (x.l \not\succ q)$. The last term is already part of the precondition. The others require use to prove $x \not\equiv q.l^i,\ 0 \leq i < n$. We'll split the proof into two parts: $x \equiv y$ and $x \not\equiv y$.

First $x \equiv y$: Since $q \xrightarrow{l} y$, we know that $\forall a, b \leq n,\ a \neq b : q.l^a \not\equiv q.l^b$, specially $\forall i \leq n,\ i \neq n : q.l^i \not\equiv q.l^n \equiv y \equiv x$. Hence $x \not\equiv q.l^i,\ 0 \leq i < n$.

If $x \not\equiv y$, we know from $\exists n \geq 0 : (x.l.l^n \equiv y) \wedge (\forall a, b \leq n,\ a \neq b : x.l.l^a \not\equiv x.l.l^b)$ that $(x.l^{n+1} \equiv y) \wedge (\forall a, b \leq n+1,\ a \neq b : x.l^a \not\equiv x.l^b)$ for that n. Since $x.l \equiv q$, we know that $\forall i \leq n : x \not\equiv x.l.l^i \equiv q.l^i$, hence $x \not\equiv q.l^i,\ 0 \leq i < n$.

Rule 5

$$\begin{aligned} x \xrightarrow{l} y \quad \Longleftrightarrow \quad & \forall z : (x \xrightarrow{l} y \ \wedge\ (\exists q : z.l \not\succ q.l^n \ \wedge\ x \equiv q)) \\ & \vee (x \xrightarrow[a]{l} z \ \wedge\ z.l \xrightarrow[b]{l} y \ \wedge\ (\exists p : z.l \not\succ p.l^a \ \wedge\ p \equiv x) \\ & \wedge (\exists q : x.l \not\succ q.l^b \ \wedge\ z.l \equiv q)) \end{aligned}$$

with n in $z.l \not\succ q.l^n$ equals the pathlength of $x \xrightarrow{l} y$.

Proof: " $\Rightarrow$ ": We denote the length of $x \xrightarrow{l} y$ by n. If z is on the connection path of $x \xrightarrow{l} y$ and $z \not\equiv y$, i.e. $\exists a < n : (x.l^n \equiv y) \wedge (x.l^a \equiv z) \wedge (\forall c, b \leq n,\ c \neq b : x.l^c \not\equiv x.l^b)$, then $x \xrightarrow[a]{l} z \ \wedge\ z.l \xrightarrow[n-a-1]{l} y$, hence $x \xrightarrow{l} z \ \wedge\ z.l \xrightarrow{l} y$. $\exists p : z.l \not\succ p.l^a \wedge p \equiv x$ follows from the proof of rule 2 since $z.l \not\succ p$ as we choose p to be a root, $\exists q : z.l \not\succ q.l^n \ \wedge\ z.l \equiv q$ follows from the proof of rule 4 with q as root (since $z.l \not\succ q$). If on the other hand z is not on the connection path, i.e. $\forall i \leq n : x.l^i \not\equiv z$, so $\exists q : z.l \not\succ q.l^n \ \wedge\ x \equiv q$ holds due to $z.l \not\succ q$ which is surely true in the case of q a root (and we can always create a new root q) and on account of z not occuring on the connection path. Finally if $z \equiv y$, so the first clause is true, derived from the proof of rule 2 with q a root.

" $\Leftarrow$ ": If the first clause holds: trivial. If the second clause holds: Since $x.l^a \equiv z$ and $z.l.l^b \equiv y$, we can derive that $x.l^a.l.l^b = x.l^{a+b+1} \equiv y$. If this new connection path contains a cycle, we can shortcut it and get a cyclefree connection path from x to y. Hence $x \xrightarrow{l} y$.

Rule 6

$$x \xrightarrow{l} y \ \wedge\ y \not\xrightarrow{l} x \quad \Longrightarrow \quad x.l \not\xrightarrow{l} x$$

Proof: According to the definition of connecticity a smallest n must exist for which $x \xrightarrow[n]{l} y$. Since $y \not\xrightarrow{l} x$, we know that this $n \neq 0$, otherwise $y \equiv x$ and $y \xrightarrow{l} x$ according to rule 3. Therefore $x.l.l^{n-1} \equiv y$, which means, that $x.l \xrightarrow[n-1]{l} y$. If $x.l$ were l-connected to x, this path couldn't go through y, since y is not l-connected to x. Furthermore x in not on the l-path from $x.l$ to y, since then the x-to-y-path would have a length shorter that $n-1$, which contradicts $x \xrightarrow[n]{l} y$. Hence $x.l \not\xrightarrow{l} x$.

Rule 7

$$x \overset{l}{\not\longrightarrow} y \ \wedge \ y.l \not\succ x \quad \Longrightarrow \quad y.l \not\succ x.l^*$$

Proof: Since $x \overset{l}{\not\longrightarrow} y$, we know, that $x.l^i \not\equiv y$ for all $i \geq 0$. If $y.l \succ x.l^a$ for some a, this means, that according to A3.3b $y.l \asymp x.l^b$ for at least one $0 < b \leq a$ or $y.l \succ x$. The opposit of the second term is explicitly assumed, and the first one infers that $y \equiv x.l^{b-1}$ for this b, thus contradicting $x.l^i \not\equiv y$. Hence $y.l \not\succ x.l^*$.

This list of rules is surely not complete, but for the proof it is sufficient.

4 Conditions and Invariants

Let's start the proof. We need now to find the precondition, the desired postcondition and a couple of invariants, expressive enough to derive the postcondition from.

4.1 The Precondition

Initially all subscript-0 properties are equal to their not subscripted correspondants and the property m is set to FALSE. Additionally we know that $Void$ is connected to root.

Hence

$$\begin{aligned} INITCond(x) &=_{def} x.l \equiv x.l_0 \ \wedge \ x.r \equiv x.r_0 \ \wedge \ \overline{x.m} \\ INIT &=_{def} \forall x \in AllNodes : \ INITCond(x) \ \wedge \ Void.l_0 \equiv root \end{aligned}$$

4.2 The Postcondition

In the postcondition state we must assure two things, first that the graph structure is again as it was before the algorithm, and second that the algorithm did what it should do, namely marking all nodes reachable from root and leaving all other nodes unmarked.

Therefore

$$\begin{aligned} POST \quad =_{def} \quad & \forall x \in AllNodes : \ x.l \equiv x.l_0 \ \wedge \ x.r \equiv x.r_0 \\ & \wedge \forall x \in ARR : \ x.m \ \wedge \ \forall x \notin ARR : \ \overline{x.m} \end{aligned}$$

4.3 The Loop Invariant

4.3.1 s-Notation

For an easy notation and efficient way to write down the loop invariant and the proof, we still need one notational convention, namely the special property s:

$$\begin{aligned} x.s &\triangleq \text{if } x.c \text{ then } x.r \text{ else } x.l \\ x.\overline{s} &\triangleq \text{if } x.c \text{ then } x.l \text{ else } x.r \\ x.s_0 &\triangleq \text{if } x.c \text{ then } x.r_0 \text{ else } x.l_0 \\ x.\overline{s_0} &\triangleq \text{if } x.c \text{ then } x.l_0 \text{ else } x.r_0 \end{aligned}$$

Note that s_0 is depending on the current c, not on some old (not existing) c_0.

An alternative way of defining this notational convention is by means of a function s:

$$s(x) = \begin{cases} x.r & \text{if } x.c \\ x.l & \text{if } \overline{x.c} \end{cases}$$

$\overline{s}(x)$, $s_0(x)$ and $\overline{s_0}(x)$ will be defined in the same way. The functional definition might be mathematically more correct and cause less ambiguities. On the other hand, the property-like definition fits more smoothly into our framework, but care must be taken that s is not a real property, since s could be influenced by s, l, r or c, but not by m or other properties.

