Verification of the Schorr-Waite Graph Marking Algorithm by Refinement

Hongseok Yang

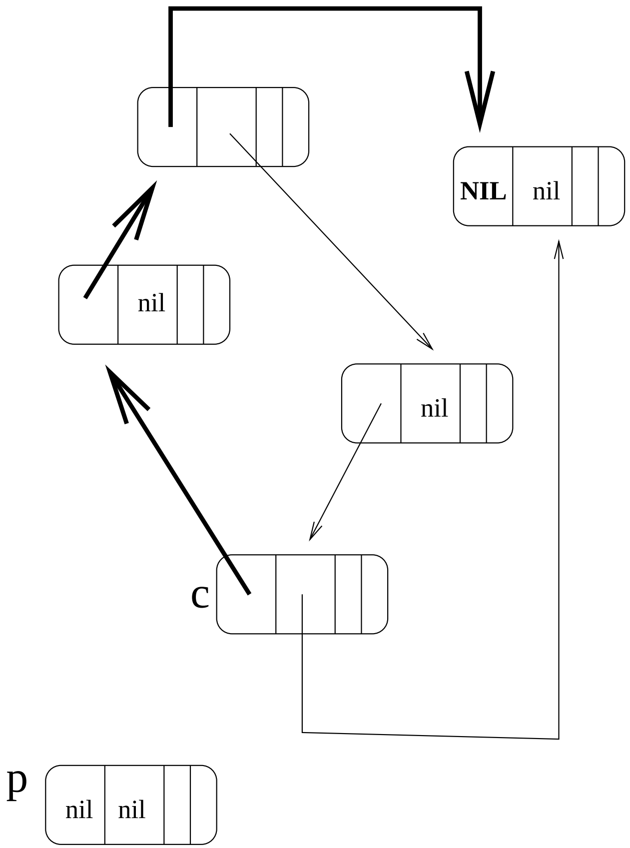
ROPAS, KAIST

Email: `hyang@ropas.kaist.ac.kr`

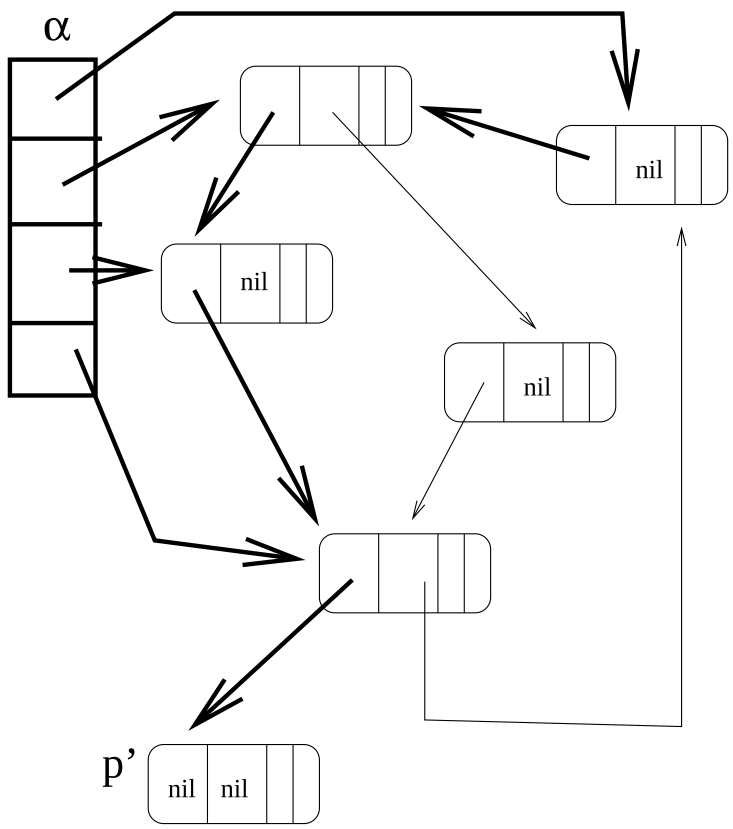
August 2002

The Schorr-Waite Graph Marking Algorithm

Given the root of a graph, the algorithm marks all the reachable nodes from the root by doing the depth-first traversal; the most creative part of the algorithm lies in implementing a stack without additional memory by reversing the link fields of nodes.



