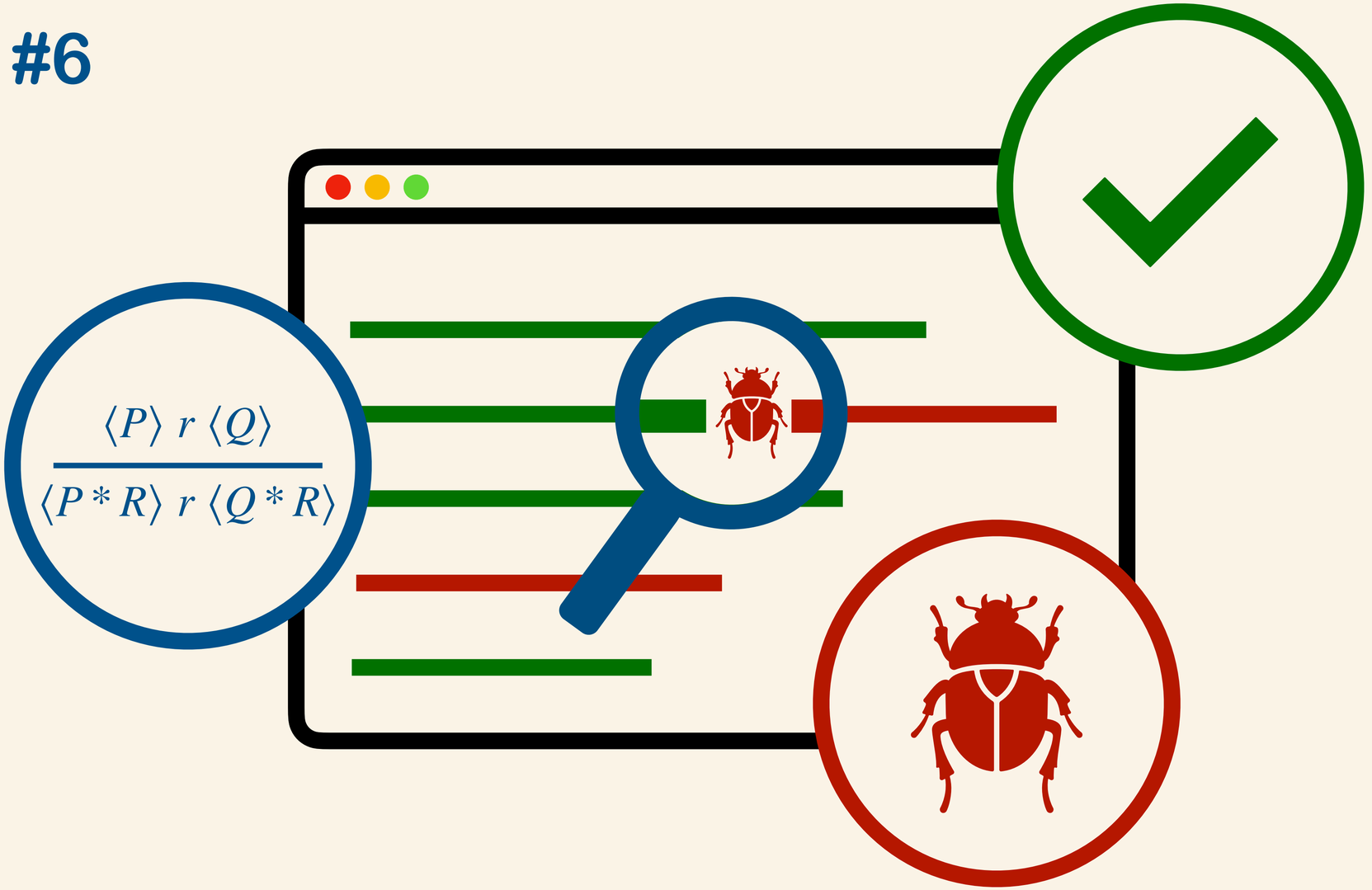
SCAN ME

Program Analysis

Lecture #6

Roberto Bruni

Università di Pisa



PhD Course
June 30 - July 4, 2025

GSSI Gran Sasso Science Institute
School of Advanced Studies
Scuola Universitaria Superiore

