Program Verification using Separation Logic

# Lecture 1 :
# Proof Rules in Separation Logic

Hongseok Yang (Queen Mary, Univ. of London)

# Automatic Verifier

- Run a program symbolically and approximately.

$\boxed{\text{emp}}$

```
x = create_list();
y = create_list();
z = create_sorted_list();
x = append_list(x,y);
free_list(x);
traverse_list(z);
```

# Automatic Verifier

- Run a program symbolically and approximately.

emp

```
x = create_list();
```

ls(x,0)

```
y = create_list();
z = create_sorted_list();
x = append_list(x,y);
free_list(x);
traverse_list(z);
```

# Automatic Verifier

- Run a program symbolically and approximately.

```
          emp
x = create_list();  ls(x,0)
y = create_list();  ls(x,0) * ls(y,0)
z = create_sorted_list();
x = append_list(x,y);
free_list(x);
traverse_list(z);
```

# Automatic Verifier

- Run a program symbolically and approximately.

emp

```
x = create_list();
```
ls(x,0)

```
y = create_list();
```
ls(x,0) * ls(y,0)

```
z = create_sorted_list();
```
ls(x,0) * ls(y,0) * ls(z,0)

```
x = append_list(x,y);
free_list(x);
traverse_list(z);
```

# Automatic Verifier

- Run a program symbolically and approximately.

```
                    emp
x = create_list();      ls(x,0)
y = create_list();      ls(x,0) * ls(y,0)
z = create_sorted_list();   ls(x,0) * ls(y,0) * ls(z,0)
x = append_list(x,y);   ls(x,y) * ls(y,0) * ls(z,0)
free_list(x);
traverse_list(z);
```

# Automatic Verifier

- Run a program symbolically and approximately.

emp

```
x = create_list();
```

ls(x,0)

```
y = create_list();
```

ls(x,0) * ls(y,0)

```
z = create_sorted_list();
```

ls(x,0) * ls(y,0) * ls(z,0)

```
x = append_list(x,y);
```

ls(x,y) * ls(y,0) * ls(z,0)

```
free_list(x);
```

ls(z,0)

```
traverse_list(z);
```

# Automatic Verifier

- Run a program symbolically and approximately.

```
                 emp
x = create_list();            ls(x,0)
y = create_list();            ls(x,0) * ls(y,0)
z = create_sorted_list();     ls(x,0) * ls(y,0) * ls(z,0)
x = append_list(x,y);         ls(x,y) * ls(y,0) * ls(z,0)
free_list(x);                 ls(z,0)
traverse_list(z);             ls(z,0)
```

# Automatic Verifier

- Run a program symbolically and approximately.

emp

```
x = create_list();
```
ls(x,0)

```
y = create_list();
```
ls(x,0) * ls(y,0)

```
z = create_sorted_list();
```
ls(x,0) * ls(y,0) * ls(z,0)

```
x = a
free_
trave
```

Three major components of a auto. verifier.
1) Symbolic representation of state sets.
2) Symbolic execution of programs.
3) Technique for losing information sensibly.

# Automatic Verifier

- Run a program symbolically and approximately.

```
x = c
y = c
z = c
x = a
free_
trave
```

Help from sep. logic.
1) Assertion lang. with *.
2) Proof rules for programs.
3) Sound implication between assertions.

Three major components of a auto. verifier.
1) Symbolic representation of state sets.
2) Symbolic execution of programs.
3) Technique for losing information sensibly.

# Automatic Verifier

- Run a program symbolically and approximately.

Help from sep. logic. 
1) Assertion lang. with *. 
2) Proof rules for programs. 
3) Sound implication between assertions.

Three major components of a auto. verifier. 
1) Symbolic representation of state sets. 
2) Symbolic execution of programs. 
3) Technique for losing information sensibly.

# Today's Goal

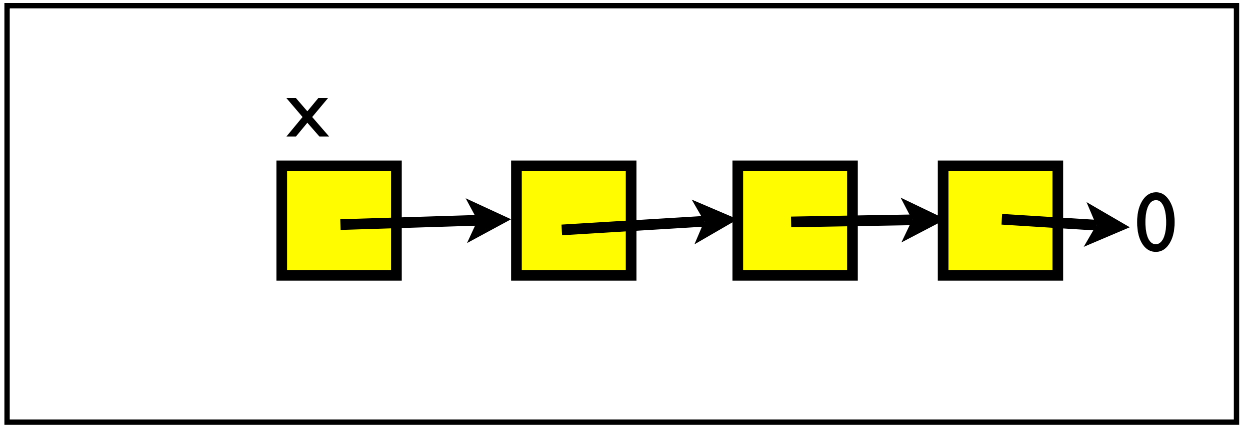
- Study proof rules of sep. logic, which have been used to implement symbolic execution.
- Prove the safety of the list reversal.

# Verification of List Reversal

```
y = 0;
while (x != 0) {
   t = x;
   x = *t;
   *t = y;
   y = t;
}
```

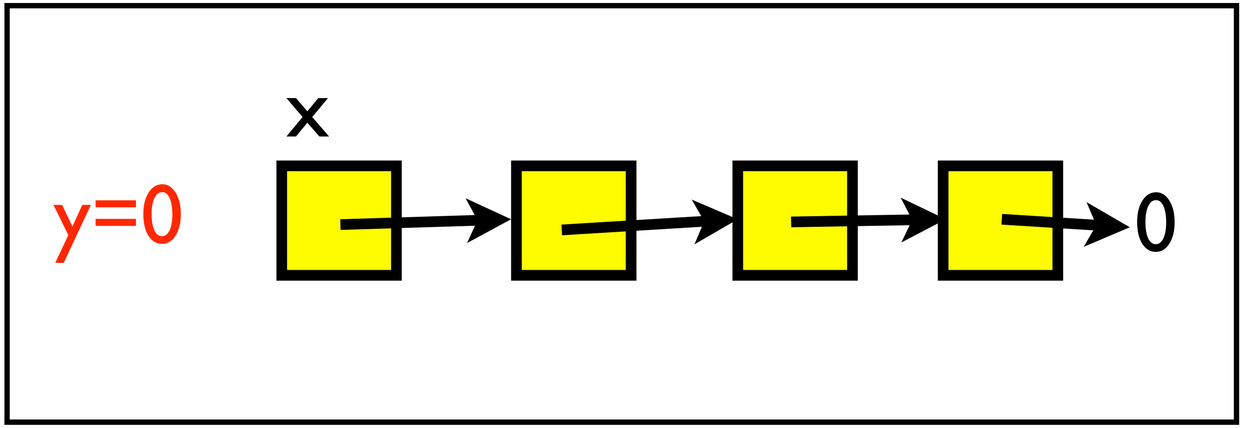
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