SWMarking Algorithm



StackMarking Algorithm

Two Specifications of the Schorr-Waite Algorithm

The full specification:

> Given a graph of unmarked nodes and its root, the algorithm marks all the nodes reachable from the root; moreover, when it terminates, the algorithm restores the link fields of the nodes to their initial values.

A partial specification:

> The algorithm does the same thing as the depth-first traversal with an explicit stack.

Question

Can we formulate a program logic to show that the Schorr-Waite algorithm does the same thing as the depth-first traversal?

Main Idea: Hoare Quadruple

$$\{\mathsf{Same} \wedge p = p' \wedge \mathsf{NoDangling}(p, p')\} \begin{array}{c}\mathsf{SWMarking}(p) \\ \mathsf{StackMarking}(p')\end{array} \{\mathsf{Same} \wedge \mathsf{NoDangling}(p, p')\}$$

- With relations, it is easy to compare the heaps of the two programs; consequently, simple relations are often enough.
- The proof rules for showing quadruples can easily be obtained from those of the Separation Logic.
- This extends the methods of Reynolds, Hoare and Morgan to handle heaps.[a]

[a] For a more complete list of references, see de Roever and Engelhart's book.

Semantic Domains

$$
\begin{aligned}
\mathrm{Locs} \cup \{\mathsf{nil}\} &\subseteq \mathrm{Vals} \\
\mathrm{Stacks} &\triangleq \mathrm{Vars} \rightharpoonup_{fin} \mathrm{Vals} \\
\mathrm{Heaps} &\triangleq \mathrm{Locs} \rightharpoonup_{fin} \mathrm{Vals} \times \mathrm{Vals} \times \mathrm{Vals} \times \mathrm{Vals} \\
\mathrm{States} &\triangleq \mathrm{Stacks} \times \mathrm{Heaps}
\end{aligned}
$$

Relations

$$\text{Assertions } P, Q := (E \mapsto E, E, E, E) \mid E = E' \mid \mathsf{emp} \mid P * Q \mid P \Rightarrow Q \mid \forall x.\, P$$

$$\text{Relations } R, S := \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} \mid E = E' \mid \mathsf{Emp} \mid R * S \mid R \Rightarrow S \mid \forall x.\, R$$

Review: Semantics of Assertions

For $s \in$ Stacks and $h \in$ Heaps,[a]

$$
\begin{aligned}
s, h \models (E \mapsto E_1, E_2, E_3, E_4) &\text{ iff } \mathsf{dom(h)} = \{[\![\mathsf{E}]\!]\mathsf{s}\} \\
&\qquad \wedge h([\![E]\!]s) = ([\![E_1]\!]s, [\![E_2]\!]s, [\![E_3]\!]s, [\![E_4]\!]s) \\
s, h \models P * Q &\text{ iff } \exists h', h''. \\
&\qquad h = h' * h'' \wedge s, h' \models P \wedge s, h'' \models Q \\
s, h \models \mathsf{emp} &\text{ iff } h = [] \\
s, h \models E = E' &\text{ iff } [\![E]\!]s = [\![E']\!]s \\
s, h \models P \Rightarrow Q &\text{ iff } s, h \models P \Longrightarrow s, h \models Q
\end{aligned}
$$

[a] $h_1 \# h_2$ holds iff $\mathsf{dom(h_1)} \cap \mathsf{dom(h_2)} = \emptyset$. When $h_1 \# h_2$, $h_1 * h_2$ is the union of h_1 and h_2.