The two properties of the algorithm, i.e. that the graph remains unchanged on termination and that all reachable nodes are marked and the others not, leed to the following two groups of Invariants. Let us call the first group the structural invariants and the second the functional.

4.3.2 The Structural Invariants

Invariant P1

Invariant P1 states that a node initially reachable from *root* is at any state either on the stack (for a description of the stack see [Morris82]) or its link properties are unchanged.

Invariant 1 (P1)

$$\begin{aligned} P1 &=_{def} \forall x \in ARR: \ OnStack(x) \vee NotOnStack(x) \\ OnStack(x) &=_{def} p \xrightarrow{s} x \wedge x.\overline{s} \equiv x.\overline{s_0} \wedge x.s.s_0 \equiv x \wedge x.m \\ NotOnStack(x) &=_{def} p \not\xrightarrow{s} x \wedge x.l \equiv x.l_0 \wedge x.r \equiv x.r_0 \end{aligned}$$

Note that $NotOnStack(p)$ is always false ($OnStack$ might be false either, $\Longleftrightarrow p \equiv Void$).

On the other hand, $NotOnStack(t)$ is true if $p \not\equiv t$.

Proof with contradiction: If $OnStack(t)$ is true, $p \xrightarrow{s} t \wedge t.s.s_0 \equiv t$ hold. Since P2 is always true, $p.s_0 \equiv t$ and hence $t.s \equiv p \implies t \xrightarrow{s} p$. As $p \xrightarrow{s} Void$ (P3), $p \xrightarrow{s} t$, $p \not\equiv t$, $Void$ is a 'final station' and connectivity is not circular, we get a contradiction (apply rule 6). Hence $NotOnStack(t)$ is true if $t \not\equiv p \wedge t \not\equiv Void$.

Invariant P2

Invariant P2 fills the connection 'hole' opened up by the algorithm, i.e. p was originally connected to t by s.

Invariant 2 (P2)

$$p.s_0 \equiv t$$

Invariant P3

Invariant P3 assures that p is on the stack. It is actually the top of the stack.

Invariant 3 (P3)

$$p \xrightarrow{s} Void$$

4.3.3 The Functional Invariant

Invariant P4

The functional invariant P4 divides all nodes in three (not necessarily distinct) parts:

1. The nodes not marked yet
2. The nodes on the stack, which are being proceeded currently by the algorithm
3. The nodes which are already done

Invariant 4 (P4)

$$\forall x \in AllNodes : \ \overline{x.m} \ \vee \ p \xrightarrow{s} x \ \vee \ (x.l.m \ \wedge \ x.r.m)$$

Invariant P5

Looking at invariant P4, a node on the stack encounters two 'events' (i.e $x.l.m := TRUE$ & $x.r.m := TRUE$) before it satisfies $x.l.m \ \wedge \ x.r.m$. It's quite hard to prove directly for a node that these two 'events' indeed happend. So it's useful to introduce an auxiliary invariant P5, which captures the first event, namely $x.l.m := TRUE$.

Invariant 5 (P5)

$$\forall x \in AllNodes : \ \overline{x.m} \ \vee \overline{x.c} \ \vee \ x.l.m$$

5 The Proof

In order to prove the algorithm we must show the following things:

1. From the precondition we should be able to prove the correctness of all invariants after the setup
2. From the loop invariants and the loop condition we should derive the postcondition
3. The invariants hold during each iteration of the loop

Since the loop contains three disjunct parts, we can devide the third proof into three subproofs, namely:

1. The invariants hold with respect to PUSH

2. The invariants hold with respect to SWING
3. The invariants hold with respect to POP

As a notational convention we will use the following abbrevations:

- A4.1a, A3.3a, ... for the axioms in [SchoellerSummary].
- R1, R2, ... for the connectivity rules.
- P1, P2, ... for the Loop Invariants
- PE for the path extension property
- TR for the transitivity of the equality relation
- DEF for per definition (normally of l-connectivity or of s)
- NC for non circularity
- LC for the loop condition. Normally we derive from the loop condition and a if guard of $t.m$ that $p \not\equiv Void$.
- h_0, h_1, or sometimes simply h denote added root variables. Sometimes used as h_* in combination with $x.*$ as $x.* \equiv h_*$ to denote that the path x dot any attribute equals a certain added root variable.

Trivial proof steps as inserting non-influence of roots and similar are usually omitted. In Section 5.3.3 a full proof is given as an example.

5.1 The Loop Invariants hold after SETUP

```
{INIT}
/* SETUP */
Void.l = Void; Void.c = FALSE;
p := Void; t := root;
```

$$\begin{aligned} INITCond(x) &=_{def} x.l \equiv x.l_0 \wedge x.r \equiv x.r_0 \wedge \overline{x.m} \\ INIT &=_{def} \forall x \in AllNodes: \ INITCond(x) \wedge Void.l_0 \equiv root \end{aligned}$$

$\{INIT\}$
Void.l := Void; Void.c = FALSE;
$\{\forall x \in (AllNodes/Void): \ INITCond(x) \wedge Void.l \equiv Void \wedge \overline{Void.c} \wedge Void.l_0 \equiv root\}$
p := Void; t := root;
$\{\forall x \in (AllNodes/Void): \ INITCond(x) \wedge p.l \equiv p \wedge \overline{p.c} \wedge p \equiv Void \wedge t \equiv root \wedge p.l_0 \equiv root\}$
$\{\forall x \in (AllNodes/Void): \ INITCond(x) \wedge p.s \equiv p \wedge p \equiv Void \wedge t \equiv root \wedge p.s_0 \equiv root\}$
$\{\forall x \in (AllNodes/Void): \ x.l \equiv x.l_0 \wedge x.r \equiv x.r_0 \wedge \overline{x.m}$
$\wedge \ p.s \equiv p \wedge p \equiv Void \wedge t \equiv root \wedge p.s_0 \equiv root\}$ (10)

5.1.1 P1

Since $ARR \in AllNodes$ and $Void \notin ARR$, $\forall x \in ARR: \ x.l \equiv x.l_0 \ \wedge \ x.r \equiv x.r_0$ is true. $p \stackrel{s}{\not\longrightarrow} x$ is derived from the fact that $p \equiv Void$ and $p.s \equiv p$. Hence P1 holds.

Actually I would like to include $Void$ into P1, with $Void$ the only node for which $OnStack(x)$ is true after setup. $Void.m$ is no problem, since it could be set explicitly in the setup part, and set to FALSE after the loop terminates. But $x.s.s_0 \equiv x$ isn't true with $Void.s \equiv Void$. Again a new Node must be introduced (e.g. $VoidVoid$), on which $Void.s$ points, which would lead to other problems. So $Void$ has got a special position, sometimes included, sometimes excluded, and its quite tedious to keep track of this problem. Hopefully not, but the proof might be therefore quite inaccurate concerning $Void$.

Invariant P3 is somehow a little fix for this problem. It is actually a special case of P1.

5.1.2 P2

P2 appears explicitly in (10).

5.1.3 P3

Since $p \equiv Void$ in (10), P3 is derived by rule 3.

5.1.4 P4 and P5

In (10) for all nodes except p $\overline{x.m}$ holds. As $p \equiv Void \Rightarrow p \stackrel{s}{\longrightarrow} Void$ by rule 3 and $Void.c$ is set to $FALSE$ during SETUP, P4 and P5 hold for all nodes. So all Invariants hold upon setup !