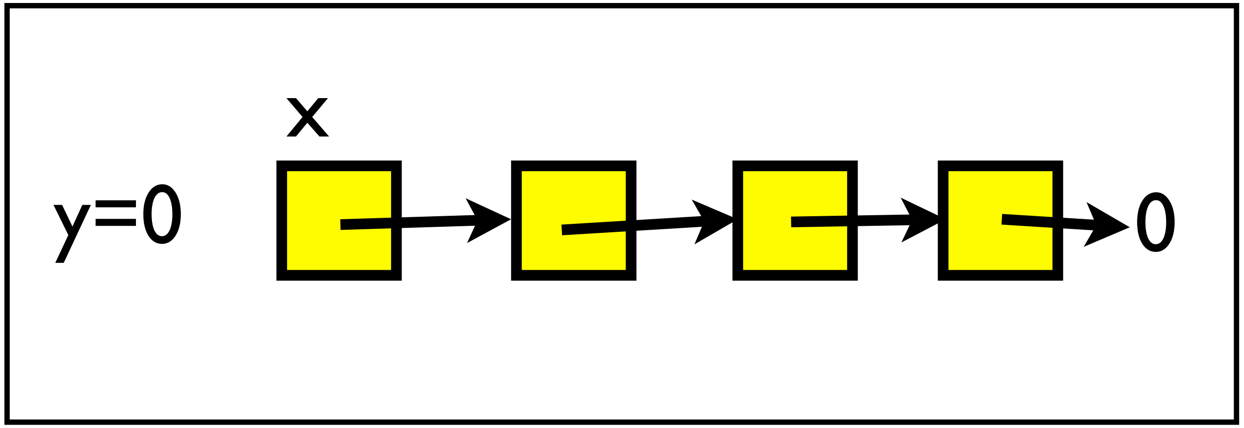
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



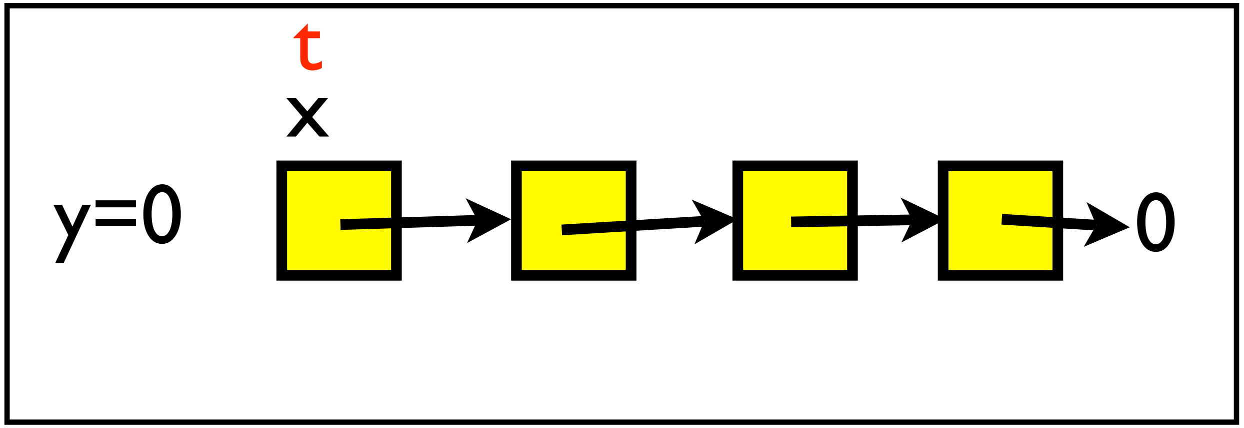
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



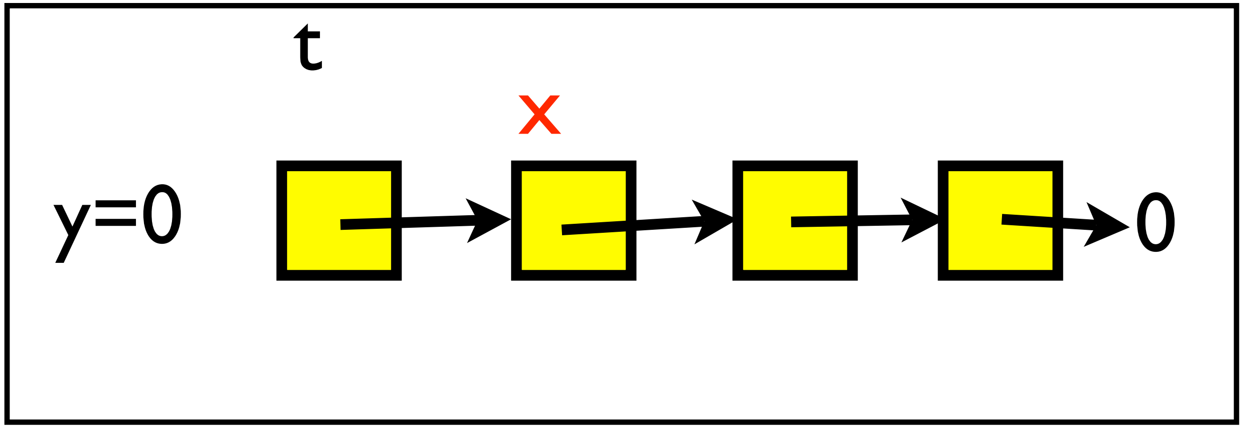
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



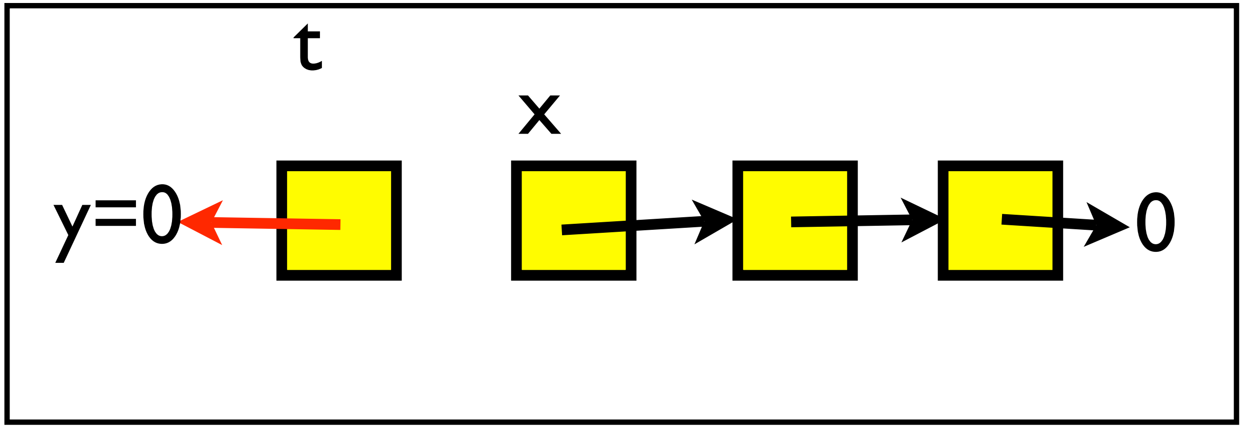
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