Semantics of Relations

For $s \in$ Stacks and $h_1, h_2 \in$ Heaps,

$$
\begin{array}{lll}
s, h_1, h_2 \models \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} & \text{iff} & s, h_1 \models P_1 \text{ and } s, h_2 \models P_2 \\
s, h_1, h_2 \models R * S & \text{iff} & \exists h_1', h_1'', h_2', h_2''. \\
 & & \qquad h_1 = h_1' * h_1'' \wedge h_2 = h_2' * h_2'' \\
 & & \qquad \wedge\, s, h_1', h_2' \models R \wedge s, h_1'', h_2'' \models S \\
s, h_1, h_2 \models \mathsf{Emp} & \text{iff} & h_1 = h_2 = [] \\
s, h_1, h_2 \models E = E' & \text{iff} & [\![E]\!]s = [\![E']\!]s \\
s, h_1, h_2 \models R \Rightarrow S & \text{iff} & s, h_1, h_2 \models R \Longrightarrow s, h_1, h_2 \models S
\end{array}
$$

Example Relations

1. $\exists x, l, r, m, d. \begin{pmatrix} x \mapsto l, r, m, d \\ x \mapsto l, r, m, d \end{pmatrix}$

2. $\mathsf{Same} \triangleq \mathsf{Emp} \vee (\exists x, l, r, m, d. \begin{pmatrix} x \mapsto l, r, m, d \\ x \mapsto l, r, m, d \end{pmatrix}) * \mathsf{Same}$

3. $\mathsf{Twisted} \triangleq \mathsf{Emp} \vee (\exists x, l, r, m, d. \begin{pmatrix} x \mapsto l, r, m, d \\ x \mapsto r, l, m, d \end{pmatrix}) * \mathsf{Twisted}$

Hoare Quadruple

When $\mathrm{FV}(C_1) \cap \mathrm{FV}(C_2) = \emptyset$, the quadruple $\{R\} \begin{matrix} C_1 \\ C_2 \end{matrix} \{S\}$ says that

for all s, h_1, h_2 such that $s, h_1, h_2 \models R$,

1. both $s|_{\mathrm{FV}(C_1)}, h_1, C_1$ and $s|_{\mathrm{FV}(C_2)}, h_2, C_2$ do not generate memory faults;
2. $s|_{\mathrm{FV}(C_1)}, h_1, C_1$ may diverge iff $s|_{\mathrm{FV}(C_2)}, h_2, C_2$ may diverge;
3. for all s', h'_1, h'_2 such that
   - $s|_{\mathrm{FV}(C_1)}, h_1, C_1 \leadsto^* s'|_{\mathrm{FV}(C_1)}, h'_1$,
   - $s|_{\mathrm{FV}(C_2)}, h_2, C_2 \leadsto^* s'|_{\mathrm{FV}(C_2)}, h'_2$ and
   - $s|_{\mathrm{Vars}-\mathrm{FV}(C_1,C_2)} = s'|_{\mathrm{Vars}-\mathrm{FV}(C_1,C_2)}$,

   we have $s', h'_1, h'_2 \models S$.

Example Specifications by Hoare Quadruples

$$\{\mathsf{Same} \wedge p = p' \wedge \mathsf{NoDangling}(p, p')\} \begin{array}{c}\mathsf{SWMarking}(p) \\ \mathsf{StackMarking}(p')\end{array} \{\mathsf{Same} \wedge \mathsf{NoDangling}(p, p')\}$$

Proof Rule: Embedding from Hoare Triples

$$\frac{[P_1]\ C_1\ [Q_1] \qquad [P_2]\ C_2\ [Q_2]}{\left\{\begin{pmatrix} P_1 \\ P_2 \end{pmatrix}\right\} \begin{matrix} C_1 \\ C_2 \end{matrix} \left\{\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}\right\}}$$

when $\mathrm{Modifies}(C_1) \cap \mathrm{FV}(P_2, Q_2) = \mathrm{Modifies}(C_2) \cap \mathrm{FV}(P_1, Q_1) = \emptyset$.

- Only total correctness triples can be embedded.
  - Recall that a *total* correctness triple $[P]C[Q]$ says that for all s, h satisfying P, (s, h, C) *always terminates* without memory faults, and all the final states satisfy Q.
- Many proof rules from the Separation Logic can be embedded.