5.2 The Postcondition holds at Loop Termination

```
{INV and p = Void and t.m}
Void.l = root;
{POST}
```

$$\begin{aligned} POST \quad =_{def} \quad & \forall x \in AllNodes: \ x.l \equiv x.l_0 \ \wedge \ x.r \equiv x.r_0 \\ & \wedge \forall x \in ARR: \ x.m \ \wedge \ \forall x \notin ARR: \ \overline{x.m} \end{aligned}$$

$$\{P1 \ \wedge \ P2 \ \wedge \ P3 \ \wedge \ P4 \ \wedge \ p \equiv Void \ \wedge \ t.m\} \tag{11}$$

P2 $\{P1 \ \wedge \ P2 \ \wedge \ P3 \ \wedge \ P4 \ \wedge \ p \equiv Void \ \wedge \ root.m\}$

Void.l := root;

$$\{P1 \ \wedge \ P2 \ \wedge \ P3 \ \wedge \ P4 \ \wedge \ p \equiv Void \ \wedge \ root.m \ \wedge \ Void.l \equiv root\} \tag{12}$$

Note: I'm not happy with the following proof. I'd like to find a nicer one, specially concerning the part that $Void$ didn't change during the algorithm.

Since at the beginning of the first loop iteration $\overline{t.m}$ and therefore POP is executed, and in POP p is moved on and $t.m$ gets TRUE, the next time any root $\equiv Void$, $t.m$ is TRUE, p is that root pointing on $Void$ and the loop terminates. Since after SETUP $Void.s \equiv Void$, we can therefore assume that from $Void$ we cannot reach any other node on s-pathes, i.e. $\forall x \in AllNodes : \ Void \stackrel{s}{\not\longrightarrow} x$. Therefore in P1 in (11) $\forall x \in ARR : \ NotOnStack(x)$ is true since $p \equiv Void$, hence $x.l \equiv x.l_0 \wedge x.r \equiv x.r_0$. For $Void$ this is true too, since $Void.r$ was never affected and $Void.l \equiv Void.l_0 \equiv root$ at (12). All other nodes couldn't have been affected, since they are not reachable neither by $root$ nor by $Void$, which are the only two roots involved in the algorithm (except for TRUE and FALSE, which are constant).

Since on loop termination $root.m \wedge p \stackrel{s}{\longrightarrow} root$ (from P1), $t.l.m \wedge t.r.m$ must be true according to P4. Taking the transitive closure with P4, all nodes reachable from $root$ have their m-property set to TRUE. Again, we know that $Void.m \equiv FALSE$. All other nodes could not have been affected, so their m-property remained FALSE.

Hence $POST$ is true on termination !

5.3 The Invariants with Respect to PUSH

```
/* PUSH */
{not t.m}
q := p; p := t; t := t.l; p.l := q;
p.m := TRUE; p.c := FALSE;
```

5.3.1 P1

$$
\begin{aligned}
P1 &=_{def} \ \forall x \in ARR : \ OnStack(x) \ \vee \ NotOnStack(x) \\
OnStack(x) &=_{def} \ p \stackrel{s}{\longrightarrow} x \ \wedge \ x.\overline{s} \equiv x.\overline{s_0} \ \wedge \ x.s.s_0 \equiv x \ \wedge \ x.m \\
NotOnStack(x) &=_{def} \ p \stackrel{s}{\not\longrightarrow} x \ \wedge \ x.l \equiv x.l_0 \ \wedge \ x.r \equiv x.r_0
\end{aligned}
$$

P1 must be proven for all nodes x. Lets do two groups: $x \not\equiv t$ and $x \equiv t$.
For $x \equiv t$:

$$
\begin{array}{rl}
 & If\ Guard\{P1 \wedge x \equiv t \wedge \overline{t.m}\} \\
P1 & \{x \equiv t \wedge \overline{t.m} \wedge t.l \equiv t.l_0 \wedge t.r \equiv t.r_0 \wedge p \not\equiv t\} \\
P2 & \{x \equiv t \wedge t.r \equiv t.r_0 \wedge p.s_0 \equiv t \wedge p \not\equiv t\} \\
 & \text{q := p :} \\
A4.2a\ \&\ A4.1a & \{x \equiv t \wedge t.r \equiv t.r_0 \wedge p.s_0 \equiv t \wedge q \equiv p \wedge p \not\equiv t\} \\
 & \{x \equiv t \wedge t.r \equiv t.r_0 \wedge q.s_0 \equiv t \wedge q \not\equiv t\} \\
 & \text{p := t;} \\
A4.2a\ \&\ A4.1a & \{x \equiv t \wedge t.r \equiv t.r_0 \wedge q.s_0 \equiv t \wedge p \equiv t \wedge q \not\equiv t\} \\
 & \{x \equiv p \wedge p.r \equiv p.r_0 \wedge q.s_0 \equiv p \wedge q \not\equiv p\} \\
 & \text{t := t.l;} \\
A4.1a & \{x \equiv p \wedge p.r \equiv p.r_0 \wedge q.s_0 \equiv p \wedge q \not\equiv p\} \\
 & \text{p.l := q;} \\
A4.2b\ \&\ A4.1a & \{x \equiv p \wedge p.l \equiv q \wedge p.r \equiv p.r_0 \wedge q.s_0 \equiv p \wedge q \not\equiv p\} \\
 & \text{p.m := TRUE;} \\
A4.2b\ \&\ A4.1a & \{x \equiv p \wedge p.m \wedge p.l \equiv q \wedge p.r \equiv p.r_0 \wedge q.s_0 \equiv p \wedge q \not\equiv p\} \\
 & \text{p.c := FALSE;} \\
A4.2b\ \&\ A4.1a & \{x \equiv p \wedge p.m \wedge p.l \equiv q \wedge p.r \equiv p.r_0 \wedge q.s_0 \equiv p \wedge \overline{p.c}\} \\
DEF\ s & \{x \equiv p \wedge p.m \wedge p.s \equiv q \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge q.s_0 \equiv p\} \\
R3\ \&\ TR & \{p \xrightarrow{s} x \wedge x.m \wedge x.s \equiv q \wedge x.\overline{s} \equiv x.\overline{s_0} \wedge q.s_0 \equiv x\} \\
PE & \{p \xrightarrow{s} x \wedge x.m \wedge x.s.s_0 \equiv q.s_0 \wedge x.\overline{s} \equiv x.\overline{s_0} \wedge q.s_0 \equiv x\} \\
TR & \{p \xrightarrow{s} x \wedge x.m \wedge x.s.s_0 \equiv x \wedge x.\overline{s} \equiv p.\overline{s_0}\} \\
 & \{P1\}
\end{array}
$$

For $x \not\equiv t$ we know, that no property of x is affected by PUSH: ($x.*$ denotes any property of x)

$\{x \not\equiv t \ \wedge \ x.* \equiv h_*\}$
q := p :
$\{x \not\equiv t \ \wedge \ x.* \equiv h_*\}$
p := t;
$\{p \equiv t \ \wedge \ x \not\equiv t \ \wedge \ x.* \equiv h_*\}$
$\{x \not\equiv p \ \wedge \ x.* \equiv h_*\}$
t := t.l;
$\{x \not\equiv p \ \wedge \ x.* \equiv h_*\}$
p.l := q;
$\{x \not\equiv p \ \wedge \ x.* \equiv h_*\}$
p.m := TRUE;
$\{x \not\equiv p \ \wedge \ x.* \equiv h_*\}$
p.c := FALSE;
$\{x \not\equiv p \ \wedge \ x.* \equiv h_*\}$

