Program analysis: from proving correctness to proving incorrectness

Roberto Bruni, Roberta Gori
(University of Pisa)
Lecture #09

BISS 2024
March 11-15, 2024

Pointer analysis

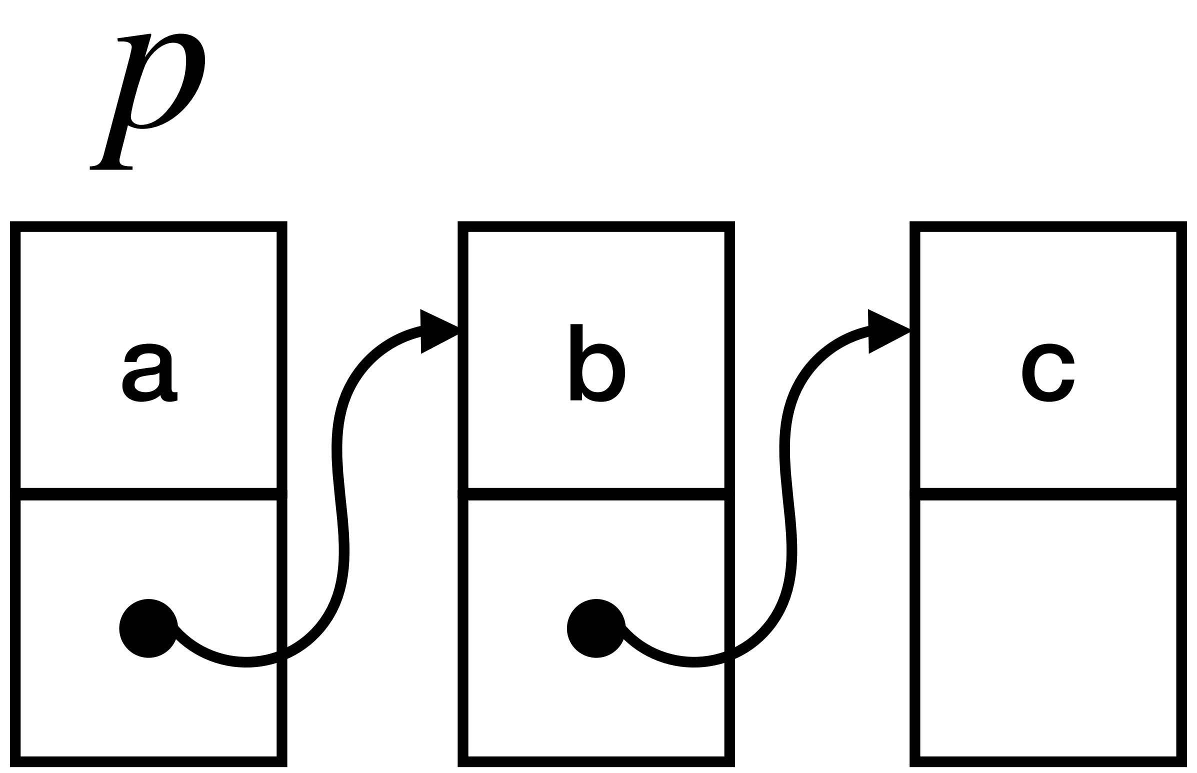
Why is pointer analysis important?