Instances

$$\frac{}{\left\{\begin{pmatrix} E \mapsto E_1, E_2, E_3, E_4 \\ P \end{pmatrix}\right\} \begin{matrix} E.2 := F \\ \mathsf{skip} \end{matrix} \left\{\begin{pmatrix} E \mapsto E_1, F, E_3, E_4 \\ P \end{pmatrix}\right\}}$$

$$\frac{}{\left\{\begin{pmatrix} P \\ Q * (E \mapsto E_1, E_2, E_3, E_4) \end{pmatrix}\right\} \begin{matrix} \mathsf{skip} \\ \mathsf{dispose}(E) \end{matrix} \left\{\begin{pmatrix} P \\ Q \end{pmatrix}\right\}}$$

Proof Rule: Sequencing

$$\frac{\{R\}\begin{array}{c}C_1\\C_2\end{array}\{R'\} \qquad \{R'\}\begin{array}{c}C_1'\\C_2'\end{array}\{S\}}{\{R\}\begin{array}{c}C_1; C_1'\\C_2; C_2'\end{array}\{S\}}$$

Whenever it is necessary, we use the fact that skip is the identity for sequencing:

$$C \equiv \mathsf{skip}; C \equiv C; \mathsf{skip}$$

Proof Rule: Conditional

$$\frac{R \Rightarrow (B_1 \Leftrightarrow B_2) \qquad \{R \wedge B_1 \wedge B_2\} \begin{matrix} C_1 \\ C_2 \end{matrix} \{S\} \qquad \{R \wedge \neg B_1 \wedge \neg B_2\} \begin{matrix} C_1' \\ C_2' \end{matrix} \{S\}}{\{R\} \begin{matrix} \textsf{if } B_1 \textsf{ then } C_1 \textsf{ else } C_1' \\ \textsf{if } B_2 \textsf{ then } C_2 \textsf{ else } C_2' \end{matrix} \{S\}}$$

Proof Rule: Loop

$$\frac{R \Rightarrow (B_1 \Leftrightarrow B_2) \qquad \{R \wedge B_1 \wedge B_2\} \begin{matrix} C_1 \\ C_2 \end{matrix} \{R\}}{\{R\} \begin{matrix} \mathsf{while}\ B_1\ \mathsf{do}\ C_1\ \mathsf{od} \\ \mathsf{while}\ B_2\ \mathsf{do}\ C_2\ \mathsf{od} \end{matrix} \{R \wedge \neg B_1 \wedge \neg B_2\}}$$

The condition of the rule implies that the one while-loop may diverge iff the other while-loop may diverge.

Structural Rules

Frame Rule

$(\text{Modifies}(C_1, C_2) \cap \text{FV}(R') = \emptyset)$

$$\frac{\{R\}\overset{C_1}{\underset{C_2}{}}\{S\}}{\{R * R'\}\overset{C_1}{\underset{C_2}{}}\{S * R'\}}$$

Consequence

$$\frac{R' \Rightarrow R \quad \{R\}\overset{C_1}{\underset{C_2}{}}\{S\} \quad S \Rightarrow S'}{\{R'\}\overset{C_1}{\underset{C_2}{}}\{S'\}}$$

Conjunction

$$\frac{\{R\}\overset{C_1}{\underset{C_2}{}}\{S\} \quad \{R'\}\overset{C_1}{\underset{C_2}{}}\{S'\}}{\{R \wedge R'\}\overset{C_1}{\underset{C_2}{}}\{S \wedge S'\}}$$

Auxiliary Variable Elimination

$(x \notin \text{FV}(C_1, C_2))$

$$\frac{\{R\}\overset{C_1}{\underset{C_2}{}}\{S\}}{\{\exists x.\, R\}\overset{C_1}{\underset{C_2}{}}\{\exists x.\, S\}}$$

Specification of Schorr-Waite Marking Algorithm

$$\{\mathsf{Same} \wedge p{=}p' \wedge \mathsf{NoDangling}(p,p')\} \begin{array}{c}\mathsf{SWMarking}(p)\\ \mathsf{StackMarking}(p')\end{array} \{\mathsf{Same} \wedge \mathsf{NoDangling}(p,p')\}$$