So we still need to prove that the connectivity of p to x does not get changed.
If $p \xrightarrow{s} x$, see the proof of P3 and substitute $Void$ by x.
For $p \not\xrightarrow{s} x$:

$$
\begin{array}{rl}
If\ Guard\{P1 \wedge x \not\equiv t \wedge p \overset{s}{\not\longrightarrow} x \wedge \overline{t.m}\} & \\
P1 & \{x \not\equiv t \wedge p \overset{s}{\not\longrightarrow} x \wedge p \overset{s}{\not\longrightarrow} t\} \\
A3.3d & \{x \not\equiv t \wedge p \overset{s}{\not\longrightarrow} x \wedge p \overset{s}{\not\longrightarrow} t \wedge q \not\succ p.s^*\} \\
 & \text{q := p :} \\
R1,\ A4.2a,\ A4.1a & \{x \not\equiv t \wedge p \overset{s}{\not\longrightarrow} x \wedge p \overset{s}{\not\longrightarrow} t \wedge q \equiv p\} \\
PE\ \&\ A3.3d & \{x \not\equiv t \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} t \wedge p \not\succ q.s^*\} \\
 & \text{p := t;} \\
R1,\ A4.2a,\ A4.1a & \{x \not\equiv t \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} t \wedge p \equiv t\} \\
PE\ \&\ A3.3d & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv t.l \wedge t \not\succ q.s^*\} \\
 & \text{t := t.l;} \\
R1,\ A4.2a,\ A4.1a & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv t\} \\
R7\ \&\ DEF\ s & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv t \wedge p.l \not\succ q.s^*\} \\
 & \text{p.l := q;} \\
R1,\ A4.2b,\ A4.1a & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q\} \\
A3.3d & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q \wedge p.m \not\succ q.s^*\} \\
 & \text{p.m := TRUE;} \\
R1,\ A4.1a & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q\} \\
R7\ \&\ DEF\ s & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q \wedge p.c \not\succ q.s^*\} \\
 & \text{p.c := FALSE;} \\
R1,\ A4.1a,\ A4.2b & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge p.l \equiv q \wedge p.c \equiv FALSE\} \\
DEF\ s & \{x \not\equiv p \wedge q \overset{s}{\not\longrightarrow} x \wedge p.s \equiv q\} \\
PE & \{x \not\equiv p \wedge p.s \overset{s}{\not\longrightarrow} x\} \\
R3 & \{p \overset{s}{\not\longrightarrow} x\} \qquad (13) \\
 & \{P1\} \qquad \blacksquare
\end{array}
$$

Explanation for the two places where we apply R7 (although $c \neq l$ respectivly $c \neq s$) is given in the proof of P3. (13) is derived from the negated form of R3.

5.3.2 P2

$$P2: \qquad p.s_0 \equiv t$$

	$If\ Guard\{P1 \wedge \overline{t.m}\}$
$NotOnStack(t)$	$\{t.l \equiv t.l_0\}$
	q := p :
$A4.1a$	$\{t.l \equiv t.l_0\}$
	p := t;
$A4.1a$ & PE	$\{p.l \equiv p.l_0 \equiv t.l\}$
	t := t.l;
$A4.2a$	$\{p.l \equiv p.l_0 \equiv t\}$
	$\{p.l_0 \equiv t\}$
	p.l := q;
$A4.1a$	$\{p.l_0 \equiv t\}$
	p.m := TRUE;
$A4.1a$	$\{p.l_0 \equiv t\}$
	p.c := FALSE;
$A4.1a$ & $A4.2a$	$\{p.c \equiv FALSE \wedge p.l_0 \equiv t\}$
$DEF\ s$	$\{p.s_0 \equiv t\}$
	$\{P2\}$ ■

5.3.3 P3

$$P3: \qquad p \xrightarrow{s} Void$$

$$
\begin{array}{rl}
If\ Guard\{P3 \wedge \overline{t.m}\} & \\
P1 & \{p \xrightarrow{s} Void \wedge p \overset{s}{\not\longrightarrow} t\} \\
A3.3d & \{p \xrightarrow{s} Void \wedge p \overset{s}{\not\longrightarrow} t \wedge q \not\succ p.s^* \wedge q \not\succ t \wedge q \not\succ Void\} \\
 & \text{q := p :} \\
R1\ \&\ A4.2a & \{p \xrightarrow{s} Void \wedge p \overset{s}{\not\longrightarrow} t \wedge q \equiv p\} \\
PE\ \&\ A3.3d & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} t \wedge p \not\succ q.s^* \wedge p \not\succ t \wedge p \not\succ Void\} \\
 & \text{p := t;} \\
R1\ \&\ A4.2a & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} t \wedge p \equiv t\} \\
PE\ \&\ A3.3d & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv t.l \wedge t \not\succ q.s^* \wedge t \not\succ Void \wedge t \not\succ p.l\} \\
 & \text{t := t.l;} \\
R1\ \&\ A4.2a & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv t\} \\
R7\ \&\ A3.3d & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv t \wedge p.l \not\succ q.s^* \wedge p.l \not\succ Void \wedge p.l \not\succ p\} \\
 & \qquad\qquad (14) \\
 & \text{p.l := q;} \\
R1\ \&\ A4.2a & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q\} \\
A3.3d & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q \wedge p.m \not\succ q.s^* \wedge p.m \not\succ Void \wedge p.m \not\succ p.l\} \\
 & \text{p.m := TRUE;} \\
R1\ \&\ A4.1a & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q\} \\
A3.3d,\ R7 & \{q \xrightarrow{s} Void \wedge q \overset{s}{\not\longrightarrow} p \wedge p.l \equiv q \wedge p.c \not\succ q.s^* \wedge p.c \not\succ Void \wedge p.c \not\succ p.l\} \\
 & \qquad\qquad (15) \\
 & \text{p.c := FALSE;} \\
R1,\ A4.1b,\ A4.2a & \{q \xrightarrow{s} Void \wedge p.l \equiv q \wedge p.c \equiv FALSE\} \\
DEF\ s & \{q \xrightarrow{s} Void \wedge p.s \equiv q\} \\
PE & \{p.s \xrightarrow{s} Void\} \\
R3 & \{p \xrightarrow{s} Void\} \\
 & \{P3\} \qquad \blacksquare
\end{array}
$$

(14) and (15) need some explanation, since we apply rule 7 although the attributes are not equal ($c \neq l$ respectivly $c \neq s$), so let's revisit the proof of rule 6 and fit it to our situation: since $q \overset{s}{\not\longrightarrow} p$, we know that $q.s^i \not\equiv p\ \forall i \geq 0$. If $p.l \succ q.s^a$ for some a, this means, that according to A3.3b $p.l \asymp q.s^b$ for at least one $0 < b \leq a$ or that $p.l \succ q$. The second term is definitely false. The first one infers ($p.c \equiv FALSE$ provided, if not, $p.l \not\succ q.s^b$ since they haven't got the same attribute) that $p \equiv q.s^{b-1}$ for this b, thus contradicting $q.s^i \not\equiv p$. Hence $p.l \not\succ q.s^*$. The same argumentation can be applied to $p.c$ instead of $p.l$: If $p.c \succ q.s^a$ for some a, this means, that according to A3.3b $p.c \asymp q.s^b$ for at least

one $0 < b \leq a$ or that $p.c \succ q$. The second term is again definitely false. The first one requires $p \equiv q.s^{b-1}$ for this b to be true, thus contradicting $q.s^i \not\equiv p$. Hence $p.c \succ q.s^*$.

5.3.4 P4

$$P4: \qquad \forall x \in AllNodes: \ \overline{x.m} \ \vee \ p \xrightarrow{s} x \ \vee \ (x.l.m \ \wedge \ x.r.m)$$

P4 must be proven for all nodes x. Lets do two groups: $x \not\equiv t$ and $x \equiv t$.

For $x \equiv t$ P4 is trivially satified since t before PUSH equals p after the loop. Hence $p \xrightarrow{s} p.$ is trivial.

$\{x \equiv t\}$
q := p :
$\{x \equiv t\}$
p := t;
$\{x \equiv t \ \wedge \ p \equiv t\}$
$\{x \equiv p\}$
t := t.l;
$\{x \equiv p\}$
p.l := q;
$\{x \equiv p\}$
p.m := TRUE;
$\{x \equiv p\}$
p.c := FALSE;
$\{x \equiv p\}$

For $x \not\equiv t$:

- All nodes satisfying $\overline{x.m}$ will still satisfy it since no m-property other than

for $x \equiv t$ is affected in PUSH.