“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

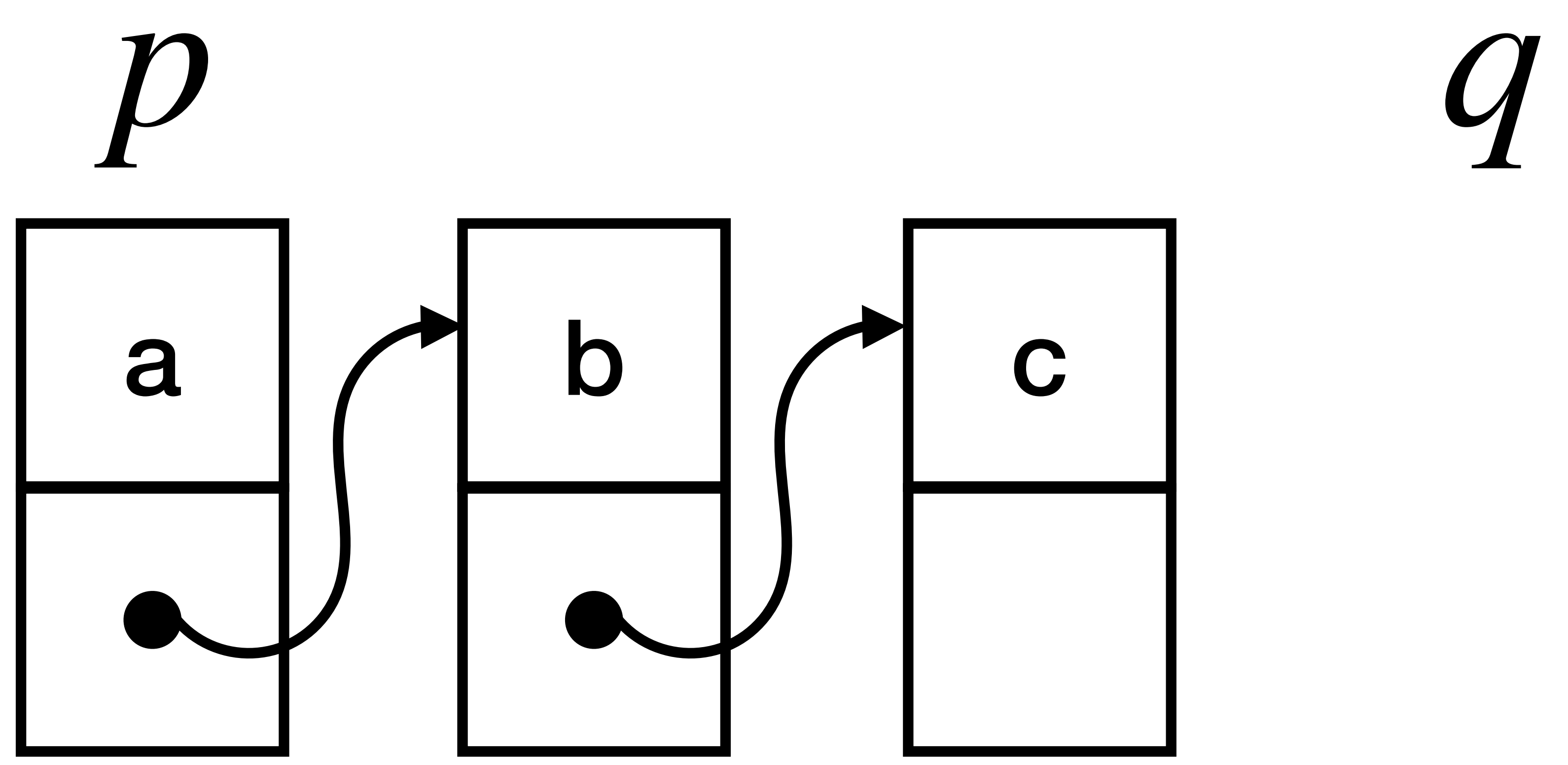
Example

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



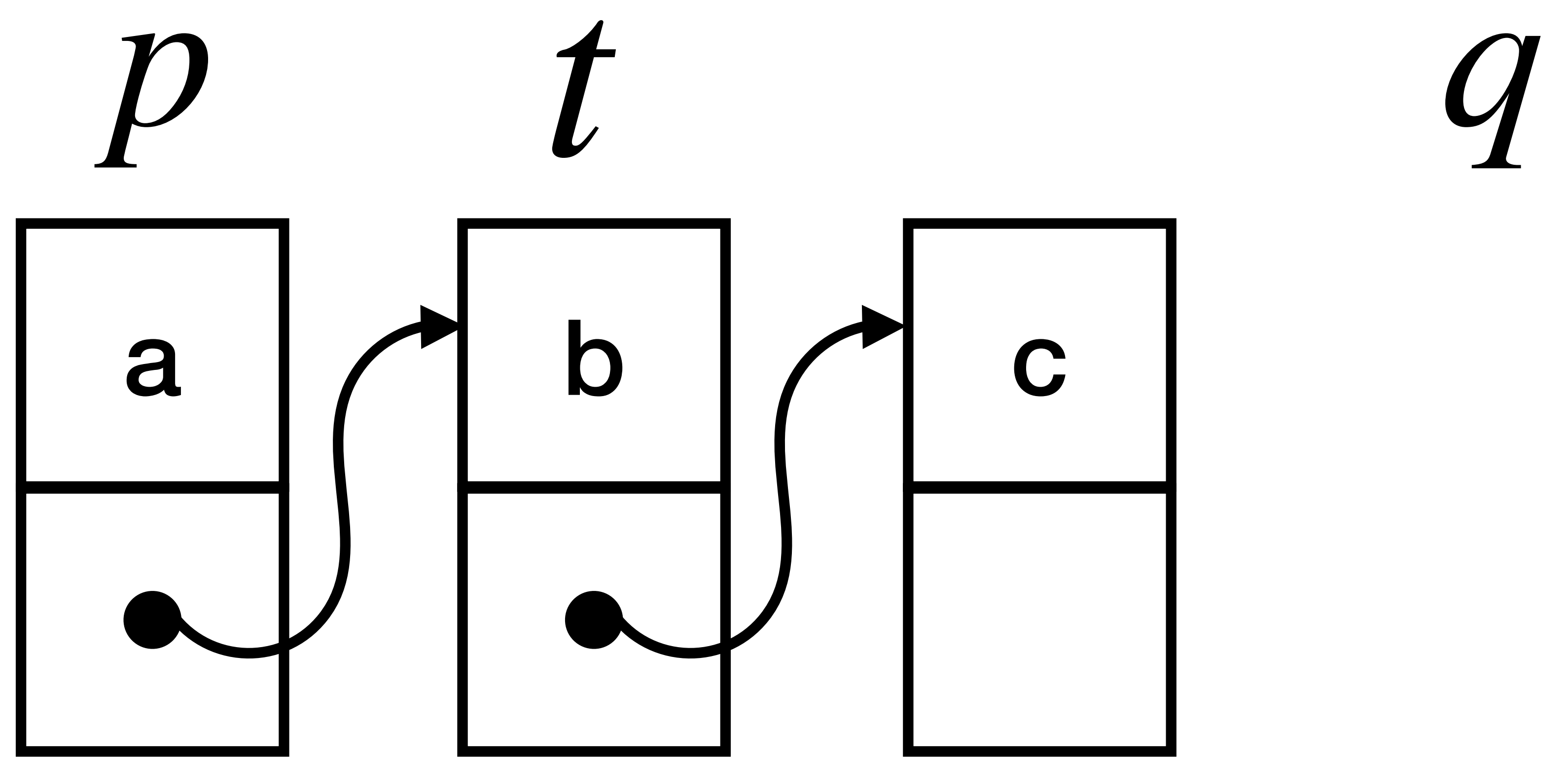
Example

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