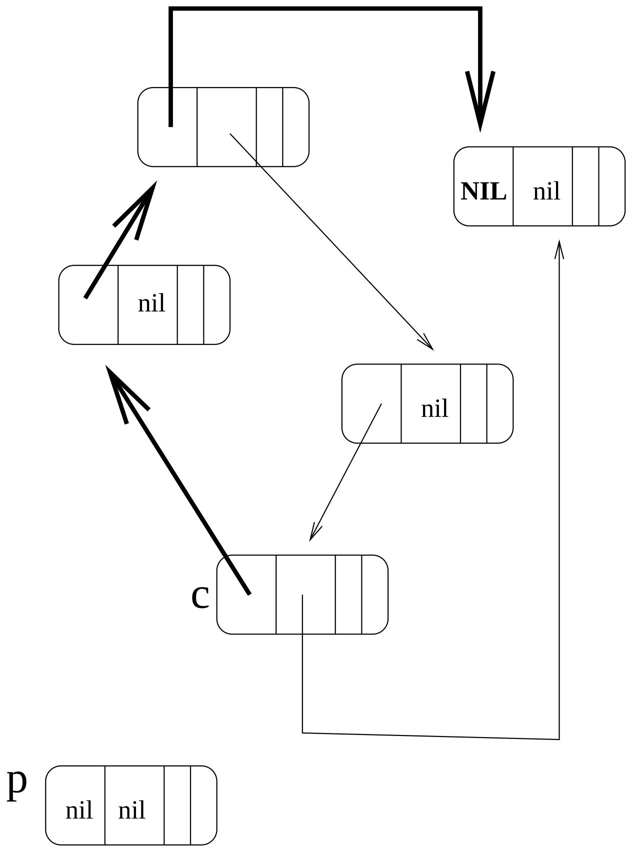
where

$$\mathsf{Same} \stackrel{\Delta}{\equiv} \begin{pmatrix}\mathsf{emp}\\ \mathsf{emp}\end{pmatrix} \vee (\exists x, l, r, m, d.\ \begin{pmatrix}x \mapsto l, r, m, d\\ x \mapsto l, r, m, d\end{pmatrix}) * \mathsf{Same}$$

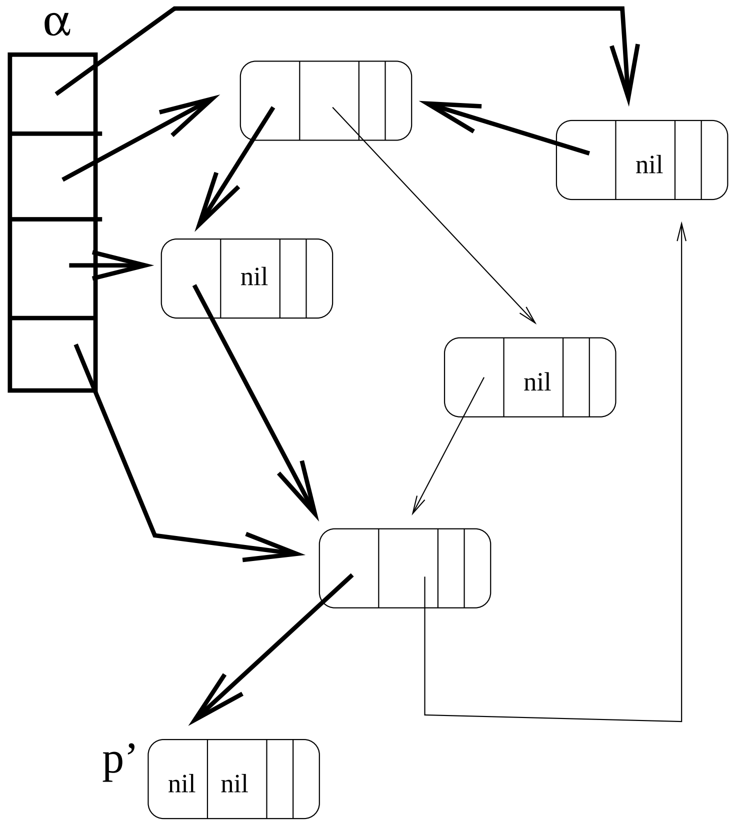
Programs

```
c := p;
if p!=nil then p:=p.1;
while (c!=nil)
  do if (p!=nil) then m := p.3
                 else m := marked;
      if (p!=nil /\ m!=marked)
      then
        t := p.1;
        p.1 := c;
        c := p;
        p := t;
        c.3 := marked;
        c.4 := left
      else r := c.4;
           if (r=left)
           then
             t := c.1;
             c.1 := p;
             p := c.2;
             c.2 := t;
             c.4 := right
           else
             t := p;
             p := c;
             c := c.2;
             p.2 := t
```

```
a := [(p',left)];
if p'!=nil then p':=p'.1;
while (a != [])
    do if (p'!=nil) then m':=p'.3
                    else m':=marked;
       if (p'!=nil /\ m'!=marked)
       then
         a := (p',left):a;
         p'.3 := marked;
         p'.4 := left;
         p' := p'.1


       else if (#2(hd a)=left)
            then
              #1(hd a).4 := right;
              p' := #1(hd a).2;
              a := (#1(hd a),right):(tl a)



            else
              p' := #1(hd a);
              a := tl a
```