$\{x \not\equiv t \wedge \overline{x.m}\}$
q := p :
$\{x \not\equiv t \wedge \overline{x.m}\}$
p := t;
$\{x \not\equiv t \wedge \overline{x.m} \wedge p \equiv t\}$
$\{x \not\equiv p \wedge \overline{x.m}\}$
t := t.l;
$\{x \not\equiv p \wedge \overline{x.m}\}$
p.l := q;
$\{x \not\equiv p \wedge \overline{x.m}\}$
p.m := TRUE;
$\{x \not\equiv p \wedge \overline{x.m}\}$
p.c := FALSE;
$\{x \not\equiv p \wedge \overline{x.m}\}$
$\{\overline{x.m}\}$

- All nodes satisfying $p \xrightarrow{s} x$, see the proof of P3 and substitute $Void$ by x.
- All nodes satisfying $x.l.m \wedge x.r.m$:

$\{x \not\equiv t \wedge x.l.m \wedge x.r.m\}$
q := p :
$\{x \not\equiv t \wedge x.l.m \wedge x.r.m\}$
p := t;
$\{p \equiv t \wedge x \not\equiv t \wedge x.l.m \wedge x.r.m\}$
$\{x \not\equiv p \wedge x.l.m \wedge x.r.m\}$
t := t.l;
$\{x \not\equiv p \wedge x.l.m \wedge x.r.m\}$
p.l := q;
$\{x \not\equiv p \wedge x.l.m \wedge x.r.m\}$
p.m := TRUE; (16)
$\{x \not\equiv p \wedge x.l.m \wedge x.r.m\}$
p.c := FALSE;
$\{x \not\equiv p \wedge x.l.m \wedge x.r.m\}$
$\{x.l.m \wedge x.r.m\}$ ■

In (16) $x.l$ or $x.r$ might equal p, but since $p.m$ is set to TRUE, this doesn't matter, since we require it to be set to TRUE. (Actually, I'm not sure if $x.l$ or $x.r$ could really equal p for a node initially satisfying $x.l.m \wedge x.r.m$)

5.3.5 P5

$$P5: \quad \forall x \in AllNodes: \ \overline{x.m} \vee \overline{x.c} \vee x.l.m$$

We'll split all nodes into the same groups as for P4.
For the group for which $x \not\equiv t$:

- If $\overline{x.m}$ or $\overline{x.c}$, no m or c-property other than for $x \equiv t$ is affected in PUSH (proof see at P4).
- If $x.l.m$, see very similar proof in P4.

For $x \equiv t$:

$\{x \equiv t\}$
q := p :
$\{x \equiv t\}$
p := t;
$\{x \equiv t \wedge p \equiv t\}$
$\{x \equiv p\}$
t := t.l;
$\{x \equiv p\}$
p.l := q;
$\{x \equiv p\}$
p.m := TRUE;
$\{x \equiv p\}$
p.c := FALSE;
$\{x \equiv p \wedge \overline{p.c}\}$
$\{\overline{x.c}\}$
$\{P5\}$ ■

5.4 The Invariants with Respect to SWING

```
/* SWING */
{t.m and not p.c}
q := t; t := p.r; p.r := p.l;
p.l := q; p.c := TRUE;
```

5.4.1 P1

$$\begin{aligned} P1 &=_{def} \forall x \in ARR: \ OnStack(x) \vee NotOnStack(x) \\ OnStack(x) &=_{def} p \xrightarrow{s} x \wedge x.\overline{s} \equiv x.\overline{s_0} \wedge x.s.s_0 \equiv x \wedge x.m \\ NotOnStack(x) &=_{def} p \overset{s}{\not\longrightarrow} x \wedge x.l \equiv x.l_0 \wedge x.r \equiv x.r_0 \end{aligned}$$

P1 must be proven for all nodes x. Lets do again two groups: $x \not\equiv p$ and $x \equiv p$.

For $x \equiv p$:

$$
\begin{array}{rl}
 & \{t.m \wedge \overline{p.c} \wedge x \equiv p\} \\
\textit{Loop Guard} & \{t.m \wedge \overline{p.c} \wedge x \equiv p \wedge p \not\equiv Void\} \\
P3 & \{t.m \wedge \overline{p.c} \wedge x \equiv p \wedge p \not\equiv Void \wedge p \xrightarrow{s} Void\} \\
R6 & \{t.m \wedge \overline{p.c} \wedge x \equiv p \wedge p.s \not\xrightarrow{s} p\} \\
R3 & \{t.m \wedge \overline{p.c} \wedge x \equiv p \wedge p \not\equiv p.s\} \\
P1 & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge p \not\equiv p.s\} \\
P2 & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge p.s_0 \equiv t \wedge p \not\equiv p.s\} \\
 & \text{q := t :} \\
A4.1a \ \& \ A4.2a & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge p.s_0 \equiv t \wedge q \equiv t \wedge p \not\equiv p.s\} \\
A4.1a & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge p.s_0 \equiv q \wedge p \not\equiv p.s\} \\
 & \text{t := p.r;} \\
A4.1a & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge p.s_0 \equiv q \wedge p \not\equiv p.s\} \\
DEF\ s & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.l.s_0 \equiv p \wedge p.r \equiv p.r_0 \wedge p.l_0 \equiv q \wedge p \not\equiv p.l\} \\
 & \text{p.r := p.l;} \\
A4.1a \ \& \ A4.2b & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.l.s_0 \equiv p \wedge p.r \equiv p.l \wedge p.l_0 \equiv q \wedge p \not\equiv p.l\} \\
PE & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.r.s_0 \equiv p \wedge p.l_0 \equiv q \wedge p \not\equiv p.r\} \\
 & \text{p.l := q;} \\
A4.1a & \{\overline{p.c} \wedge x \equiv p \wedge p.m \wedge p.r.s_0 \equiv p \wedge p.l_0 \equiv q \wedge p.l \equiv q \wedge p \not\equiv p.r\} \\
 & \text{p.c := TRUE;} \\
A4.1a \ \& \ A4.2b & \{p.c \wedge x \equiv p \wedge p.m \wedge p.r.s_0 \equiv p \wedge p.l_0 \equiv q \wedge p.l \equiv q\} \\
DEF\ s & \{x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s_0} \equiv q \wedge p.\overline{s} \equiv q\} \\
R3 & \{p \xrightarrow{s} x \wedge x \equiv p \wedge p.m \wedge p.s.s_0 \equiv p \wedge p.\overline{s_0} \equiv q \wedge p.\overline{s} \equiv q\} \\
TR & \{p \xrightarrow{s} x \wedge x.m \wedge x.s.s_0 \equiv x \wedge x.\overline{s} \equiv x.\overline{s_0}\} \\
 & \{P1\}
\end{array}
$$

For $x \not\equiv p$ we know that no property of x is affected by SWING, since p itself doesn't change as there is no assignment to p and only properties of p, not of any other root, are affected by SWING.

So we only need to prove that the connectivity of p to x does not get changed.