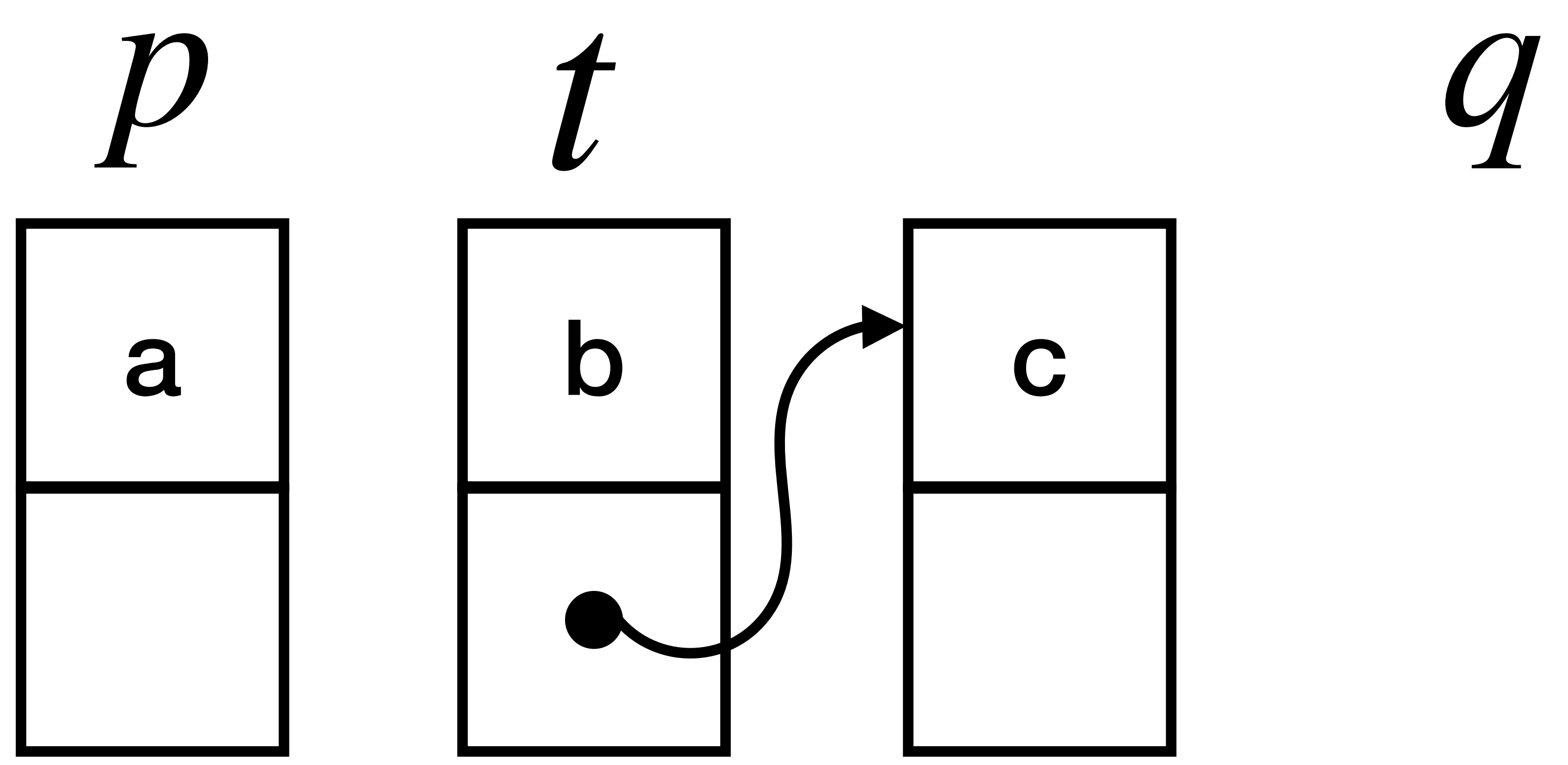
Example

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



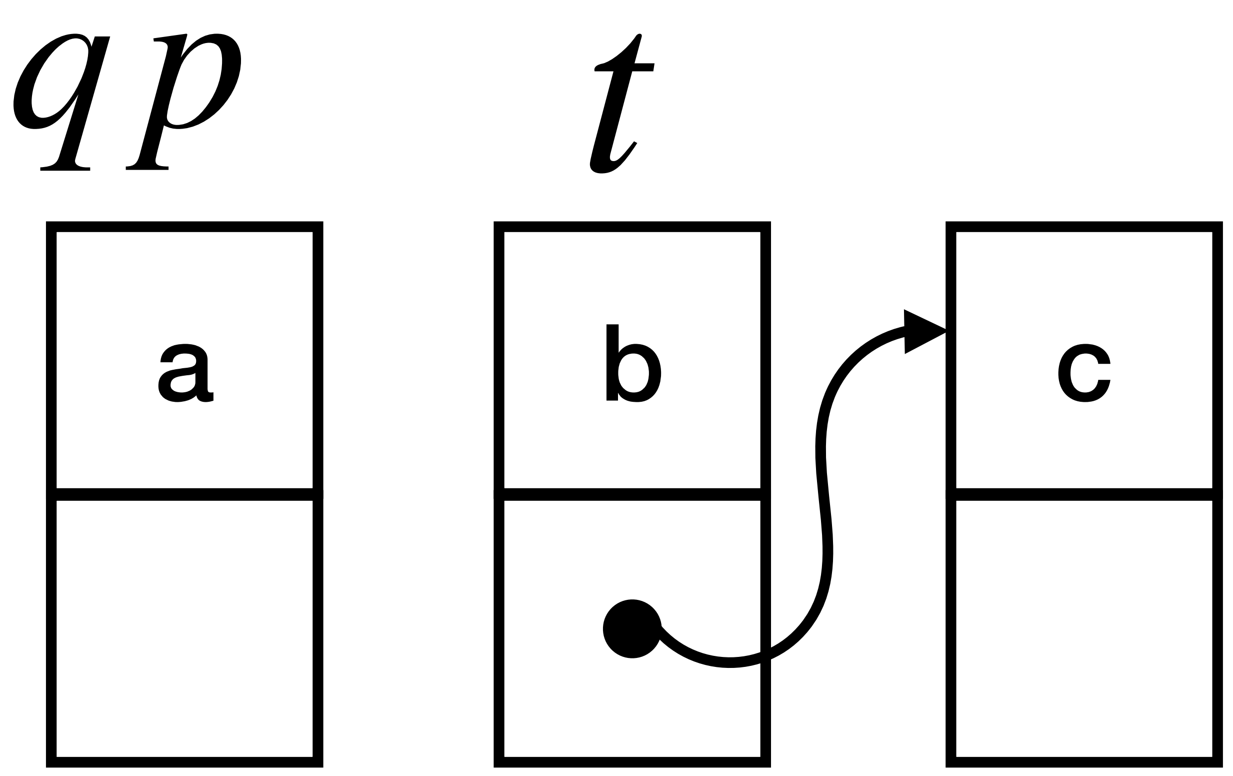
Example

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



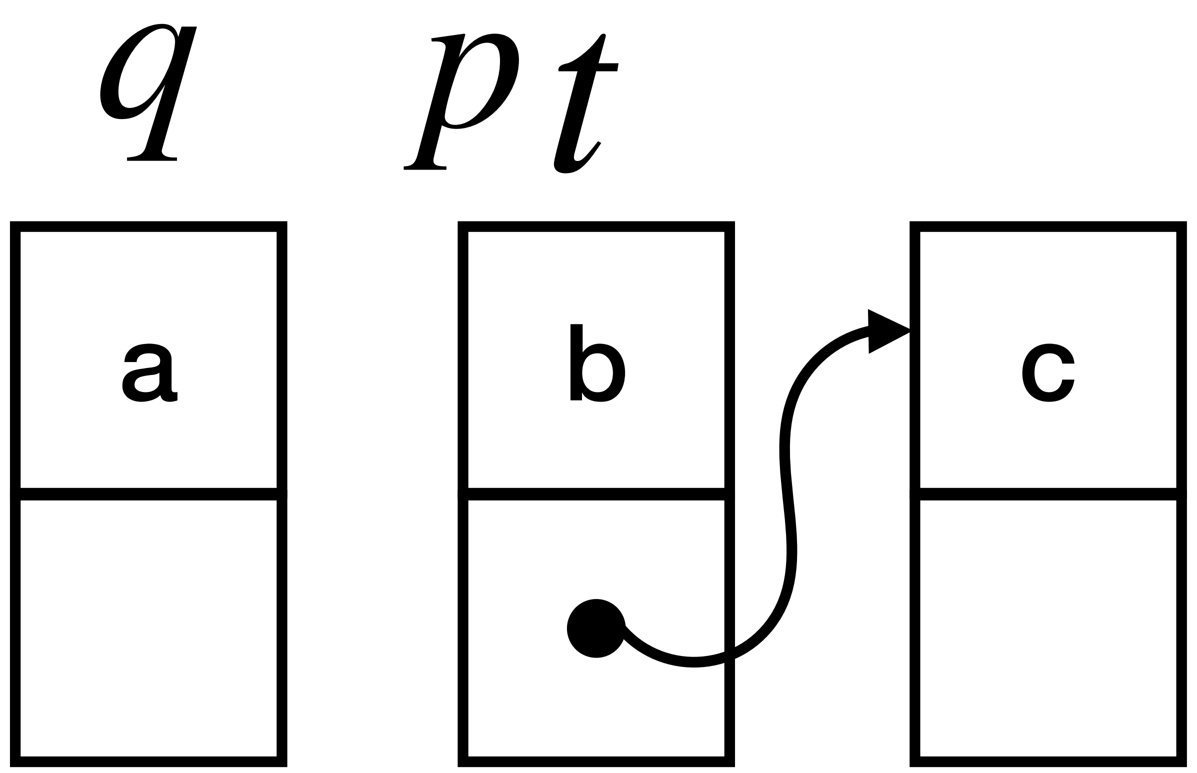
Example