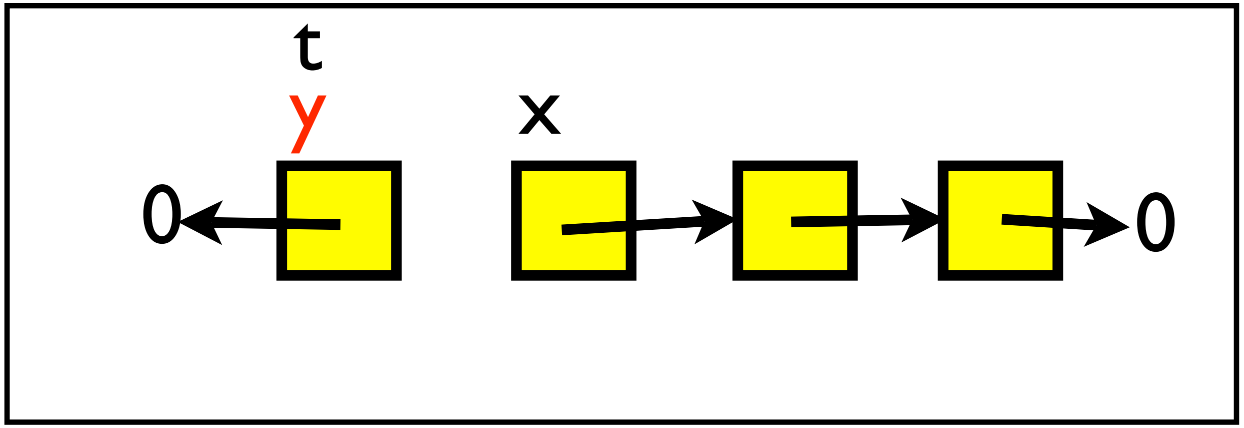
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



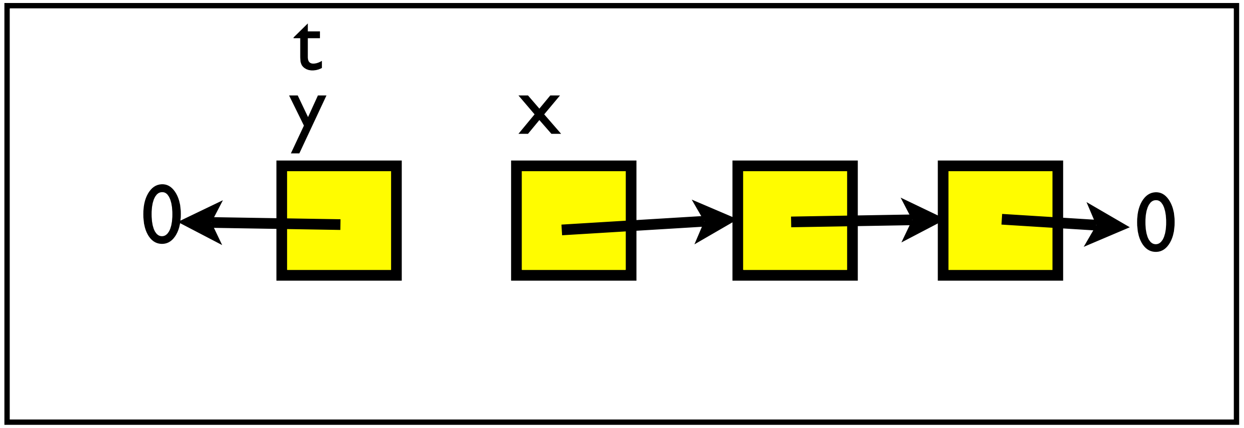
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



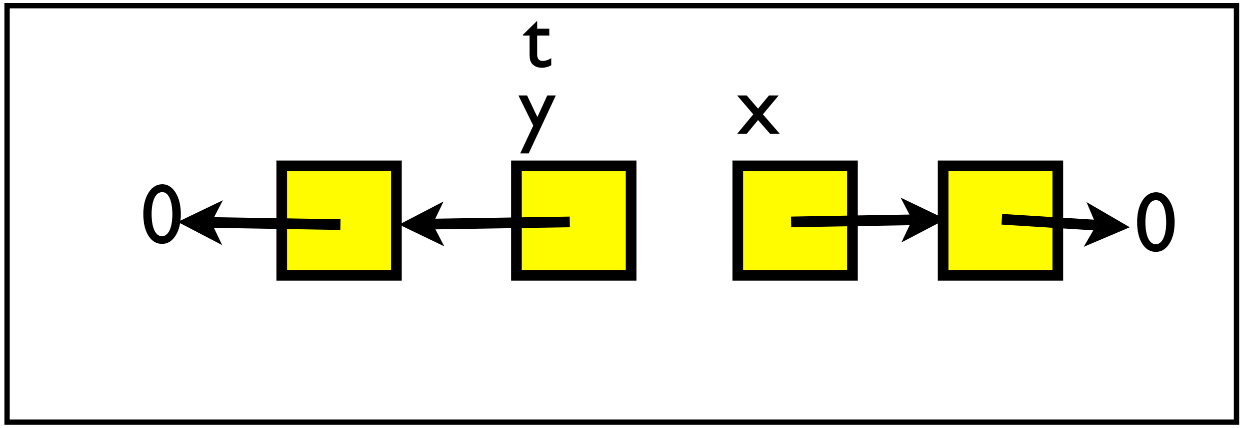
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



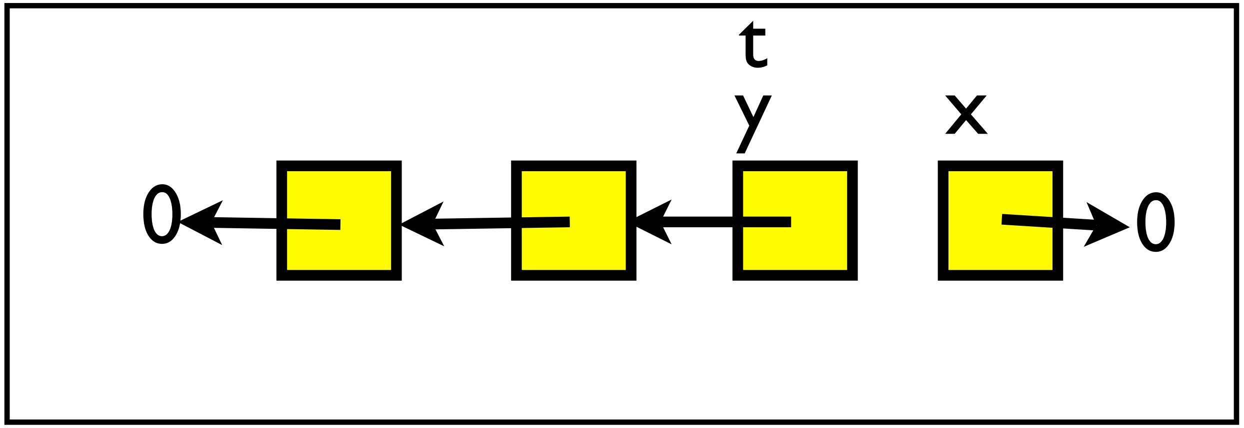
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