If $p \xrightarrow{s} x$, see the proof of P3 and substitute $Void$ by x. (Instead of $\textit{Loop Guard} \Longrightarrow p \not\equiv Void$ just write $p \not\equiv x$, since we assume it explicitly here)

For $p \not\xrightarrow{s} x$:

$$
\begin{array}{rl}
If\ Guard & \{P3 \wedge t.m \wedge \overline{p.c}\} \\
P1\ \&\ LC & \{P3 \wedge t.m \wedge \overline{p.c} \wedge p \wedge p \not\equiv Void \wedge p \xrightarrow{s} Void\} \\
R6 & \{P3 \wedge t.m \wedge \overline{p.c} \wedge p \wedge p.s \overset{s}{\not\longrightarrow} p\} \\
Assumption & \{p \overset{s}{\not\longrightarrow} x \wedge t.m \wedge \overline{p.c} \wedge p.s \overset{s}{\not\longrightarrow} p\} \\
Assumption & \{p \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p \not\equiv x \wedge p.s \overset{s}{\not\longrightarrow} p\} \\
R3 & \{p.s \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p \not\equiv x \wedge p.s \overset{s}{\not\longrightarrow} p\} \\
DEF\ s & \{p.l \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p \not\equiv x \wedge p.l \overset{s}{\not\longrightarrow} p\} \\
 & \text{q := t :} \\
R1 & \{p.l \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p \not\equiv x \wedge p.l \overset{s}{\not\longrightarrow} p\} \\
 & \text{t := p.r;} \\
R1 & \{p.l \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p \not\equiv x \wedge p.l \overset{s}{\not\longrightarrow} p\} \\
 & \text{p.r := p.l;} \\
R1\ \&\ DEF\ s & \{p.l \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p.r \equiv p.l \wedge p \not\equiv x \wedge p.l \overset{s}{\not\longrightarrow} p\} \\
DEF\ s & \{p.r \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p.r \equiv p.l \wedge p \not\equiv x \wedge p.r \overset{s}{\not\longrightarrow} p\} \\
R7 & \{p.r \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p.r \equiv p.l \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p.l \not> p.r.s^* \wedge p \not\equiv x\} \qquad (17) \\
 & \text{p.l := q;} \\
R1 & \{p.r \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p \not\equiv x\} \\
R7 & \{p.r \overset{s}{\not\longrightarrow} x \wedge \overline{p.c} \wedge p.c \not> p.r.s^* \wedge p \not\equiv x\} \\
 & \text{p.c := TRUE;} \\
R1 & \{p.r \overset{s}{\not\longrightarrow} x \wedge p.c \equiv TRUE \wedge p \not\equiv x\} \\
DEF\ s & \{p.s \overset{s}{\not\longrightarrow} x \wedge p \not\equiv x\} \\
R3 & \{p \overset{s}{\not\longrightarrow} x\} \qquad \blacksquare
\end{array}
$$

For (17) see Section 5.3.3.

5.4.2 P2

$$P2: \qquad p.s_0 \equiv t$$

	$\{\overline{p.c}\}$
$P1$	$\{\overline{p.c} \wedge p.\bar{s} \equiv p.\overline{s_0}\}$
$DEF\ s$	$\{\overline{p.c} \wedge p.r \equiv p.r_0\}$
	q := t :
$A4.1a$	$\{p.r \equiv p.r_0\}$
	t := p.r;
$A4.1a$ & $A4.2a$	$\{p.r \equiv p.r_0 \equiv t\}$
	p.r := p.l;
$A4.1a$	$\{p.r_0 \equiv t\}$
	p.l := q;
$A4.1a$	$\{p.r_0 \equiv t\}$
	p.c := TRUE;
$A4.1a$ & $A4.2a$	$\{p.r_0 \equiv t \wedge p.c := TRUE\}$
$DEF\ s$	$\{p.s_0 \equiv t\}$ ■

5.4.3 P3

$$P3: \qquad p \xrightarrow{s} Void$$

$\{P3 \wedge t.m \wedge \overline{p.c}\}$

$\{p \xrightarrow{s} Void \wedge t.m \wedge \overline{p.c}\}$

Loop guard $\{p \xrightarrow{s} Void \wedge \overline{p.c} \wedge p \not\equiv Void\}$

*R*6 $\{p \xrightarrow{s} Void \wedge \overline{p.c} \wedge p \not\equiv Void \wedge p.s \overset{s}{\not\longrightarrow} p\}$

*R*3 $\{p.s \xrightarrow{s} Void \wedge \overline{p.c} \wedge p \not\equiv Void \wedge p.s \overset{s}{\not\longrightarrow} p\}$

DEF s $\{p.l \xrightarrow{s} Void \wedge \overline{p.c} \wedge p \not\equiv Void \wedge p.l \overset{s}{\not\longrightarrow} p\}$

q := t :

*R*1 $\{p.l \xrightarrow{s} Void \wedge \overline{p.c} \wedge p \not\equiv Void \wedge p.l \overset{s}{\not\longrightarrow} p\}$

t := p.r;

*R*1 $\{p.l \xrightarrow{s} Void \wedge \overline{p.c} \wedge p \not\equiv Void \wedge p.l \overset{s}{\not\longrightarrow} p\}$

p.r := p.l;

*R*1 & *DEF s* $\{p.l \xrightarrow{s} Void \wedge \overline{p.c} \wedge p.r \equiv p.l \wedge p \not\equiv Void \wedge p.l \overset{s}{\not\longrightarrow} p\}$

DEF s $\{p.r \xrightarrow{s} Void \wedge \overline{p.c} \wedge p.r \equiv p.l \wedge p \not\equiv Void \wedge p.r \overset{s}{\not\longrightarrow} p\}$

*R*7 $\{p.r \xrightarrow{s} Void \wedge \overline{p.c} \wedge p.r \equiv p.l \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p.l \not> p.r.s^* \wedge p \not\equiv Void\}$

p.l := q;

*R*1 $\{p.r \xrightarrow{s} Void \wedge \overline{p.c} \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p \not\equiv Void\}$

*R*7 $\{p.r \xrightarrow{s} Void \wedge \overline{p.c} \wedge p.c \not> p.r.s^* \wedge p \not\equiv Void\}$

p.c := TRUE;

*R*1 $\{p.r \xrightarrow{s} Void \wedge p.c \equiv TRUE \wedge p \not\equiv Void\}$

DEF s $\{p.s \xrightarrow{s} Void \wedge p \not\equiv Void\}$

*R*3 $\{p \xrightarrow{s} Void\}$ ■

5.4.4 P4

$$P4: \quad \forall x \in AllNodes: \ \overline{x.m} \vee p \xrightarrow{s} x \vee (x.l.m \wedge x.r.m)$$

P4 must be proven for all nodes x. Lets do two groups: $x \not\equiv p$ and $x \equiv p$. For $x \equiv p$ P4 is trivially satisfied since there is no assignment to p in SWING and $p \xrightarrow{s} p$.

For $x \not\equiv p$:

- All nodes satisfying $\overline{x.m}$ will still satisfy it since no m-properties are affected in SWING.
- All nodes satisfying $p \xrightarrow{s} x$, see the proof of P3 and substitute $Void$ by x. $p \not\equiv x$ is true since we assume it explicitly.
- All nodes satisfying $x.l.m \wedge x.r.m$:

$$
\begin{array}{rl}
P4 & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{q := t :} \\
A4.1a,b & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{t := p.r;} \\
A4.1a,b & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{p.r := p.l;} \\
A4.1a,b & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{p.l := q;} \\
A4.1a,b & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{p.c := TRUE;} \\
A4.1a,b & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \{x.l.m \wedge x.r.m\} \qquad \blacksquare
\end{array}
$$

5.4.5 P5

$$P5: \qquad \forall x \in AllNodes: \ \overline{x.m} \vee \overline{x.c} \vee x.l.m$$

Again lets split the nodes in the same two groups as for P4. For the group of nodes for which $x \not\equiv p$, all arguments can be taken from P4, since SWING will not affect their properties (very similar proof see P4).

For $x \equiv p$:

$$
\begin{array}{l}
\{x \equiv p \wedge t.m \wedge \overline{p.c}\} \\
\text{q := t :} \\
\{x \equiv p \wedge t.m \wedge q \equiv t\} \\
\{x \equiv p \wedge q.m\} \\
\text{t := p.r;} \\
\{x \equiv p \wedge q.m\} \\
\text{p.r := p.l;} \\
\{x \equiv p \wedge q.m\} \\
\text{p.l := q;} \\
\{x \equiv p \wedge q.m \wedge p.l \equiv q\} \\
\{x \equiv p \wedge q.m \wedge p.l.m \equiv q.m\} \\
\{x \equiv p \wedge p.l.m\} \\
\text{p.c := TRUE;} \\
\{x \equiv p \wedge p.l.m\} \\
\{x.l.m\} \\
\{P5\} \qquad \blacksquare
\end{array}
$$

5.5 The Invariants with Respect to POP

```
/* POP */
{t.m and p.c}
q := t; t := p; p := p.r; t.r := q;
```

5.5.1 P1

$$
\begin{aligned}
P1 &=_{def} \forall x \in ARR:\ OnStack(x) \vee NotOnStack(x) \\
OnStack(x) &=_{def} p \xrightarrow{s} x \wedge x.\overline{s} \equiv x.\overline{s_0} \wedge x.s.s_0 \equiv x \wedge x.m \\
NotOnStack(x) &=_{def} p \overset{s}{\not\longrightarrow} x \wedge x.l \equiv x.l_0 \wedge x.r \equiv x.r_0
\end{aligned}
$$

As we are already used to, we'll prove P1 separately for two groups: $x \equiv p$ and $x \not\equiv p$.