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



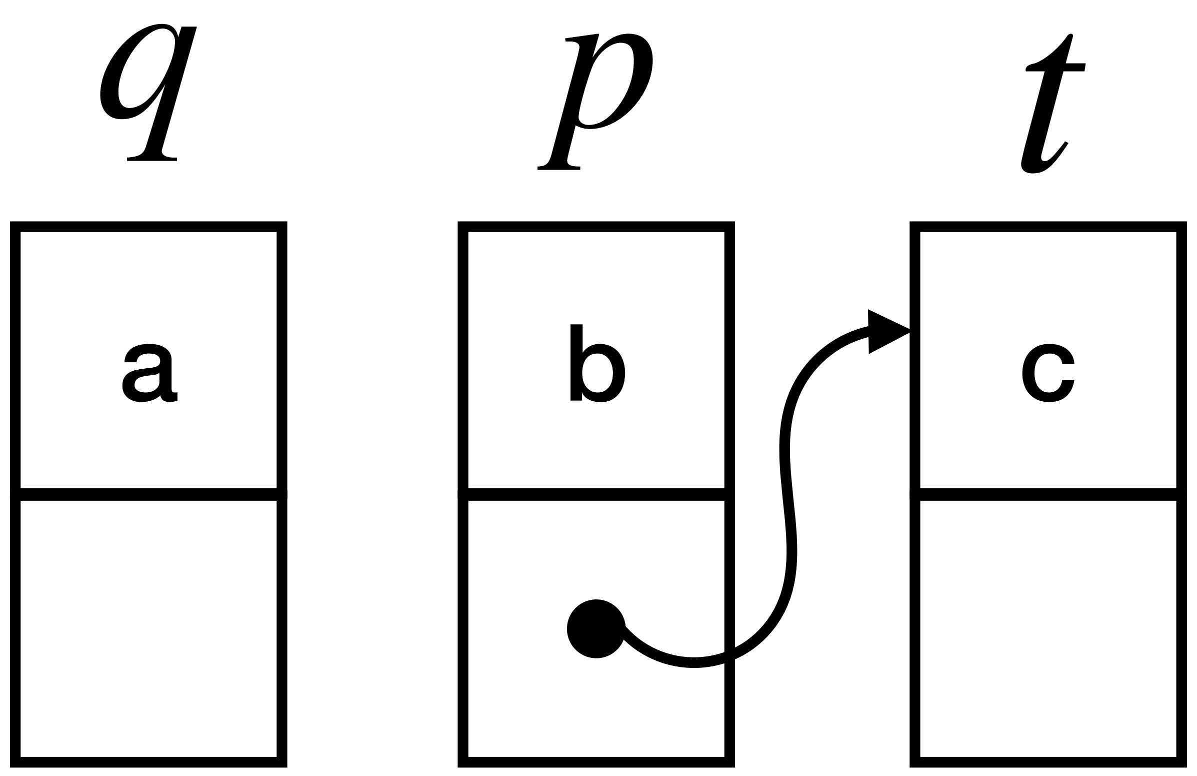
Example

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



Example

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



Example

$q := \text{nil};$

$\text{while } p \neq \text{nil do } ($