Pointer analysis

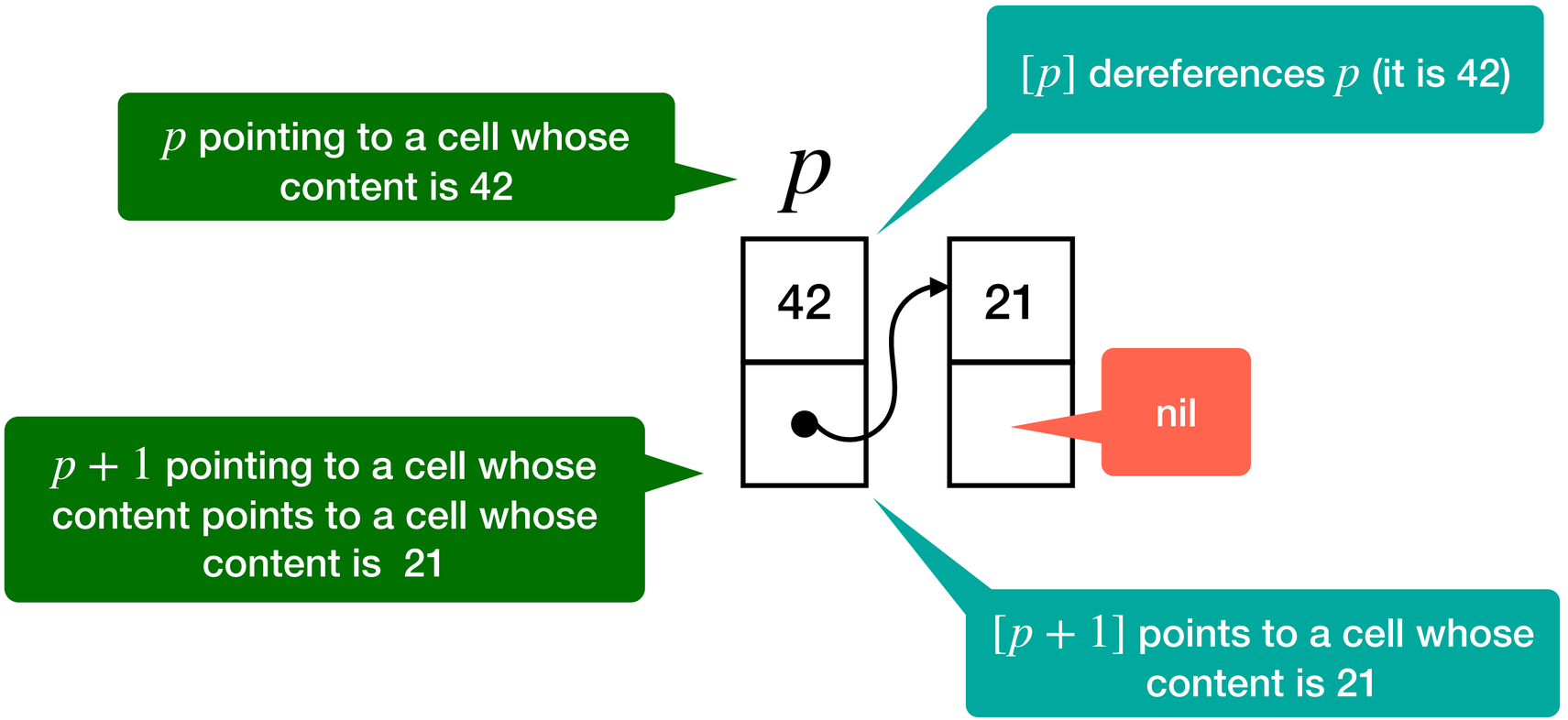
Why is pointer analysis important?