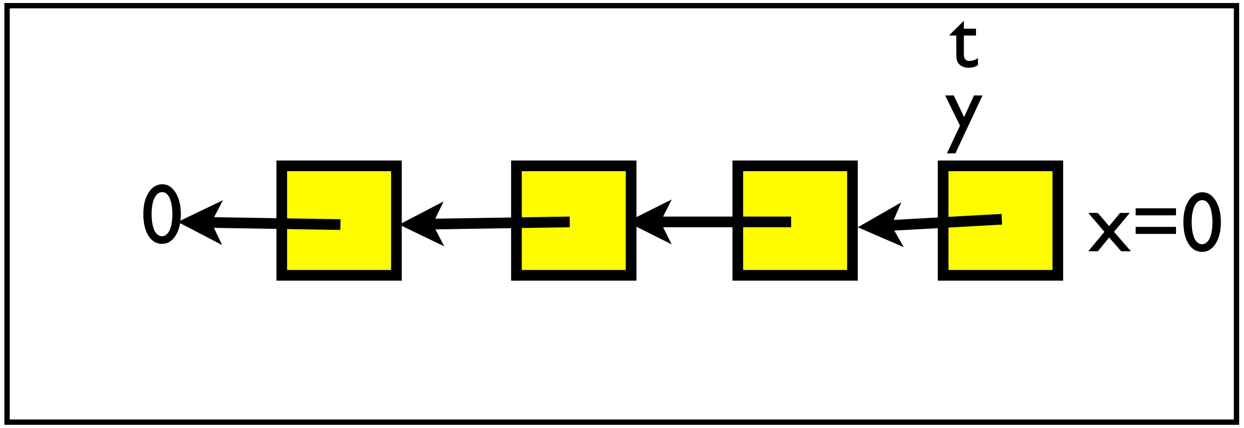
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



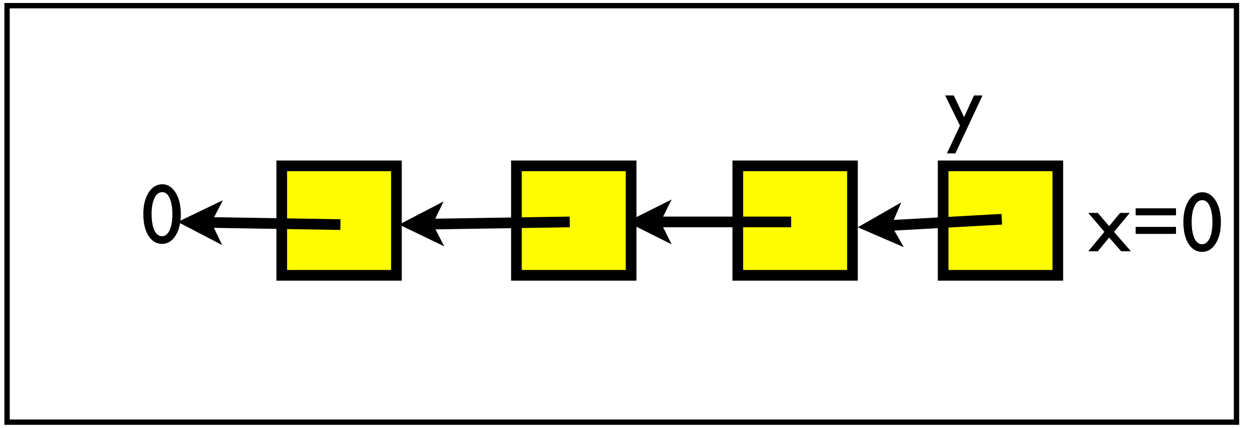
# Verification of List Reversal

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```



# Verification of List Reversal

```
y = 0;
while (x != 0) {
   t = x;
   x = *t;
   *t = y;
   y = t;
}
```



# Verification of List Reversal

$\boxed{\text{ls } \alpha \ (x,0)}$

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```

$\boxed{\text{ls } (\text{rev}(\alpha)) \ (y,0)}$

# Verification of List Reversal

$\boxed{\text{ls } \alpha\ (x,0)}$

```
y = 0;
while (x != 0) {
    t = x;
    x = *t;
    *t = y;
    y = t;
}
```

$\boxed{\text{ls } (\text{rev}(\alpha))\ (y,0)}$

$\boxed{\text{ls } (x,0)}$

$\boxed{\text{ls } (y,0)}$

Will prove this simpler shape property.

# Simple Imperative Language

$$
\begin{array}{rcl}
B & ::= & E = E \mid E \geq E \mid B \wedge B \mid B \vee B \mid \neg B \\
C, D & ::= & x = \texttt{new}(E) \mid x = *E \mid *E = E \mid \texttt{free}(E) \\
& \mid & x = E \mid C; C \mid \texttt{if}\ B\ C\ C \mid \texttt{while}\ B\ C
\end{array}
$$

- Explicit heap access with *E.
- Booleans and expressions do not access heap cells.
- E.g. x=new(3); *x=0; *(x+1)=0; *(x+2)=0
- Q: Write a program that swaps the values of cells x,y.
- Use two temporary vars. What about none?

Sol1) t1=*x; t2=*y; *x=t2; *y=t1
"Sol2") x=XOR(x,y); y=XOR(x,y); x= XOR(x,y)

$$
\begin{array}{rcl}
B & ::= & E = E \mid E \geq E \mid B \wedge B \mid B \vee B \mid \neg B \\
C, D & ::= & x = \texttt{new}(E) \mid x = *E \mid *E = E \mid \texttt{free}(E) \\
& \mid & x = E \mid C; C \mid \texttt{if}\ B\ C\ C \mid \texttt{while}\ B\ C
\end{array}
$$

- Explicit heap access with *E.
- Booleans and expressions do not access heap cells.
- E.g. x=new(3); *x=0; *(x+1)=0; *(x+2)=0
- Q: Write a program that swaps the values of cells x,y.
- Use two temporary vars. What about none?

# Semantics of Programs

$$
\begin{aligned}
[\![B]\!] &: \mathsf{Stacks} \to \mathsf{Vals} \\
[\![C]\!] &: \mathsf{States} \to \mathcal{P}(\mathsf{States} \cup \{\mathsf{err}\})
\end{aligned}
$$

- err occurs when a command accesses an unallocated heap cell.
- So, no err means that the initial heap has enough cells.

$$
[\![*E{=}F]\!](s,h) \stackrel{def}{=} \begin{cases} \{(s,\ h[[\![E]\!]s{\mapsto}[\![F]\!]s])\} & \text{if } [\![E]\!]s \in \mathsf{dom}(h) \\ \{\mathsf{err}\} & \text{otherwise} \end{cases}
$$

$$
[\![x{=}\mathtt{new}(2)]\!](s,h) \stackrel{def}{=} \{(s, h * h') \mid h\#h' \wedge \mathsf{dom}(h'){=}\{l, l{+}1\} \text{ for some } l\}
$$

$$
\begin{aligned}
[\![C_1; C_2]\!](s,h) \stackrel{def}{=} &\ \{(s'', h'') \mid (s', h') \in [\![C_1]\!](s,h) \ \wedge\ (s'', h'') \in [\![C_2]\!](s', h')\} \\
&\cup \{\mathsf{err} \mid \mathsf{err} \in [\![C_1]\!](s,h)\} \\
&\cup \{\mathsf{err} \mid (s', h') \in [\![C_1]\!](s,h) \ \wedge\ \mathsf{err} \in [\![C_2]\!](s', h')\}
\end{aligned}
$$

# Hoare Triple {P}C{Q}

- P is called precondition and Q postcondition.
- {P}C{Q} holds (or is valid) if and only if

$$\forall(s,h).\ \text{if } (s,h) \models P, \text{then } 1)\ \text{err} \notin [\![C]\!](s,h) \text{ and}$$
$$2)\ \forall(s',h') \in [\![C]\!](s,h).\ (s',h') \models Q.$$

- Let $E \mapsto \_ \overset{def}{=} \exists x. E \mapsto x.$

| | |
|---|---|
| $\{x \mapsto \_\}{*}x = 0\{x \mapsto 0\}$ | Valid |
| $\{x \mapsto \_\}{*}x = 0\{x \mapsto 1\}$ | Invalid |
| $\{\text{true}\}{*}x = 0\{x \mapsto 0\}$ | Invalid |

# Structural Rules

- Rules that are independent of language constructs.