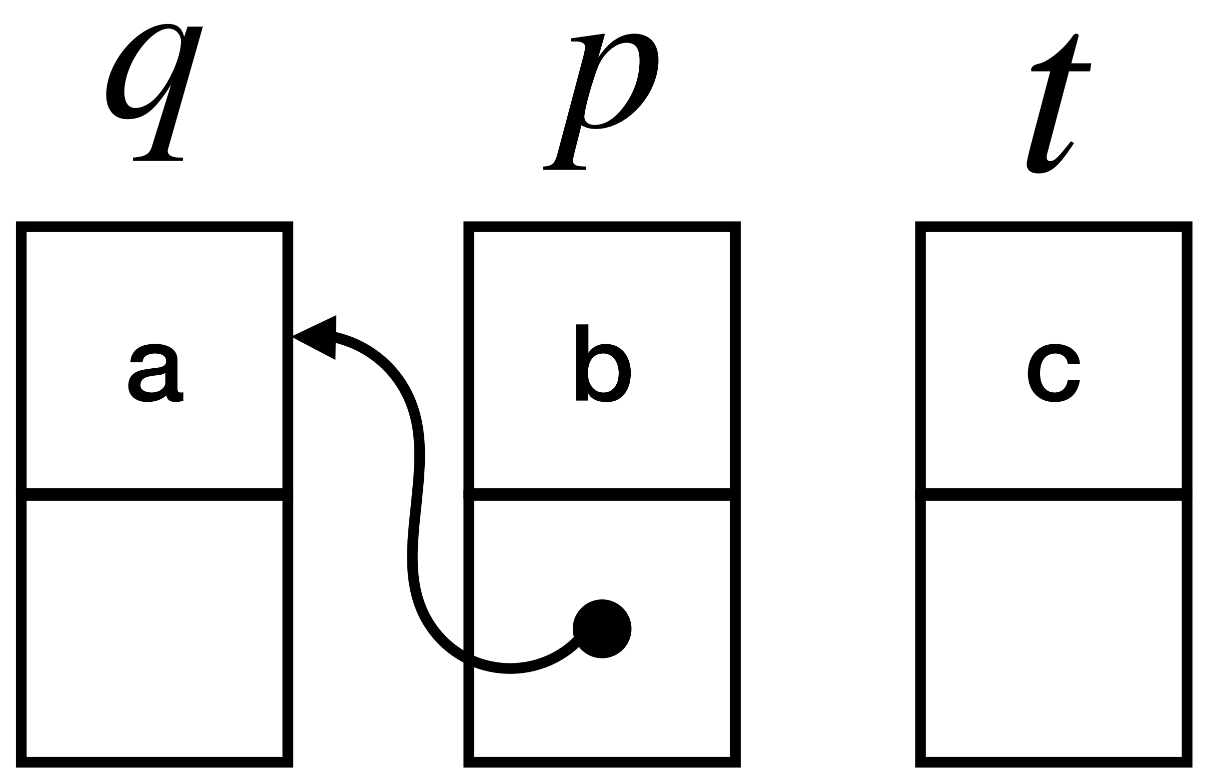
$\quad t := [p+1];$

$\quad [p+1] := q;$

$\quad q := p;$

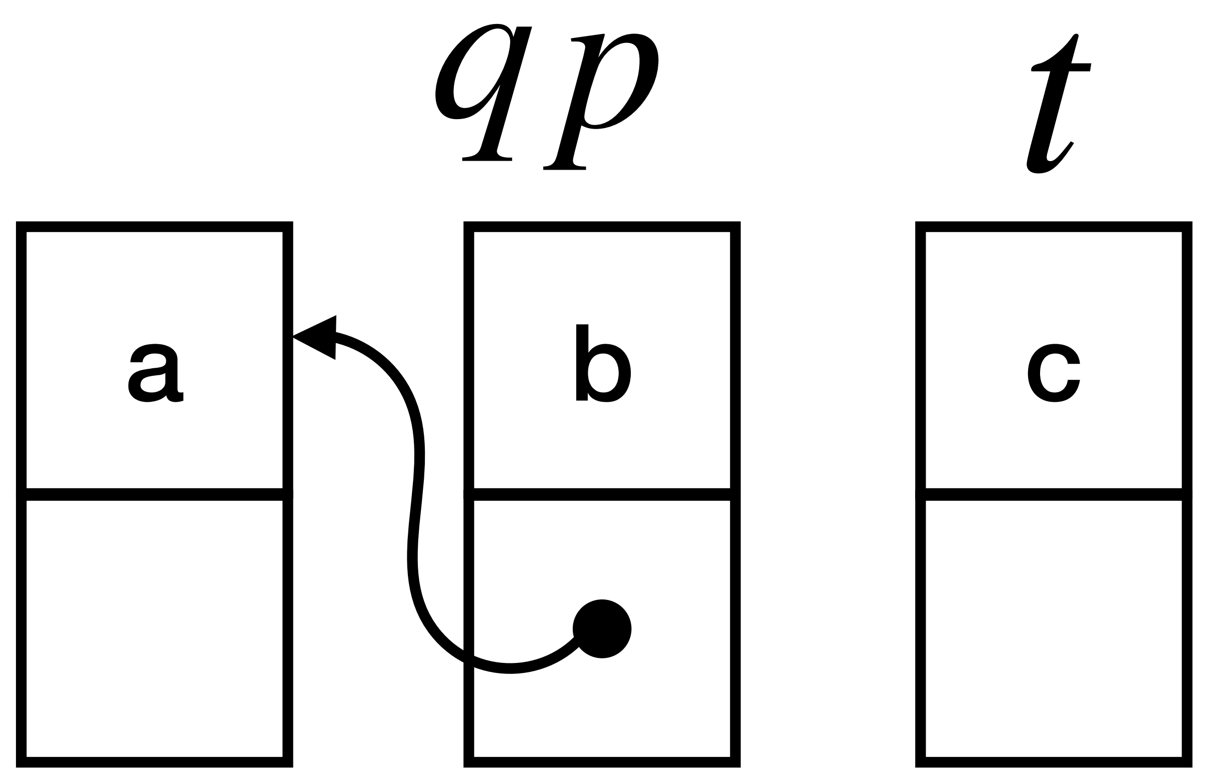
$\quad p := t$

$)$



Example

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



Example

$q := \text{nil};$