“If the pointer is not pointing to a valid object or function, bad things may happen”

Robert C. Seacord

Effective C: An Introduction to Professional C Programming

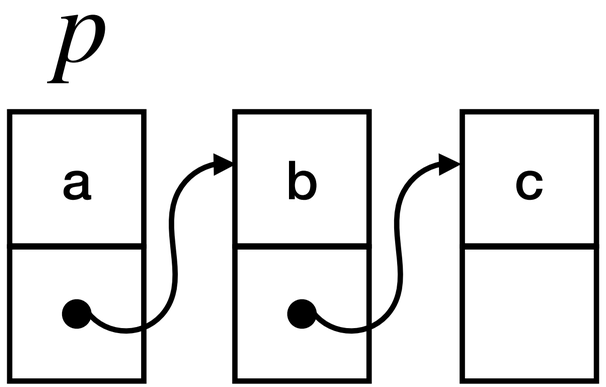
Pointer notation



e ::= skip
| $b?$
| $x := a$
| $x := [a]$ // read
| $[a_1] := a_2$ // write
| $x :=$ alloc( )
| free(x)

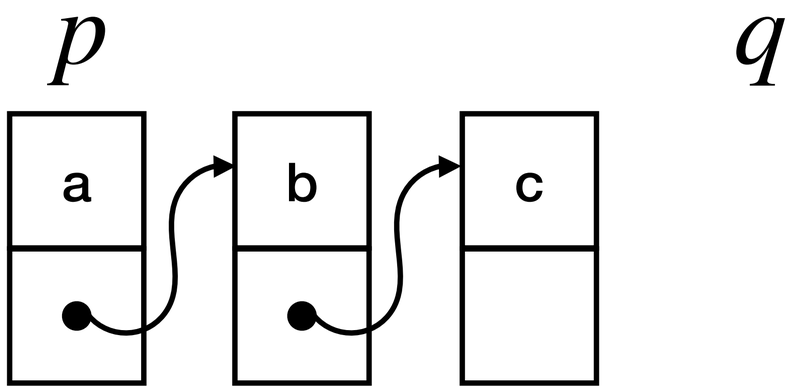
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



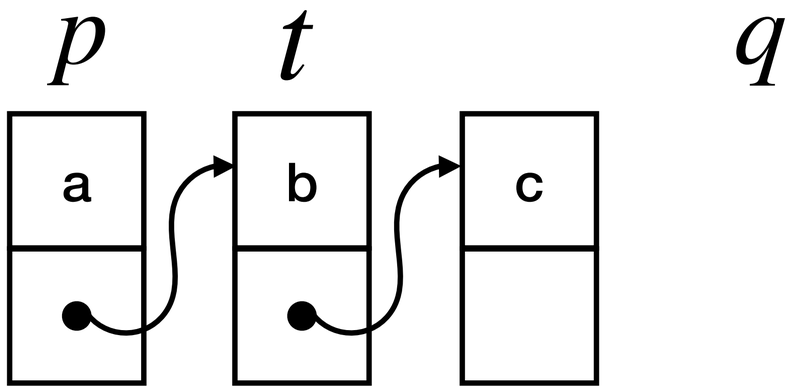
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



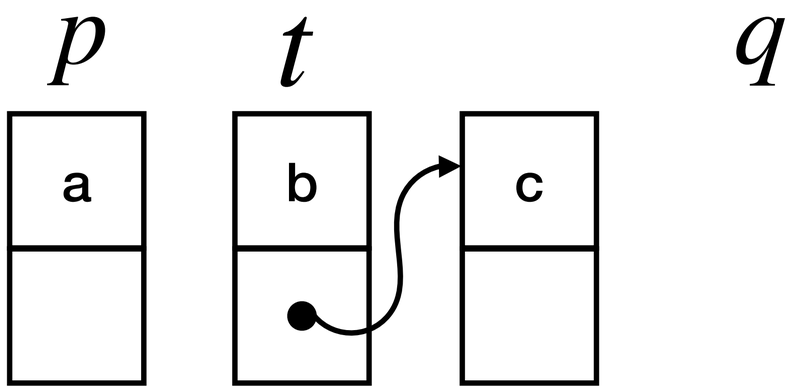
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