Consequence

$$\frac{P \Rightarrow P' \quad \{P'\}C\{Q'\} \quad Q' \Rightarrow Q}{\{P\}C\{Q\}}$$

Existential Elimination

$$\frac{\{P\}C\{Q\}}{\{\exists x.\, P\}C\{\exists x.\, Q\}}\ x \notin \mathsf{FV}(C)$$

Disjunction

$$\frac{\{P\}C\{Q\} \quad \{P'\}C\{Q'\}}{\{P \vee P'\}C\{Q \vee Q'\}}$$

- E.g.

$$\cfrac{\cfrac{\{\alpha{\neq}\epsilon \wedge \mathsf{ls}\,\alpha\,(x,0)\}C\{\alpha{\neq}\epsilon \wedge \mathsf{ls}\,\alpha^{\mathsf{rev}}\,(x,0)\}}{\{\exists\alpha.\ \alpha{\neq}\epsilon \wedge \mathsf{ls}\,\alpha\,(x,0)\}C\{\exists\alpha.\ \alpha{\neq}\epsilon \wedge \mathsf{ls}\,\alpha^{\mathsf{rev}}\,(x,0)\}}\ \text{Exist.}}{\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}C\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}}\ \text{Conseq.}$$

$$\frac{\{x{=}0 \wedge \mathsf{emp}\}\mathtt{rev}\{y{=}0 \wedge \mathsf{emp}\} \quad \{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}\mathtt{rev}\{y{\neq}0 \wedge \mathsf{ls}\,(y,0)\}}{\{(x{=}0 \wedge \mathsf{emp}) \vee (x{\neq}0 \wedge \mathsf{ls}\,(x,0))\}\mathtt{rev}\{(y{=}0 \wedge \mathsf{emp}) \vee (y{\neq}0 \wedge \mathsf{ls}\,(y,0))\}}\ \text{Disj.}$$

# Principle behind Rules to Come

- Will show many rules.
  - Rule for each language construct.
- But, simple principles:
  - The rules implement symbolic execution.
  - They ensure the absence of null/dangling references.

# Proof Rules for C;C', *E=F, free(E)

$$\frac{\{P\}C\{P'\} \quad \{P'\}C'\{Q\}}{\{P\}C;C'\{Q\}}$$

$$\overline{\{P * E \mapsto E_0\}{*E{=}F}\{P * E \mapsto F\}} \qquad \overline{\{P * E \mapsto E_0\}\mathtt{free}(E)\{P\}}$$

- E.g.

$$\frac{\overline{\{x{\mapsto}x' * y{\mapsto}3\}{*x{=}0}\{x{\mapsto}0 * y{\mapsto}3\}} \quad \overline{\{x{\mapsto}0 * y{\mapsto}3\}\mathtt{free}(y)\{x{\mapsto}0\}}}{\{x{\mapsto}x' * y{\mapsto}3\}{*x{=}0};\mathtt{free}(y)\{x{\mapsto}0\}}$$

# Proof Rules for C;C', *E=F, free(E)

$$\frac{\{P\}C\{P'\} \quad \{P'\}C'\{Q\}}{\{P\}C;C'\{Q\}}$$

$$\frac{}{\{P * E \mapsto E_0\} {*E{=}F} \{P * E \mapsto F\}} \qquad \frac{}{\{P * E \mapsto E_0\}\texttt{free}(E)\{P\}}$$

- E.g.

$$\frac{\dfrac{}{\{x{\mapsto}x' * y{\mapsto}3\}{*x{=}0}\{x{\mapsto}0 * y{\mapsto}3\}} \quad \dfrac{}{\{x{\mapsto}0 * y{\mapsto}3\}\texttt{free}(y)\{x{\mapsto}0\}}}{\{x{\mapsto}x' * y{\mapsto}3\}{*x{=}0};\texttt{free}(y)\{x{\mapsto}0\}}$$

Proof Sketch:

$\{x \mapsto x' * y \mapsto 3\}$
$*x{=}0;$
$\{x \mapsto 0 * y \mapsto 3\}$
$\texttt{free}(y)$
$\{x \mapsto 0\}$

# Proof Rules for C;C', *E=F, free(E)

$$\frac{\{P\}C\{P'\} \quad \{P'\}C'\{Q\}}{\{P\}C;C'\{Q\}}$$

$$\overline{\boxed{\{P * E \mapsto E_0\}} *E{=}F\{P * E \mapsto F\}} \qquad \overline{\boxed{\{P * E \mapsto E_0\}}\mathtt{free}(E)\{P\}}$$

- E.g.

Preconditions of particular forms.
Ensures the safe execution of command.
Yet, flexible because of the structural rules.

Proof Sketch: $\{x \mapsto x' * y \mapsto 3\}$
$*x{=}0;$
$\{x \mapsto 0 * y \mapsto 3\}$
$\mathtt{free}(y)$
$\{x \mapsto 0\}$

# Proof Rules for x=...

- Assume that x does not appear in P, E, F.

$$\frac{}{\{P\}x{=}E\{x{=}E \wedge P\}} \quad \frac{}{\{P\}x = \texttt{new}(1)\{\exists x_0'.\, P * x{\mapsto}x_0'\}}$$

$$\frac{}{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

- The general rules use x' to denote the old value of x.

$$\frac{}{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \quad \frac{}{\{P\}x = \texttt{new}(1)\{\exists x' x_0'.\, P[x'/x] * x \mapsto x_0'\}}$$

$$\frac{}{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

# Proof Rules f

$\{x \mapsto x' * y \mapsto y'\}$

$t_1 = *x;$

$t_2 = *y;$

$*x = t_2;$

$*y = t_1$

$\{x \mapsto y' * y \mapsto x'\}$

- Assume that x does not appe

$$\overline{\{P\}x{=}E\{x{=}E \wedge P\}} \qquad \overline{\{P\}x = \texttt{new}(1)\{\exists}$$

$$\overline{\{P * E \mapsto F\}x{=}{*}E\{x{=}F \wedge (P * E \mapsto F)\}}$$

- The general rules use x' to denote the old value of x.

$$\overline{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \qquad \overline{\{P\}x = \texttt{new}(1)\{\exists x' x_0'.\, P[x'/x] * x \mapsto x_0'\}}$$

$$\overline{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E \mapsto F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x \mapsto x' * y \mapsto y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x \mapsto y' * y \mapsto x'\}$$

# Proof Rules f

- Assume that x does not appe

$$\frac{}{\{P\}x{=}E\{x{=}E \wedge P\}} \quad \frac{}{\{P\}x = \mathtt{new}(1)\{\exists}$$

$$\frac{}{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

$$\{x{\mapsto}x' * y{\mapsto}y'\}$$
$$\color{red}{t_1{=}{*}x;}$$
$$\{t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$$
$$\color{red}{t_2{=}{*}y;}$$
$$\color{red}{{*}x{=}t_2;}$$
$$\color{red}{{*}y{=}t_1}$$
$$\{x{\mapsto}y' * y{\mapsto}x'\}$$

- The general rules use x' to denote the old value of x.

$$\frac{}{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \quad \frac{}{\{P\}x = \mathtt{new}(1)\{\exists x' x'_0.\, P[x'/x] * x \mapsto x'_0\}}$$

$$\frac{}{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

# Proof Rules f

$$\{x{\mapsto}x' * y{\mapsto}y'\}$$
$$t_1{=}{*}x;$$
$$\{t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$$
$$t_2{=}{*}y;$$
$$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$$
$${*}x{=}t_2;$$

$${*}y{=}t_1$$

$$\{x{\mapsto}y' * y{\mapsto}x'\}$$

- Assume that x does not appe

$$\frac{}{\{P\}x{=}E\{x{=}E \wedge P\}} \qquad \frac{}{\{P\}x = \mathtt{new}(1)\{\exists}$$

$$\frac{}{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

- The general rules use x' to denote the old value of x.