$\text{while } p \neq \text{nil do } ($

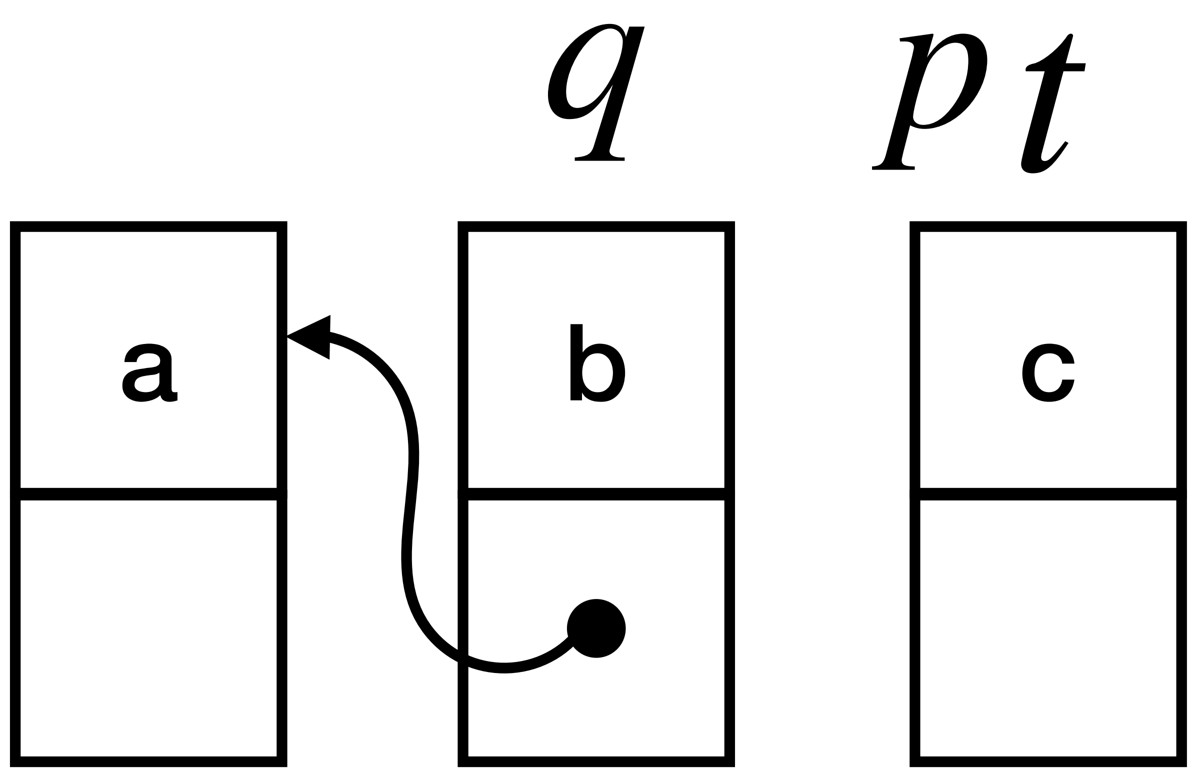
$\quad t := [p + 1];$

$\quad [p + 1] := q;$

$\quad q := p;$

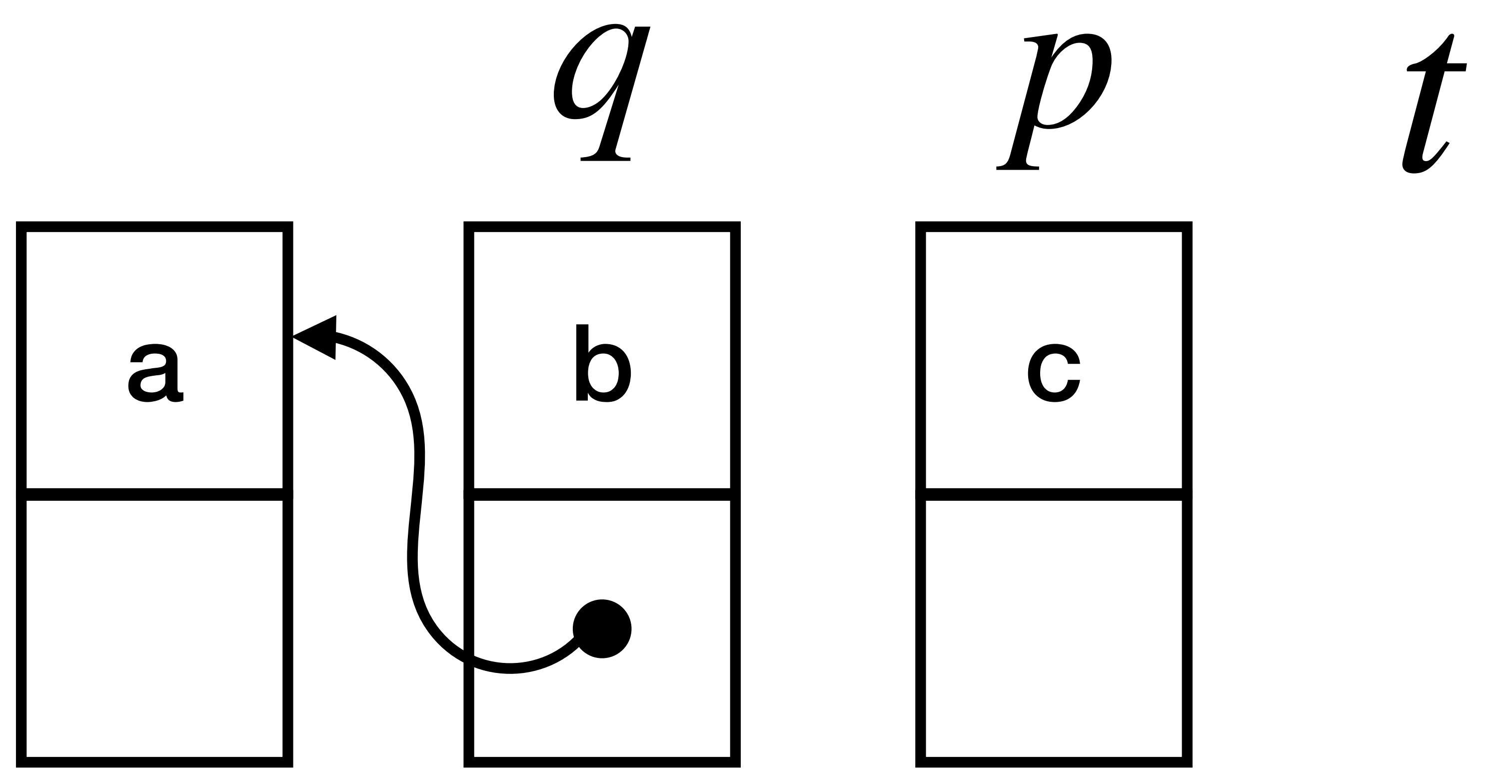
$\quad p := t$

$)$



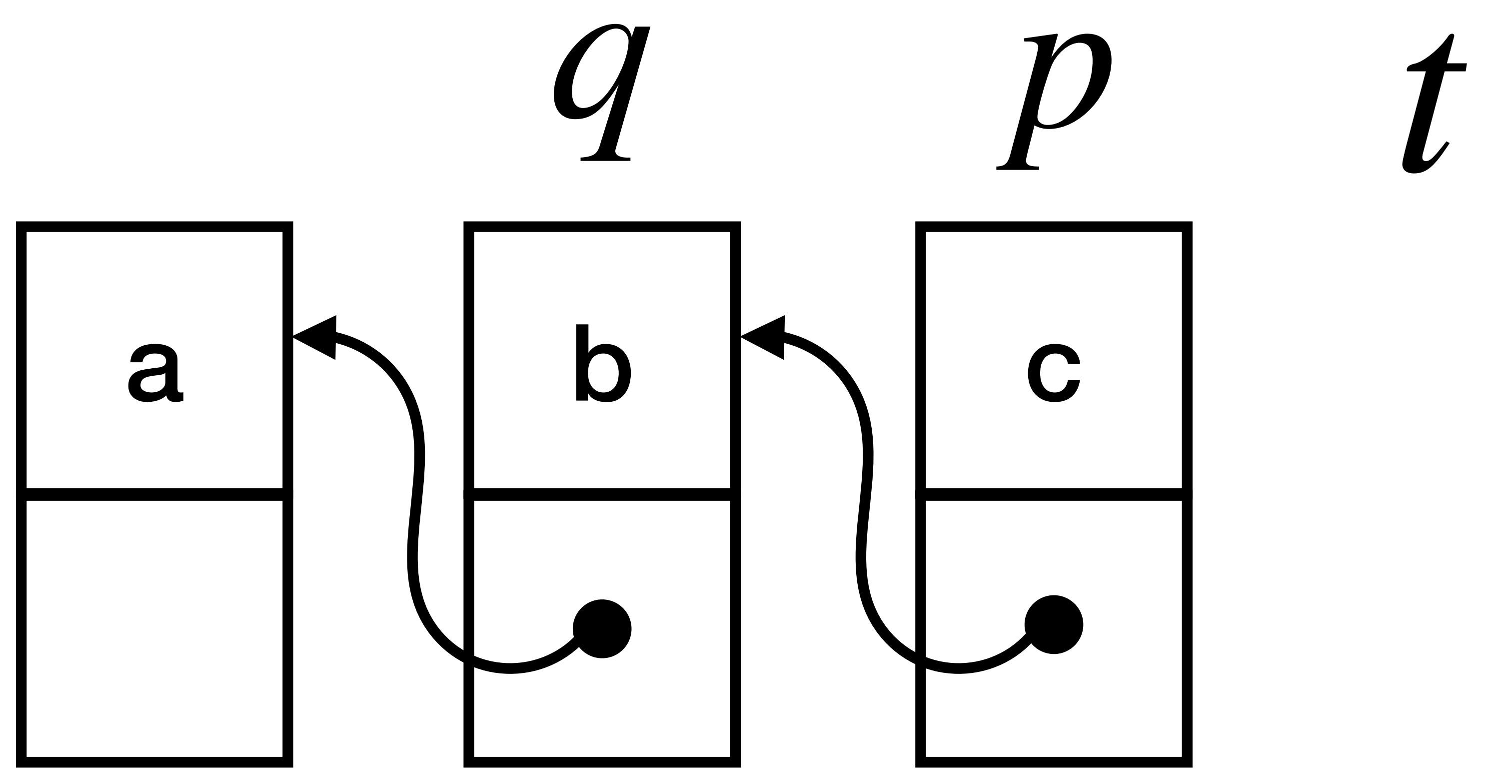
Example