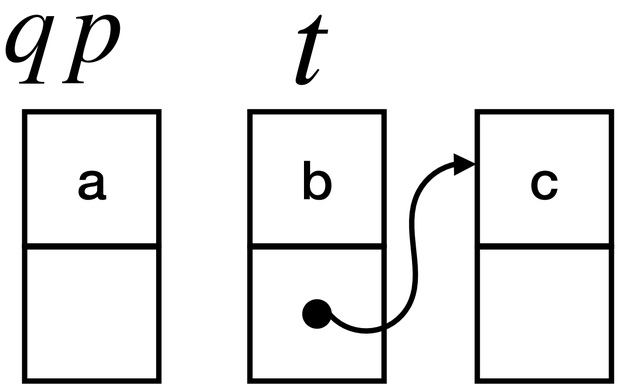
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



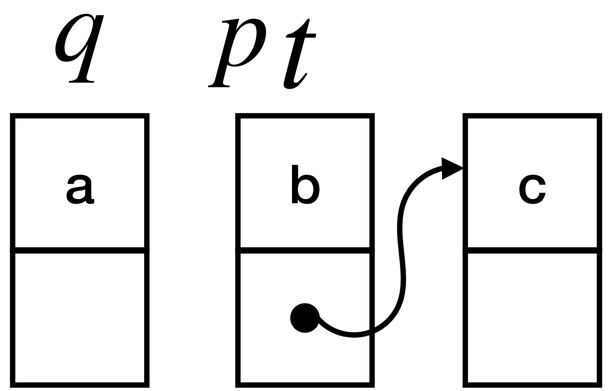
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



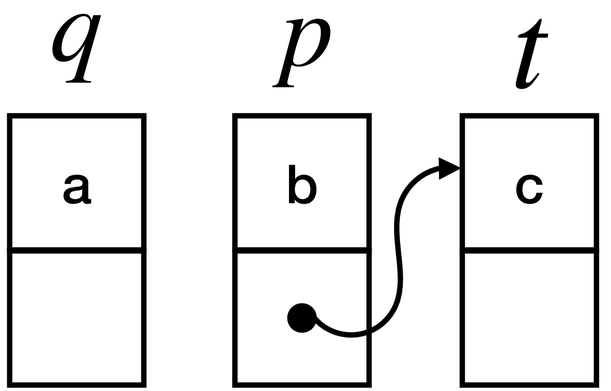
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



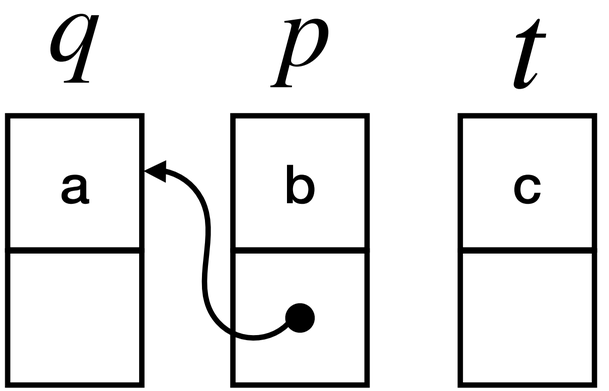
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



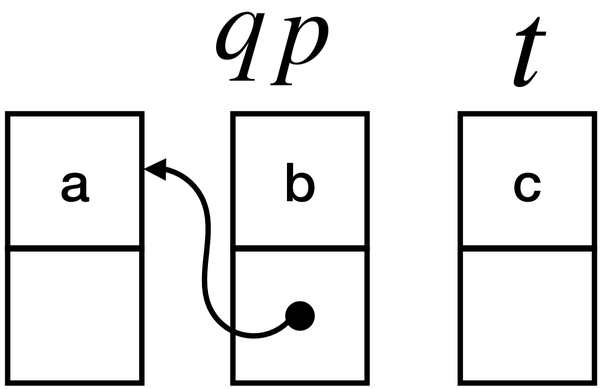
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