$$\frac{}{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \qquad \frac{}{\{P\}x = \mathtt{new}(1)\{\exists x' x'_0.\, P[x'/x] * x \mapsto x'_0\}}$$

$$\frac{}{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

# Proof Rules f

- Assume that x does not appe

$$\overline{\{P\}x{=}E\{x{=}E \wedge P\}} \quad \overline{\{P\}x = \texttt{new}(1)\{\exists}$$

$$\overline{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

- The general rules use x' to denote the old value of x.

$$\overline{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \quad \overline{\{P\}x = \texttt{new}(1)\{\exists x' x'_0.\, P[x'/x] * x \mapsto x'_0\}}$$

$$\overline{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

$\{x{\mapsto}x' * y{\mapsto}y'\}$

$t_1{=}{*}x;$

$\{t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$

$t_2{=}{*}y;$

$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$

${*}x{=}t_2;$

$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}t_2 * y{\mapsto}y'\}$

${*}y{=}t_1$

$\{x{\mapsto}y' * y{\mapsto}x'\}$

# Proof Rules f

- Assume that x does not appe

$$\frac{}{\{P\}x{=}E\{x{=}E \wedge P\}} \qquad \frac{}{\{P\}x = \texttt{new}(1)\{\exists}$$

$$\frac{}{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

$\{x{\mapsto}x' * y{\mapsto}y'\}$
$t_1{=}{*}x;$
$\{t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$
$t_2{=}{*}y;$
$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$
${*}x{=}t_2;$
$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}t_2 * y{\mapsto}y'\}$
$\{t_1{=}x' \wedge x{\mapsto}y' * y{\mapsto}y'\}$
${*}y{=}t_1$

$\{x{\mapsto}y' * y{\mapsto}x'\}$

- The general rules use x' to denote the old value of x.

$$\frac{}{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \qquad \frac{}{\{P\}x = \texttt{new}(1)\{\exists x' x'_0.\, P[x'/x] * x \mapsto x'_0\}}$$

$$\frac{}{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

# Proof Rules f

$\{x{\mapsto}x' * y{\mapsto}y'\}$
$t_1{=}{*}x;$
$\{t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$
$t_2{=}{*}y;$
$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$
${*}x{=}t_2;$
$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}t_2 * y{\mapsto}y'\}$
$\{t_1{=}x' \wedge x{\mapsto}y' * y{\mapsto}y'\}$
${*}y{=}t_1$
$\{t_1{=}x' \wedge x{\mapsto}y' * y{\mapsto}t_1\}$
$\{x{\mapsto}y' * y{\mapsto}x'\}$

- Assume that x does not appe

$$\overline{\{P\}x{=}E\{x{=}E \wedge P\}} \qquad \overline{\{P\}x = \mathtt{new}(1)\{\exists}$$

$$\overline{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

- The general rules use x' to denote the old value of x.

$$\overline{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \qquad \overline{\{P\}x = \mathtt{new}(1)\{\exists x' x_0'.\, P[x'/x] * x \mapsto x_0'\}}$$

$$\overline{\{P * E \mapsto F\}x{=}{*}E\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

# Proof Rules f

- Assume that x does not appe

$$\overline{\{P\}x{=}E\{x{=}E \wedge P\}} \quad \overline{\{P\}x = \mathtt{new}(1)\{\exists}$$

$$\overline{\{P * E{\mapsto}F\}x{=}{*}E\{x{=}F \wedge (P * E{\mapsto}F)\}}$$

$\{x{\mapsto}x' * y{\mapsto}y'\}$
$t_1{=}{*}x;$
$\{t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$
$t_2{=}{*}y;$
$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}x' * y{\mapsto}y'\}$
${*}x{=}t_2;$
$\{t_2{=}y' \wedge t_1{=}x' \wedge x{\mapsto}t_2 * y{\mapsto}y'\}$
$\{t_1{=}x' \wedge x{\mapsto}y' * y{\mapsto}y'\}$
${*}y{=}t_1$
$\{t_1{=}x' \wedge x{\mapsto}y' * y{\mapsto}t_1\}$
$\{x{\mapsto}y' * y{\mapsto}x'\}$

- The general rules use x' to denote the old value of x.

$$\overline{\{P\}x{=}E\{\exists x'.\,x}$$

$$\overline{\{P *}$$

Almost like running the program symbolically. Except that we use Consequence to massage assertions.

- Q: Prove the correctness of swap:

$$\{x{\mapsto}x' * y{\mapsto}y'\}t_1{=}{*}x; t_2{=}{*}y; {*}x{=}t_2; {*}y{=}t_1\{x{\mapsto}y' * y{\mapsto}x'\}$$

# Rules for if and while

$$\frac{\{B \wedge P\}C\{Q\} \quad \{\neg B \wedge P\}C'\{Q\}}{\{P\}\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C'\{Q\}} \qquad \frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

- The rule for if does case-analysis. The rule for while uses invariant-based reasoning.
- E.g.
  - freeList(x) ::= while(x!=0) (t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x,0)\}$

INV : $\mathsf{ls}\,(x,0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\quad t{=}x;$

$\quad x{=}*t;$

$\quad \texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x, 0)\}$
INV : $\mathsf{ls}\,(x, 0)$
$\texttt{while}\ (x \neq 0)\ \{$
$\{x \neq 0 \wedge \mathsf{ls}\,(x, 0)\}$

$t = x;$

$x = *t;$

$\texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while
g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x,0)\}$

INV : $\mathsf{ls}\,(x,0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}$

$\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x',0)\}$

$t{=}x;$

$x{=}*t;$

$\texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x, 0)\}$

INV : $\mathsf{ls}\,(x, 0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\,(x, 0)\}$

$\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$t{=}x;$

$\{\exists x'.\,t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$x{=}*t;$

$\texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x,0)\}$
INV : $\mathsf{ls}\,(x,0)$
 $\texttt{while}\ (x \neq 0)\ \{$
  $\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}$
  $\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x',0)\}$
  $t{=}x;$
  $\{\exists x'.\,t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x',0)\}$

  $x{=}*t;$

  $\texttt{free}(t)$

 $\}$

$\{\mathsf{emp}\}$

d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

Used assignment rule + existential rule.

$$\frac{}{\{P\}x{=}E\{x{=}E \wedge P\}}$$

$$\frac{\{P\}C\{Q\}}{\{\exists x.\,P\}C\{\exists x.\,Q\}}\ x \notin \mathsf{FV}(C)$$

$\{\mathsf{ls}\,(x,0)\}$

$\mathsf{INV}: \mathsf{ls}\,(x,0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}$

$\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x',0)\}$

$t{=}x;$

$\{\exists x'.\,t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x',0)\}$

$\{\exists x'.\,(t \mapsto x') * \mathsf{ls}\,(x',0)\}$

$x{=}*t;$

$\texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x, 0)\}$

INV : $\mathsf{ls}\,(x, 0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\,(x, 0)\}$

$\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$t{=}x;$

$\{\exists x'.\,t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$\{\exists x'.\,(t \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$x{=}*t;$

$\{\exists x'.\,x{=}x' \wedge (t \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$\texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x, 0)\}$

$\mathsf{INV} : \mathsf{ls}\,(x, 0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\,(x, 0)\}$