SWMarking Algorithm



StackMarking Algorithm

Invariant Relation

The invariant relation is:

$$\mathsf{Same} * (\mathsf{Stack}\ p\ c\ \alpha) * p \mathrel{\overset{\circ}{=}} p' \wedge \begin{pmatrix} \mathsf{NoDanglingSW}(p, c) \\ \mathsf{NoDanglingStack}(p', \alpha) \end{pmatrix}$$

where

$$E \mathrel{\overset{\circ}{=}} E' \ \stackrel{\Delta}{\equiv}\ E = E' \wedge \mathsf{Emp}$$

$$\mathsf{Stack}\ p\ c\ [] \ \stackrel{\Delta}{\equiv}\ (c \mathrel{\overset{\circ}{=}} \mathsf{nil})$$

$$\mathsf{Stack}\ p\ c\ (x, \mathsf{left}){:}\alpha \ \stackrel{\Delta}{\equiv}\ \exists n, r.\ c \mathrel{\overset{\circ}{=}} x * \begin{pmatrix} c \mapsto n, r, \mathsf{marked}, \mathsf{left} \\ x \mapsto p, r, \mathsf{marked}, \mathsf{left} \end{pmatrix} * \mathsf{Stack}\ c\ n\ \alpha$$

$$\mathsf{Stack}\ p\ c\ (x, \mathsf{right}){:}\alpha \ \stackrel{\Delta}{\equiv}\ \exists n, l.\ c \mathrel{\overset{\circ}{=}} x * \begin{pmatrix} c \mapsto l, n, \mathsf{marked}, \mathsf{right} \\ x \mapsto l, p, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ c\ n\ \alpha$$

Verification of Pop

We like to show:

$$\{\mathsf{Same} * \mathsf{Stack}\, p\, c\, \alpha * p \mathrel{\overset{\circ}{=}} p' * \alpha \mathrel{\overset{\circ}{=}} (c, \mathsf{right}){:}\alpha_0 \wedge \begin{pmatrix} \mathsf{NoDanglingSW}(p, c) \wedge c{\neq}\mathsf{nil} \\ \mathsf{NoDanglingStack}(p', \alpha) \wedge \alpha{\neq}[] \end{pmatrix}\}$$

$$\mathsf{SWPop}(p, c) \qquad\qquad \mathsf{StackPop}(p', \alpha)$$

$$\{\mathsf{Same} * \mathsf{Stack}\ p\ c\ \alpha * p \mathrel{\overset{\circ}{=}} p' \wedge \begin{pmatrix} \mathsf{NoDanglingSW}(p, c) \\ \mathsf{NoDanglingStack}(p', \alpha) \end{pmatrix}\}$$