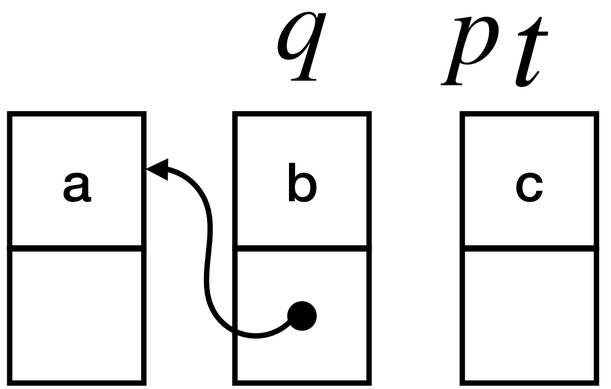
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



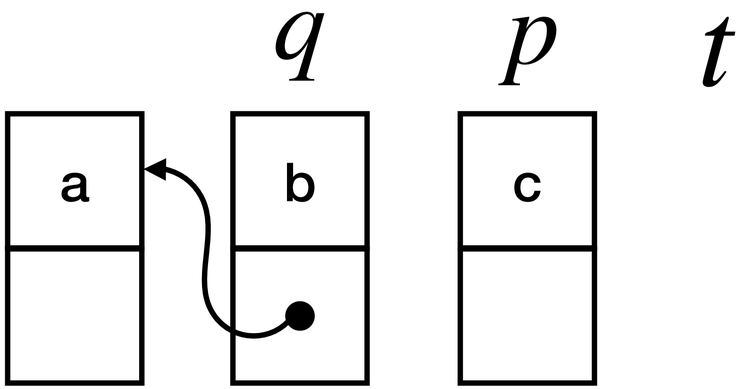
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



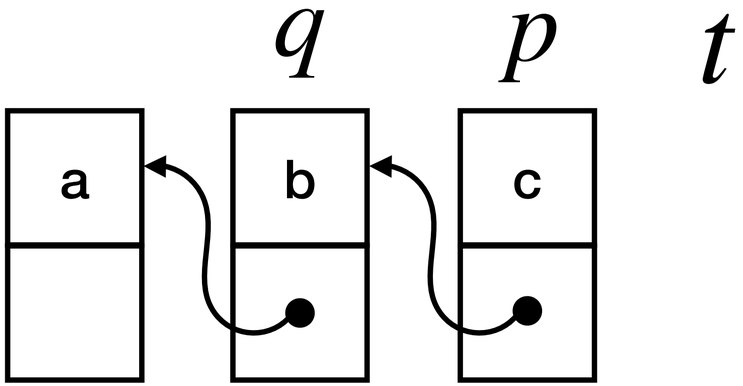
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



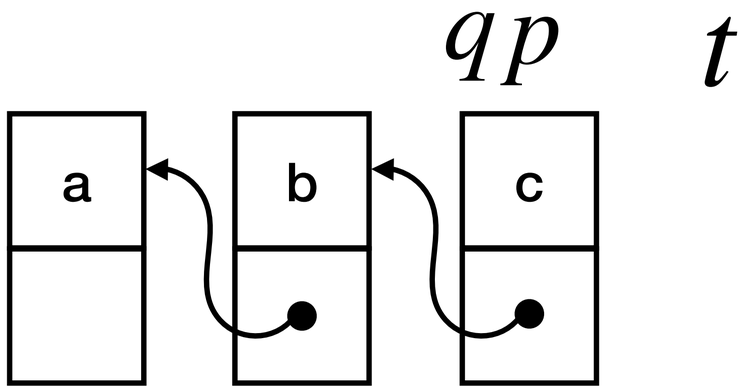
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