$\{\exists x'.\, (x \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$t{=}x;$

$\{\exists x'.\, t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$\{\exists x'.\, (t \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$x{=}*t;$

$\{\exists x'.\, x{=}x' \wedge (t \mapsto x') * \mathsf{ls}\,(x', 0)\}$

$\{(t \mapsto x) * \mathsf{ls}\,(x, 0)\}$

$\texttt{free}(t)$

$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x,0)\}$
INV : $\mathsf{ls}\,(x,0)$
$\mathtt{while}\ (x \neq 0)\ \{$
$\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}$
$\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x',0)\}$
$t{=}x;$
$\{\exists x'.\,t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x',0)\}$
$\{\exists x'.\,(t \mapsto x') * \mathsf{ls}\,(x',0)\}$
$x{=}*t;$
$\{\exists x'.\,x{=}x' \wedge (t \mapsto x') * \mathsf{ls}\,(x',0)\}$
$\{(t \mapsto x) * \mathsf{ls}\,(x,0)\}$
$\mathtt{free}(t)$
$\{\mathsf{ls}\,(x,0)\}$
$\}$

$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\mathtt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

$\{\mathsf{ls}\,(x,0)\}$
INV : $\mathsf{ls}\,(x,0)$
$\texttt{while}\ (x \neq 0)\ \{$
$\{x{\neq}0 \wedge \mathsf{ls}\,(x,0)\}$
$\{\exists x'.\,(x \mapsto x') * \mathsf{ls}\,(x',0)\}$
$t{=}x;$
$\{\exists x'.\,t{=}x \wedge (x \mapsto x') * \mathsf{ls}\,(x',0)\}$
$\{\exists x'.\,(t \mapsto x') * \mathsf{ls}\,(x',0)\}$
$x{=}*t;$
$\{\exists x'.\,x{=}x' \wedge (t \mapsto x') * \mathsf{ls}\,(x',0)\}$
$\{(t \mapsto x) * \mathsf{ls}\,(x,0)\}$
$\texttt{free}(t)$
$\{\mathsf{ls}\,(x,0)\}$
$\}$
$\{x{=}0 \wedge \mathsf{ls}\,(x,0)\}$
$\{\mathsf{emp}\}$

# d while

$$\frac{\{B \wedge P\}C\{P\}}{\{P\}\texttt{while}\ B\ C\{\neg B \wedge P\}}$$

ysis. The rule for while

g.

(t=x; x=*t; free(t))

# Summary

- Proof rules are like symbolic execution.
- Structural rules help us to adjust symbolic states.

# Prove the list reversal example.

- Use the below rules you learned.

$$\frac{P \Rightarrow P' \quad \{P'\}C\{Q'\} \quad Q' \Rightarrow Q}{\{P\}C\{Q\}} \qquad \frac{\{P\}C\{Q\}}{\{\exists x.\, P\}C\{\exists x.\, Q\}}\ x \notin \mathsf{FV}(C)$$

$$\frac{\{P\}C\{Q\} \quad \{P'\}C\{Q'\}}{\{P \vee P'\}C\{Q \vee Q'\}} \qquad \frac{\{P\}C\{P'\} \quad \{P'\}C'\{Q\}}{\{P\}C; C'\{Q\}}$$

$$\frac{}{\{P * E \mapsto E_0\}{*E}{=}F\{P * E \mapsto F\}} \qquad \frac{}{\{P * E \mapsto E_0\}\mathtt{free}(E)\{P\}}$$

$$\frac{}{\{P\}x{=}E\{\exists x'.\, x{=}E[x'/x] \wedge P[x'/x]\}} \qquad \frac{}{\{P\}x = \mathtt{new}(1)\{\exists x' x'_0.\, P[x'/x] * x \mapsto x'_0\}}$$

$$\frac{}{\{P * E \mapsto F\}x{=}{*E}\{\exists x'.\, x{=}F[x'/x] \wedge (P * E{\mapsto}F)[x'/x]\}}$$

$$\frac{\{B \wedge P\}C\{Q\} \quad \{\neg B \wedge P\}C'\{Q\}}{\{P\}\mathtt{if}\ B\ \mathtt{then}\ C\ \mathtt{else}\ C'\{Q\}} \qquad \frac{\{B \wedge P\}C\{P\}}{\{P\}\mathtt{while}\ B\ C\{\neg B \wedge P\}}$$

# Prove … e.

- Use the bel…

$\{\mathsf{ls}\ (x, 0)\}$

$y = 0;$

$\mathsf{INV}: \ \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\quad t = x;$

$\quad x = *t;$

$\quad *t = y;$

$\quad y = t$

$\}$

$\{\mathsf{ls}\ (y, 0)\}$

$$\frac{P \Rightarrow P' \quad \cdots}{\cdots}$$

$$\frac{\{P\ \cdots}{\{\cdots}$$

$$\frac{}{\{P * E \mapsto \cdots P\}}$$

$$\frac{}{\{P\}x{=}E\{\exists x'.\, x{=}\cdots x \mapsto x_0'\}}$$

$$\frac{}{\{P * \cdots}$$

$$\frac{\{B \wedge P\}\ \cdots}{\{P\}\texttt{if}\ \cdots}$$

# Prove ... e.

- Use the bel...

$\{\mathsf{ls}\ (x,0)\}$

$y = 0;$

$\mathsf{INV}:\ \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\quad \{x{\neq}0 \wedge \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)\}$

$\quad t = x;$

$\quad x = *t;$

$\quad *t = y;$

$\quad y = t$

$\}$

$\{\mathsf{ls}\ (y,0)\}$

$P \Rightarrow P'$

$\{P$

$\{$

$\{P * E \mapsto$ ... $P\}$

$\{P\}x{=}E\{\exists x'.\, x{=}$ ... $x \mapsto x_0'\}$

$\{P *$

$\{B \wedge P\}$

$\{P\}\texttt{if}$

# Prove e.

- Use the bel

$\{\mathsf{ls}\ (x, 0)\}$

$y = 0;$

$\mathsf{INV}: \ \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\quad \{x{\neq}0 \wedge \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad t = x;$

$\quad x = *t;$

$\quad *t = y;$

$\quad y = t$

$\}$

$\{\mathsf{ls}\ (y, 0)\}$

$\dfrac{P \Rightarrow P'}{}$

$\dfrac{\{P}{\{}$

$\overline{\{P * E \mapsto \quad P\}}$

$\overline{\{P\}x{=}E\{\exists x'.\, x{=} \quad x \mapsto x'_0\}}$

$\overline{\{P *}$

$\dfrac{\{B \wedge P\}}{\{P\}\texttt{if}}$

# Prove ... e.

- Use the bel

$P \Rightarrow P'$

$\{P$

$\{P * E \mapsto$ ... $P\}$

$\{P\}x{=}E\{\exists x'.\, x{=}$ ... $x \mapsto x_0'\}$

$\{P *$

$\{B \wedge P\}$

$\{P\}$`if`

$\{\mathsf{ls}\ (x, 0)\}$

$y = 0;$

$\mathsf{INV}:\ \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\{\exists x'.\, \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$t = x;$

$\{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$x = *t;$

$*t = y;$

$y = t$

$\}$

$\{\mathsf{ls}\ (y, 0)\}$

$\{\mathsf{ls}\ (x, 0)\}$

$y = 0;$

$\mathsf{INV}: \ \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)$

$\mathtt{while}\ (x \neq 0)\ \{$

$\quad \{x{\neq}0 \wedge \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad t = x;$

$\quad \{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y, 0) * (t \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad x = *t;$

$\quad *t = y;$

$\quad y = t$

$\}$

$\{\mathsf{ls}\ (y, 0)\}$

# Prove ... e.

- Use the bel...