Assuming:

$$\{\mathsf{Stack}\, p\, c\, \alpha * p \stackrel{\circ}{=} p' * \alpha \stackrel{\circ}{=} (c, \mathsf{right}){:}\alpha_0\} \begin{array}{c} \mathsf{SWPop}(p, c) \\ \mathsf{StackPop}(p', \alpha) \end{array} \{\mathsf{Stack}\, p\, c\, \alpha * p \stackrel{\circ}{=} p' * \mathsf{Same}\}$$

we can write the following proof outline:

$$\begin{array}{l}
\{\mathsf{Stack}\ p\ c\ \alpha * p \stackrel{\circ}{=} p' * \alpha \stackrel{\circ}{=} (c, \mathsf{right}){:}\alpha_0 * \mathsf{Same}\} \\
\left[\begin{array}{ll}
\multicolumn{2}{l}{\{\mathsf{Stack}\ p\ c\ \alpha * p \stackrel{\circ}{=} p' * \alpha \stackrel{\circ}{=} (c, \mathsf{right}){:}\alpha_0\}} \\
\mathsf{SWPop}(p, c) & \mathsf{StackPop}(p', \alpha) \\
\multicolumn{2}{l}{\{\mathsf{Stack}\ p\ c\ \alpha * p \stackrel{\circ}{=} p' * \mathsf{Same}\}}
\end{array}\right] \text{Frame Rule} \\
\{\mathsf{Stack}\ p\ c\ \alpha * p \stackrel{\circ}{=} p' * \mathsf{Same} * \mathsf{Same}\} \\
\{\mathsf{Stack}\ p\ c\ \alpha * p \stackrel{\circ}{=} p' * \mathsf{Same}\}
\end{array}$$

The last step uses the fact that $\mathsf{Same} * \mathsf{Same} \Rightarrow \mathsf{Same}$.

Since the following triples hold:

$$[\mathsf{NoDanglingSW}(p,c) \wedge c{\neq}\mathsf{nil}]\ \mathsf{SWPop}(p,c)\ [\mathsf{NoDanglingSW}(p,c)]$$

$$[\mathsf{NoDanglingStack}(p',\alpha) \wedge \alpha{\neq}[]]\ \mathsf{StackPop}(p',\alpha)\ [\mathsf{NoDanglingStack}(p',\alpha)]$$

we have:

$$\left\{\begin{pmatrix}\mathsf{NoDanglingSW}(p,c) \wedge c{\neq}\mathsf{nil} \\ \mathsf{NoDanglingStack}(p',\alpha) \wedge \alpha{\neq}[]\end{pmatrix}\right\}\begin{matrix}\mathsf{SWPop}(p,c) \\ \mathsf{StackPop}(p',\alpha)\end{matrix}\left\{\begin{pmatrix}\mathsf{NoDanglingSW}(p,c) \\ \mathsf{NoDanglingStack}(p',\alpha)\end{pmatrix}\right\}$$

By combining the two quadruples, we obtain the conclusion:

$$\left\{\mathsf{Same} * \mathsf{Stack}\,p\,c\,\alpha * p \stackrel{\circ}{=} p' * \alpha \stackrel{\circ}{=} (c,\mathsf{right}){:}\alpha_0 \wedge \begin{pmatrix}\mathsf{NoDanglingSW}(p,c) \wedge c{\neq}\mathsf{nil} \\ \mathsf{NoDanglingStack}(p',\alpha) \wedge \alpha{\neq}[]\end{pmatrix}\right\}$$

$$\mathsf{SWPop}(p,c) \qquad\qquad \mathsf{StackPop}(p',\alpha)$$

$$\left\{\mathsf{Same} * \mathsf{Stack}\ p\ c\ \alpha * p \stackrel{\circ}{=} p' \wedge \begin{pmatrix}\mathsf{NoDanglingSW}(p,c) \\ \mathsf{NoDanglingStack}(p',\alpha)\end{pmatrix}\right\}$$

Discharging the Assumption

Still need to show:

$$\{\mathsf{Stack}\, p\, c\, \alpha * p \mathrel{\stackrel{\circ}{=}} p' * \alpha \mathrel{\stackrel{\circ}{=}} (c, \mathsf{right}){:}\alpha_0\} \begin{array}{c}\mathsf{SWPop}(p, c)\\ \mathsf{StackPop}(p', \alpha)\end{array} \{\mathsf{Stack}\, p\, c\, \alpha * p \mathrel{\stackrel{\circ}{=}} p' * \mathsf{Same}\}$$