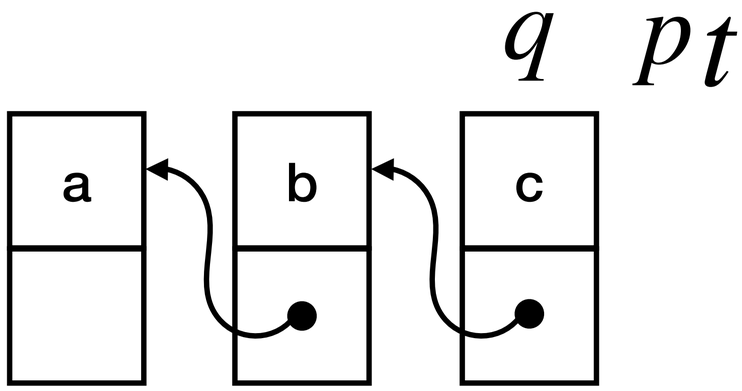
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



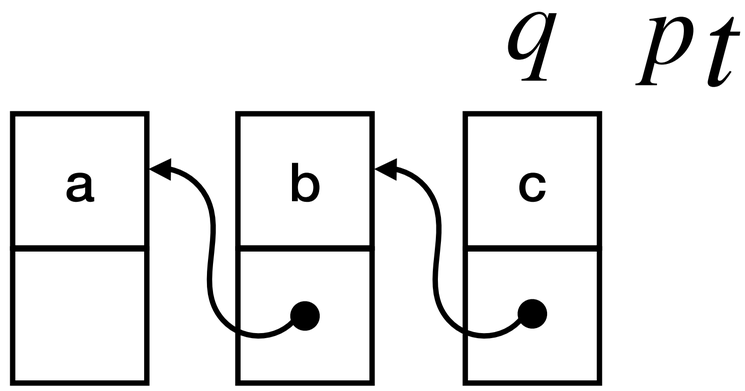
Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



Example

q := nil;
while $p \neq$ nil do (
    $t := [p + 1]$;
    $[p + 1] := q$;
    $q := p$;
    $p := t$
)



Example

$q := \text{nil};$
while $p \neq \text{nil}$ do (
    $t := [p + 1];$
    $[p + 1] := q;$
    $q := p;$
    $p := t$
)

invariant?

intuitively

q points to the list fragment β already reversed

p points to the list fragment α still to be reversed

the reverse $\alpha_0^\dagger$ of the original list α_0

is given by composing the reverse of α followed by β

Example

$q := \text{nil};$
while $p \neq \text{nil}$ do (
  $t := [p + 1];$
  $[p + 1] := q;$
  $q := p;$
  $p := t$
)

invariant?

reversal

$$P \triangleq \exists \alpha, \beta \,.\, \text{list}(\alpha, p) \wedge \text{list}(\beta, q) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$$

an inductive list predicate:

points to

$$\text{list}(\epsilon, p) \triangleq (p = \text{nil})$$

$$\text{list}(n \cdot \alpha, p) \triangleq \exists q \,.\, p \mapsto \langle n, q \rangle \wedge \text{list}(\alpha, q)$$

Example

$q := \text{nil};$
while $p \neq \text{nil}$ do (
  $t := [p + 1];$
  $[p + 1] := q;$
  $q := p;$
  $p := t$
)

invariant?

reversal

$$P \triangleq \exists \alpha, \beta \,.\, \text{list}(\alpha, p) \wedge \text{list}(\beta, q) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$$

would fail if lists overlap!

Example

$q := \text{nil};$
while $p \neq \text{nil}$ do (
  $t := [p + 1];$
  $[p + 1] := q;$
  $q := p;$
  $p := t$
)

invariant?

$$P \triangleq \exists \alpha, \beta \,.\, \text{list}(\alpha, p) \wedge \text{list}(\beta, q) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$$
$$\wedge\, (\forall k \,.\, \text{reach}(p, k) \wedge \text{reach}(q, k) \Rightarrow k = \text{nil})$$

do not overlap

an inductive reachability predicate:

$$\text{reach}(p, q) \triangleq p = q$$
$$\vee\ \exists n, t \,.\, p \mapsto \langle n, t \rangle \wedge \text{reach}(t, q)$$

Example

$q := \text{nil};$
$\text{while } p \neq \text{nil do (}$
$\quad t := [p+1];$
$\quad [p+1] := q;$
$\quad q := p;$
$\quad p := t$
)

invariant?

$$P \triangleq \exists \alpha, \beta\,.\ \text{list}(\alpha, p) \wedge \text{list}(\beta, q) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$$
$$\wedge\ (\forall k\,.\ \text{reach}(p,k) \wedge \text{reach}(q,k) \Rightarrow k = \text{nil})$$

what if other lists are used?