$P \Rightarrow P'$

$\{P$

$\{$

$\{P * E \mapsto$ ... $P\}$

$\{P\}x{=}E\{\exists x'.\, x{=}$ ... $x \mapsto x'_0\}$

$\{P *$

$\{B \wedge P\}$

$\{P\}$`if`

$\{\mathsf{ls}\ (x, 0)\}$

$y = 0;$

$\mathsf{INV}:\ \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)$

`while` $(x \neq 0)\ \{$

$\{x{\neq}0 \wedge \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\{\exists x'.\, \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$t = x;$

$\{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\{\exists x'.\, \mathsf{ls}\ (y, 0) * (t \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$x = *t;$

$\{\exists x'.\, x{=}x' \wedge \mathsf{ls}\ (y, 0) * (t \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$*t = y;$

$y = t$

$\}$

$\{\mathsf{ls}\ (y, 0)\}$

$\{\mathsf{ls}\ (x,0)\}$

$y = 0;$

$\mathsf{INV}:\ \ \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\quad \{x{\neq}0 \land \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad t = x;$

$\quad \{\exists x'.\, t{=}x \land \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad x = *t;$

$\quad \{\exists x'.\, x{=}x' \land \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad \{\mathsf{ls}\ (y,0) * (t \mapsto x) * \mathsf{ls}\ (x,0)\}$

$\quad *t = y;$

$\quad y = t$

$\}$

$\{\mathsf{ls}\ (y,0)\}$

# Prove … e.

- Use the bel…

$P \Rightarrow P'$

$\{P$

$\{$

$\{P * E \mapsto$ … $P\}$

$\{P\}x{=}E\{\exists x'.\, x{=}$ … $x \mapsto x_0'\}$

$\{P *$

$\{B \land P\}$

$\{P\}\texttt{if}$

# Prove … e.

- Use the bel…

$\{\mathsf{ls}\ (x,0)\}$

$y = 0;$

$\mathsf{INV}:\ \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)$

`while` $(x \neq 0)\ \{$

$\quad\{x{\neq}0 \wedge \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)\}$

$\quad\{\exists x'.\, \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad t = x;$

$\quad\{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad\{\exists x'.\, \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad x = *t;$

$\quad\{\exists x'.\, x{=}x' \wedge \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad\{\mathsf{ls}\ (y,0) * (t \mapsto x) * \mathsf{ls}\ (x,0)\}$

$\quad *t = y;$

$\quad\{\mathsf{ls}\ (y,0) * (t \mapsto y) * \mathsf{ls}\ (x,0)\}$

$\quad y = t$

$\}$

$\{\mathsf{ls}\ (y,0)\}$

# Prove … e.

- Use the bel

$\dfrac{P \Rightarrow P'}{}$

$\{P$

$\{$

$\{P * E \mapsto$ … $P\}$

$\{P\}x{=}E\{\exists x'.\, x{=}$ … $x \mapsto x_0'\}$

$\{P *$

$\{B \wedge P\}$

$\{P\}$if

$\{\mathsf{ls}\ (x,0)\}$

$y = 0;$

$\mathsf{INV}:\ \ \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)$

$\texttt{while}\ (x \neq 0)\ \{$

$\quad \{x{\neq}0 \wedge \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad t = x;$

$\quad \{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad x = *t;$

$\quad \{\exists x'.\, x{=}x' \wedge \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$

$\quad \{\mathsf{ls}\ (y,0) * (t \mapsto x) * \mathsf{ls}\ (x,0)\}$

$\quad *t = y;$

$\quad \{\mathsf{ls}\ (y,0) * (t \mapsto y) * \mathsf{ls}\ (x,0)\}$

$\quad y = t$

$\quad \{\exists y'.\, y{=}t \wedge \mathsf{ls}\ (y',0) * (t \mapsto y') * \mathsf{ls}\ (x,0)\}$

$\}$

$\{\mathsf{ls}\ (y,0)\}$

# Prove … e.

- Use the bel…

$$\{\mathsf{ls}\ (x,0)\}$$

$$\color{red}{y = 0;}$$

$$\mathsf{INV}:\ \ \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)$$

$$\color{red}{\texttt{while}\ (x \neq 0)\ \{}$$

$$\{x{\neq}0 \wedge \mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)\}$$

$$\{\exists x'.\, \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$$

$$\color{red}{t = x;}$$

$$\{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y,0) * (x \mapsto x') * \mathsf{ls}\ (x',0)\}$$

$$\{\exists x'.\, \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$$

$$\color{red}{x = *t;}$$

$$\{\exists x'.\, x{=}x' \wedge \mathsf{ls}\ (y,0) * (t \mapsto x') * \mathsf{ls}\ (x',0)\}$$

$$\{\mathsf{ls}\ (y,0) * (t \mapsto x) * \mathsf{ls}\ (x,0)\}$$

$$\color{red}{*t = y;}$$

$$\{\mathsf{ls}\ (y,0) * (t \mapsto y) * \mathsf{ls}\ (x,0)\}$$

$$\color{red}{y = t}$$

$$\{\exists y'.\, y{=}t \wedge \mathsf{ls}\ (y',0) * (t \mapsto y') * \mathsf{ls}\ (x,0)\}$$

$$\{\mathsf{ls}\ (y,0) * \mathsf{ls}\ (x,0)\}$$

$$\color{red}{\}}$$

$$\{\mathsf{ls}\ (y,0)\}$$

$\{\mathsf{ls}\ (x, 0)\}$

$y = 0;$

$\mathsf{INV}: \ \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)$

$\mathtt{while}\ (x \neq 0)\ \{$

$\quad \{x{\neq}0 \wedge \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad t = x;$

$\quad \{\exists x'.\, t{=}x \wedge \mathsf{ls}\ (y, 0) * (x \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad \{\exists x'.\, \mathsf{ls}\ (y, 0) * (t \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad x = *t;$

$\quad \{\exists x'.\, x{=}x' \wedge \mathsf{ls}\ (y, 0) * (t \mapsto x') * \mathsf{ls}\ (x', 0)\}$

$\quad \{\mathsf{ls}\ (y, 0) * (t \mapsto x) * \mathsf{ls}\ (x, 0)\}$

$\quad *t = y;$

$\quad \{\mathsf{ls}\ (y, 0) * (t \mapsto y) * \mathsf{ls}\ (x, 0)\}$

$\quad y = t$

$\quad \{\exists y'.\, y{=}t \wedge \mathsf{ls}\ (y', 0) * (t \mapsto y') * \mathsf{ls}\ (x, 0)\}$

$\quad \{\mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\}$

$\{x{=}0 \wedge \mathsf{ls}\ (y, 0) * \mathsf{ls}\ (x, 0)\}$

$\{\mathsf{ls}\ (y, 0)\}$

# Prove ... e.

- Use the bel...

$P \Rightarrow P'$

$\{P$

$\{$

$\{P * E \mapsto$ ... $P\}$

$\{P\}x{=}E\{\exists x'.\, x{=}$ ... $x \mapsto x'_0\}$

$\{P *$

$\{B \wedge P\}$

$\{P\}\mathtt{if}$

# HW1

- Verify the full correctness of list reversal.

# HW2

- Find proof rules for weakest precondition.

  {??}C{Q}

- The rule for free is already of that form.
- Use separating implication for x=new(1) and *E=E'.
- What about strongest postconditions?

# Reference

- Reynolds's LICS 2001 paper.