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



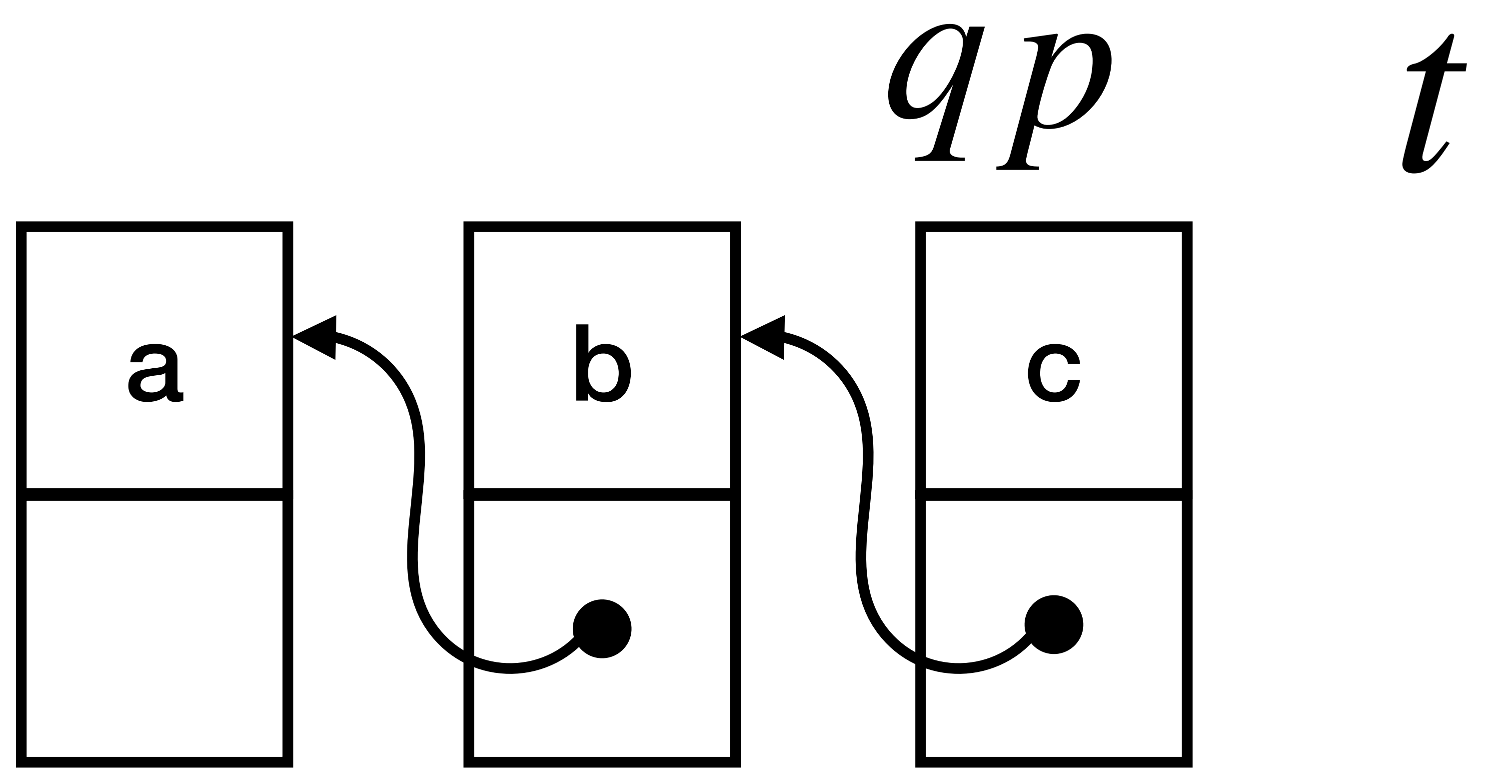
Example

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



Example

$$
\begin{aligned}
&q := \mathsf{nil};\\
&\mathsf{while}\ p \neq \mathsf{nil}\ \mathsf{do}\ (\\
&\quad t := [p+1];\\