Example

$q := \text{nil};$
$\text{while } p \neq \text{nil do (}$
$\quad t := [p+1];$
$\quad [p+1] := q;$
$\quad q := p;$
$\quad p := t$
)

invariant?

$$P \triangleq \exists \alpha, \beta\,.\ \text{list}(\alpha, p) \wedge \text{list}(\beta, q) \wedge \text{list}(\gamma, x) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$$
$$\wedge\ (\forall k\,.\ \text{reach}(p,k) \wedge \text{reach}(q,k) \Rightarrow k = \text{nil})$$
$$\wedge\ (\forall k\,.\ \text{reach}(x,k) \wedge (\text{reach}(p,k) \vee \text{reach}(q,k)) \Rightarrow k = \text{nil})$$

Example

$q := \text{nil};$
while $p \neq \text{nil}$ do (
  $t := [p + 1];$
  $[p + 1] := q;$
  $q := p;$
  $p := t$
)

$P \triangleq \exists \alpha, \beta \,.\, (\text{list}(\alpha, p) * \text{list}(\beta, q)) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$

separating conjunction!

the two lists speak about different parts of the heap

Example

$\{\mathsf{true}\}$

$[x] := 1;$

$[y] := 2;$

$[z] := 3;$

$\{z \mapsto 3\}$

is it valid?

Example

$\{\mathsf{true}\}$

$[x] := 1;$

$[y] := 2;$

$[z] := 3;$

$\{z \mapsto 3\}$

valid!

Example

$\{\mathsf{true}\}$

$[x] := 1;$

$[y] := 2;$

$[z] := 3;$

$\{x \mapsto 1 \land y \mapsto 2 \land z \mapsto 3\}$

is it valid?

Example

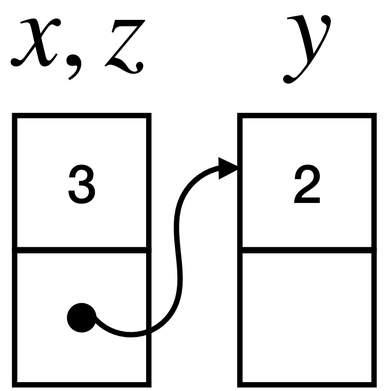
$\{\text{true}\}$

$[x] := 1;$

$[y] := 2;$

$[z] := 3;$

$\{x \mapsto 1 \land y \mapsto 2 \land z \mapsto 3\}$



we must exclude aliasing!

is it valid?

Example

$\{x \neq y \wedge x \neq z \wedge y \neq z\}$

$[x] := 1;$

$[y] := 2;$

$[z] := 3;$

$\{x \mapsto 1 \wedge y \mapsto 2 \wedge z \mapsto 3\}$

valid!

Example

$\{x_1 \neq x_2 \wedge \ldots\}$

$[x_1] := 1;$

$[x_2] := 2;$

...

$[x_n] := n;$

$\{x_1 \mapsto 1 \wedge \ldots \wedge x_n \mapsto n\}$

$n(n-1)/2$ conjuncts!

Example

ownership of heap cell at x

separating conjunction!

$$\{x_1 \mapsto \_ * \ldots * x_n \mapsto \_\}$$

n conjuncts!

$$[x_1] := 1;$$
$$[x_2] := 2;$$
$$\ldots$$
$$[x_n] := n;$$

$$\{x_1 \mapsto 1 * \ldots * x_n \mapsto n\}$$