For $x \equiv p$: (AR means by a combination of rule 1, A4.1a, A4.2a)

$$
\begin{aligned}
If &\quad \{x \equiv p \wedge t.m \wedge p.c\} \\
P3\&LC &\quad \{x \equiv p \wedge t.m \wedge p.c \wedge p \not\equiv Void \wedge p \xrightarrow{s} Void\} \\
R6 &\quad \{x \equiv p \wedge p.c \wedge p.s \overset{s}{\not\longrightarrow} p\} \\
P2 &\quad \{x \equiv p \wedge p.c \wedge p.s \overset{s}{\not\longrightarrow} p \wedge p.s_0 \equiv t\} \\
P1 &\quad \{x \equiv p \wedge p.c \wedge p.s \overset{s}{\not\longrightarrow} p \wedge p.\overline{s} \equiv p.\overline{s_0} \wedge p.s.s_0 \equiv p \wedge p.m \wedge p.s_0 \equiv t\} \\
DEF\ s &\quad \{x \equiv p \wedge p.c \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p.l \equiv p.l_0 \wedge p.r.s_0 \equiv p \wedge p.m \wedge p.r_0 \equiv t\} \\
&\quad \text{q := t :} \\
AR &\quad \{x \equiv p \wedge p.c \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p.l \equiv p.l_0 \wedge p.r.s_0 \equiv p \wedge p.m \wedge q \equiv t \wedge p.r_0 \equiv t\} \\
&\quad \{x \equiv p \wedge p.c \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p.l \equiv p.l_0 \wedge p.r.s_0 \equiv p \wedge p.m \wedge p.r_0 \equiv q\} \\
&\quad \text{t := p;} \\
AR &\quad \{x \equiv p \wedge p.c \wedge p.r \overset{s}{\not\longrightarrow} p \wedge p.l \equiv p.l_0 \wedge p.r.s_0 \equiv p \wedge p.m \wedge t \equiv p \wedge p.r_0 \equiv q\} \\
&\quad \{x \equiv t \wedge t.c \wedge t.r \overset{s}{\not\longrightarrow} t \wedge t.l \equiv t.l_0 \wedge t.r.s_0 \equiv t \wedge t.m \wedge t.r \equiv p.r \wedge t.r_0 \equiv q\} \\
&\quad \text{p := p.r;} \\
AR &\quad \{x \equiv t \wedge t.c \wedge t.r \overset{s}{\not\longrightarrow} t \wedge t.l \equiv t.l_0 \wedge t.r.s_0 \equiv t \wedge t.m \wedge t.r \equiv p \wedge t.r_0 \equiv q\} \\
&\quad \{x \equiv t \wedge t.c \wedge p \overset{s}{\not\longrightarrow} t \wedge t.l \equiv t.l_0 \wedge t.r.s_0 \equiv t \wedge t.m \wedge t.r_0 \equiv q\} \\
R7 &\quad \{x \equiv t \wedge t.c \wedge p \overset{s}{\not\longrightarrow} t \wedge t.l \equiv t.l_0 \wedge p.s_0 \equiv t \wedge t.m \wedge t.r \not\succ p.s^* \wedge t.r_0 \equiv q\} \\
&\quad \text{t.r := q;} \\
AR &\quad \{x \equiv t \wedge t.c \wedge p \overset{s}{\not\longrightarrow} t \wedge t.l \equiv t.l_0 \wedge p.s_0 \equiv t \wedge t.m \wedge t.r \equiv q \wedge t.r_0 \equiv q\} \\
&\quad \{x \equiv t \wedge p \overset{s}{\not\longrightarrow} t \wedge t.l \equiv t.l_0 \wedge t.r \equiv t.r_0\} \\
&\quad \{p \overset{s}{\not\longrightarrow} x \wedge x.l \equiv x.l_0 \wedge x.r \equiv x.r_0\} \\
&\quad \{P1\}
\end{aligned}
$$

For $x \not\equiv p$:

$\{x \not\equiv p \ \wedge \ x.* \equiv h_0\}$
q := t :
$\{x \not\equiv p \ \wedge \ x.* \equiv h_0\}$
t := p;
$\{x \not\equiv p \ \wedge \ x.* \equiv h_0 \ \wedge \ t \equiv p\}$
$\{x \not\equiv t \ \wedge \ x.* \equiv h_0\}$
p := p.r;
$\{x \not\equiv t \ \wedge \ x.* \equiv h_0\}$
t.r := q;
$\{x \not\equiv t \ \wedge \ x.* \equiv h_0\}$

Since no property of x gets affected, we only need to prove that the connectivity property of p to x remains unchanged.

If $p \xrightarrow{s} x$, see proof of R3 and substitute $Void$ by x. $p \not\equiv x$ is assumed explicitly, so we don't need to consider the Loop Guard.

If $p \not\xrightarrow{s} x$

$\{t.m \ \wedge \ p.c \ \wedge \ p \not\xrightarrow{s} x\}$

$P1$ $\{t.m \ \wedge \ p.c \ \wedge \ p \not\xrightarrow{s} x \ \wedge \ p \xrightarrow{s} Void\}$

LG $\{t.m \ \wedge \ p.c \ \wedge \ p \not\xrightarrow{s} x \ \wedge \ p \xrightarrow{s} Void \ \wedge \ p \not\equiv Void\}$

$R6$ $\{t.m \ \wedge \ p.c \ \wedge \ p \not\xrightarrow{s} x \ \wedge \ p.s \not\xrightarrow{s} p\}$

q := t :

$A4.1a$ $\{p.c \ \wedge \ p \not\xrightarrow{s} x \ \wedge \ p.s \not\xrightarrow{s} p\}$

t := p;

$A4.1a$ $\{p.c \ \wedge \ p \not\xrightarrow{s} x \ \wedge \ p \equiv t \ \wedge \ p.s \not\xrightarrow{s} p\}$

$R3 \ \& \ PE$ $\{p.c \ \wedge \ p.s \not\xrightarrow{s} x \ \wedge \ p \equiv t \ \wedge \ t.s \not\xrightarrow{s} t\}$

$DEF \ s \ \& \ PE$ $\{t.c \ \wedge \ p.r \not\xrightarrow{s} x \ \wedge \ t.r \equiv p.r \ \wedge \ t.r \xrightarrow{s} t\}$

PE $\{t.c \ \wedge \ t.r \not\xrightarrow{s} x \ \wedge \ t.r \equiv p.r \ \wedge \ t.r \not\xrightarrow{s} t\}$

p := p.r;

$A4.1a \ \& \ A4.2a$ $\{t.c \ \wedge \ t.r \equiv p \ \wedge \ t.r \not\xrightarrow{s} x \ \wedge \ t.r \not\xrightarrow{s} t\}$

PE $\{t.c \ \wedge \ t.r \equiv p \ \wedge \ p \not\xrightarrow{s} x \ \wedge \ p \not\xrightarrow{s} t\}$

$R7$ $\{p \not\xrightarrow{s} x \ \wedge \ t.r \not\succ p.s^*\}$

t.r := q;