Proof Outline

$$\{\mathsf{Stack}\ p\ c\ \alpha * p \mathring{=} p' * \alpha \mathring{=} (c, \mathsf{right}){:}\alpha_0\}$$

$$\{\mathsf{Stack}\ p\ c\ (c, \mathsf{right}){:}\alpha_0 * p \mathring{=} p' * \alpha \mathring{=} (c, \mathsf{right}){:}\alpha_0\}$$

$$\{\exists l_0, n_0.\ \begin{pmatrix} c \mapsto l_0, n_0, \mathsf{marked}, \mathsf{right} \\ c \mapsto l_0, p, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ c\ n_0\ \alpha_0 * \alpha \mathring{=} (c, \mathsf{right}){:}\alpha_0\}$$

$$\{\begin{pmatrix} c \mapsto l_0, n_0, \mathsf{marked}, \mathsf{right} \\ c \mapsto l_0, p, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ c\ n_0\ \alpha_0 * \alpha \mathring{=} (c, \mathsf{right}){:}\alpha_0\}$$

$t := p;$ skip;

$$\{\begin{pmatrix} c \mapsto l_0, n_0, \mathsf{marked}, \mathsf{right} \\ c \mapsto l_0, t, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ c\ n_0\ \alpha_0 * \alpha \mathring{=} (c, \mathsf{right}){:}\alpha_0\}$$

$p := c;$ skip;

$$\{\begin{pmatrix} p \mapsto l_0, n_0, \mathsf{marked}, \mathsf{right} \\ p \mapsto l_0, t, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ p\ n_0\ \alpha_0 * \alpha \mathring{=} (p, \mathsf{right}){:}\alpha_0\}$$

$$\{\begin{pmatrix} p \mapsto l_0, n_0, \mathsf{marked}, \mathsf{right} \\ p \mapsto l_0, t, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ p\ n_0\ \alpha_0 * \alpha \stackrel{\circ}{=} (p, \mathsf{right}){:}\alpha_0\}$$

$c := p.2;$ $\qquad\qquad$ $\mathsf{skip};$

$$\{\begin{pmatrix} p \mapsto l_0, c, \mathsf{marked}, \mathsf{right} \\ p \mapsto l_0, t, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ p\ c\ \alpha_0 * \alpha \stackrel{\circ}{=} (p, \mathsf{right}){:}\alpha_0\}$$

$p.2 := t;$ $\qquad\qquad$ $\mathsf{skip};$

$$\{\begin{pmatrix} p \mapsto l_0, t, \mathsf{marked}, \mathsf{right} \\ p \mapsto l_0, t, \mathsf{marked}, \mathsf{right} \end{pmatrix} * \mathsf{Stack}\ p\ c\ \alpha_0 * \alpha \stackrel{\circ}{=} (p, \mathsf{right}){:}\alpha_0\}$$

$\{\mathsf{Same} * \mathsf{Stack}\ p\ c\ (\mathsf{tl}\ \alpha) * \alpha \stackrel{\circ}{=} (p, \mathsf{right}){:}\alpha_0\}$

skip $\qquad\qquad$ $p' := \#1(\mathsf{hd}\ \alpha);$

$\{p \stackrel{\circ}{=} p' * \mathsf{Same} * \mathsf{Stack}\ p\ c\ (\mathsf{tl}\ \alpha) * \alpha \stackrel{\circ}{=} (p, \mathsf{right}){:}\alpha_0\}$

skip $\qquad\qquad$ $\alpha := \mathsf{tl}\ \alpha;$

$\{p \stackrel{\circ}{=} p' * \mathsf{Same} * \mathsf{Stack}\ p\ c\ \alpha\}$

Conclusion

- When the two programs have similar structures, the proof rules for the quadruples are useful.
- The proof rules for Hoare quadruples are incomplete.
- It is still necessary to prove the correctness of the "more abstract program."