$R1$ $\{p \not\xrightarrow{s} x\}$ ■

5.5.2 P2

$$P2: \qquad p.s_0 \equiv t$$

	$\{t.m \wedge p.c\}$
$P1$	$\{p.s.s_0 \equiv p \wedge t.m \wedge p.c\}$
	q := t :
$A4.1a$	$\{p.s.s_0 \equiv p \wedge p.c\}$
	t := p;
$A4.1a$ & $A4.2a$	$\{p.s.s_0 \equiv p \wedge p.c \wedge t \equiv p\}$
PE	$\{t.s.s_0 \equiv t \wedge t.c \wedge t \equiv p \wedge t.r \equiv p.r\}$
	p := p.r;
$A4.1a$ & $A4.2a$	$\{t.s.s_0 \equiv t \wedge t.c \wedge t.r \equiv p\}$
$DEF\ s$	$\{t.r.s_0 \equiv t \wedge t.c \wedge t.r \equiv p\}$
PE	$\{p.s_0 \equiv t \wedge t.c\}$
$DEF\ s$	$\{p.s_0 \equiv t\}$
	t.r := q;
$A4.2a$	$\{p.s_0 \equiv t\}$ ■

5.5.3 P3

$$P3: \qquad p \xrightarrow{s} Void$$

$$
\begin{array}{rl}
 & \{t.m \wedge p.c \wedge p \xrightarrow{s} Void\} \\
\textit{Loop guard} & \{t.m \wedge p.c \wedge p \xrightarrow{s} Void \wedge p \not\equiv Void\} \\
R6 & \{t.m \wedge p.c \wedge p \xrightarrow{s} Void \wedge p \not\equiv Void \wedge p.s \overset{l}{\not\longrightarrow} p\} \\
 & \text{q := t :} \\
A4.1a & \{p.c \wedge p \xrightarrow{s} Void \wedge p \not\equiv Void \wedge p.s \overset{l}{\not\longrightarrow} p\} \\
 & \text{t := p;} \\
A4.1a & \{p.c \wedge p \xrightarrow{s} Void \wedge p \not\equiv Void \wedge p \equiv t \wedge p.s \overset{l}{\not\longrightarrow} p\} \\
R3 \;\&\; PE & \{p.c \wedge p.s \xrightarrow{s} Void \wedge p \equiv t \wedge t.s \overset{l}{\not\longrightarrow} t\} \\
DEF\ s \;\&\; PE & \{t.c \wedge p.r \xrightarrow{s} Void \wedge t.r \equiv p.r \wedge t.r \overset{l}{\not\longrightarrow} t\} \\
PE & \{t.c \wedge t.r \xrightarrow{s} Void \wedge t.r \equiv p.r \wedge t.r \overset{l}{\not\longrightarrow} t\} \\
 & \text{p := p.r;} \\
A4.1a \;\&\; A4.2a & \{t.c \wedge t.r \xrightarrow{s} Void \wedge t.r \equiv p \wedge t.r \overset{l}{\not\longrightarrow} t\} \\
PE & \{p \xrightarrow{s} Void \wedge p \overset{l}{\not\longrightarrow} t\} \\
R7 & \{p \xrightarrow{s} Void \wedge t.r \not\succ p.s^n\} \qquad (18) \\
 & \text{t.r := q;} \\
R4 & \{p \xrightarrow{s} Void\} \qquad \blacksquare
\end{array}
$$

Explanation for (18) see Section 5.3.3.

5.5.4 P4

$$P4: \qquad \forall x \in AllNodes: \ \overline{x.m} \vee p \xrightarrow{s} x \vee (x.l.m \wedge x.r.m)$$

P4 must be proven for all nodes x. Lets do again two groups: $x \not\equiv p$ and $x \equiv p$.

For $x \equiv p$:

$$
\begin{array}{rl}
 & \{t.m \wedge p.c \wedge x \equiv p\} \\
P1 & \{t.m \wedge p.c \wedge x \equiv p \wedge p.m\} \\
P5 & \{t.m \wedge p.c \wedge x \equiv p \wedge p.m \wedge p.l.m\} \\
PE & \{t.m \wedge x.l.m \wedge x \equiv p\} \\
 & \text{q := t :} \\
A4.1a \ \& \ A4.2a & \{q \equiv t \wedge t.m \wedge x.l.m \wedge x \equiv p\} \\
PE & \{q.m \wedge x.l.m \wedge x \equiv p\} \\
 & \text{t := p;} \\
A4.1a \ \& \ A4.2a & \{q.m \wedge x.l.m \wedge x \equiv p \wedge t \equiv p\} \\
PE & \{q.m \wedge x.l.m \wedge x \equiv t\} \\
 & \text{p := p.r;} \\
A4.1a & \{q.m \wedge x.l.m \wedge x \equiv t\} \\
 & \text{t.r := q;} \\
A4.1a \ \& \ A4.2a & \{q.m \wedge x.l.m \wedge t.r \equiv q \wedge x \equiv t\} \\
PE & \{q.m \wedge x.l.m \wedge x.r.m \equiv q.m\} \\
PE & \{x.l.m \wedge x.r.m\} \qquad \blacksquare
\end{array}
$$

For $x \not\equiv p$:

- All nodes satisfying $\overline{x.m}$ will still satisfy it since no m-properties are affected in POP.
- All nodes satisfying $p \xrightarrow{s} x$, see the proof of P3 and substitute $Void$ by x. $p \not\equiv x$ is true since we assume it explicitly.
- All nodes satisfying $x.l.m \wedge x.r.m$:

$$
\begin{array}{rl}
 & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{q := t :} \\
A4.1a, b & \{x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
 & \text{t := p;} \\
A4.1a, b \ \& \ A4.2a & \{t \equiv p \wedge x \not\equiv p \wedge x.l.m \wedge x.r.m\} \\
PE & \{t \not\equiv x \wedge x.l.m \wedge x.r.m\} \\
 & \text{p := p.r;} \\
A4.1a, b & \{t \not\equiv x \wedge x.l.m \wedge x.r.m\} \\
 & \text{t.r := q;} \\
A4.1a, b & \{t \not\equiv x \wedge x.l.m \wedge x.r.m\} \\
 & \{x.l.m \wedge x.r.m\} \qquad \blacksquare
\end{array}
$$

5.5.5 P5

$$P5: \qquad \forall x \in AllNodes: \ \overline{x.m} \vee \overline{x.c} \vee x.l.m$$

P5 is trivially true since during POP no m, c or l-properties are changed. $\blacksquare$

6 Conclusion

The proof is complete, the first mountain has been climbed ! Bernd Schoellers new formalism for pointer aliasing based on his path properties showed its simplicity and expression power. Expression power since the proof is complete, and simplicity since the individual proofs are quite simple and clear, even though they look long and crowded, it's because nearly every step has been listed.

What's is the next step ? Probably the next step is that an automatic proof program like Isabelle/HOL should be able to do the proof.

References

[Schoeller05] Bernd Schoeller. Alias-based Reasoning for Object-Oriented Programs. 2005

[SchoellerSummary] Bernd Schoeller. Mathematics of Path Properties. Summary of formulas of [Schoeller05]. 2005

[Morris82] Joseph M. Morris. A general axiom of assignment. Assignment and linked data structures. A proof of the Schorr-Waite algorithm. In M. Broy and G. Schmidt, editors, Theoretical Foundations of Programming Methodology, (Lecture Notes International Summer School, Markoberdorf), pages 25-51, Reidel, Dortrecht, Netherlands, 1982.

[Hoare69] C.A.R.Hoare, An axiomatic basis for computer programming. Communications of the ACM, pages 576-583, October 1969.

[SchorrWaite67] H. Schorr and W.M.Waite. An efficient machine-independent procedure for garbage collection in various list structures. Commun. ACM, 10:501-506, 1967.

[HubertMarché05] Hubert and Marché: A case study of C source code verification: the Schorr-Waite algoritm. 2005. (Not sure, might be 04)

[Bornat00] R. Bornat. Proving pointer programs in Hoare logic. In Mathematics of Program Construction, pages 102-126, 2000.