&\quad [p+1] := q;\\
&\quad q := p;\\
&\quad p := t\\
&)
\end{aligned}
$$



Example

$q := \mathsf{nil};$

$\mathsf{while}\ p \neq \mathsf{nil}\ \mathsf{do}\ ($

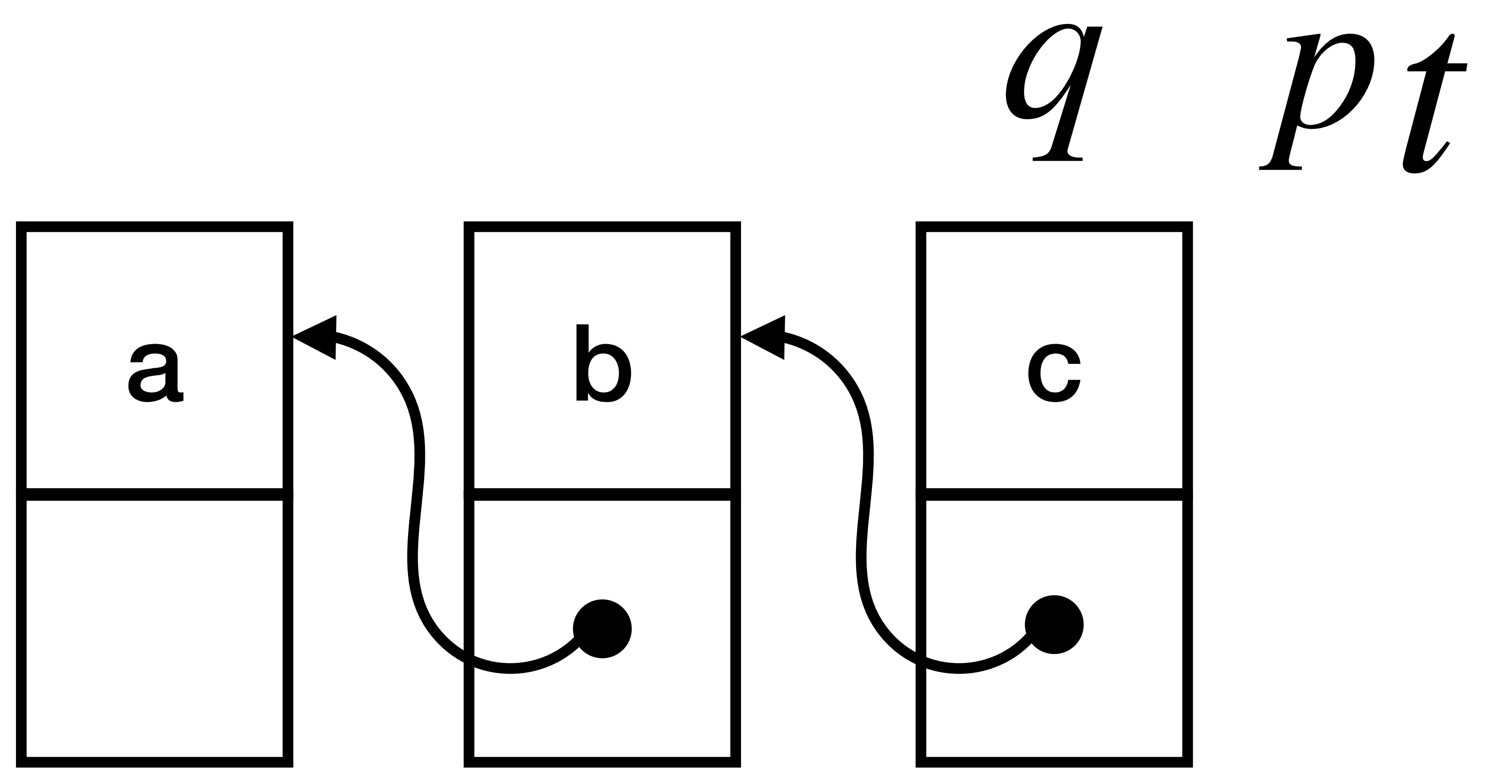
$\quad t := [p + 1];$

$\quad [p + 1] := q;$

$\quad q := p;$

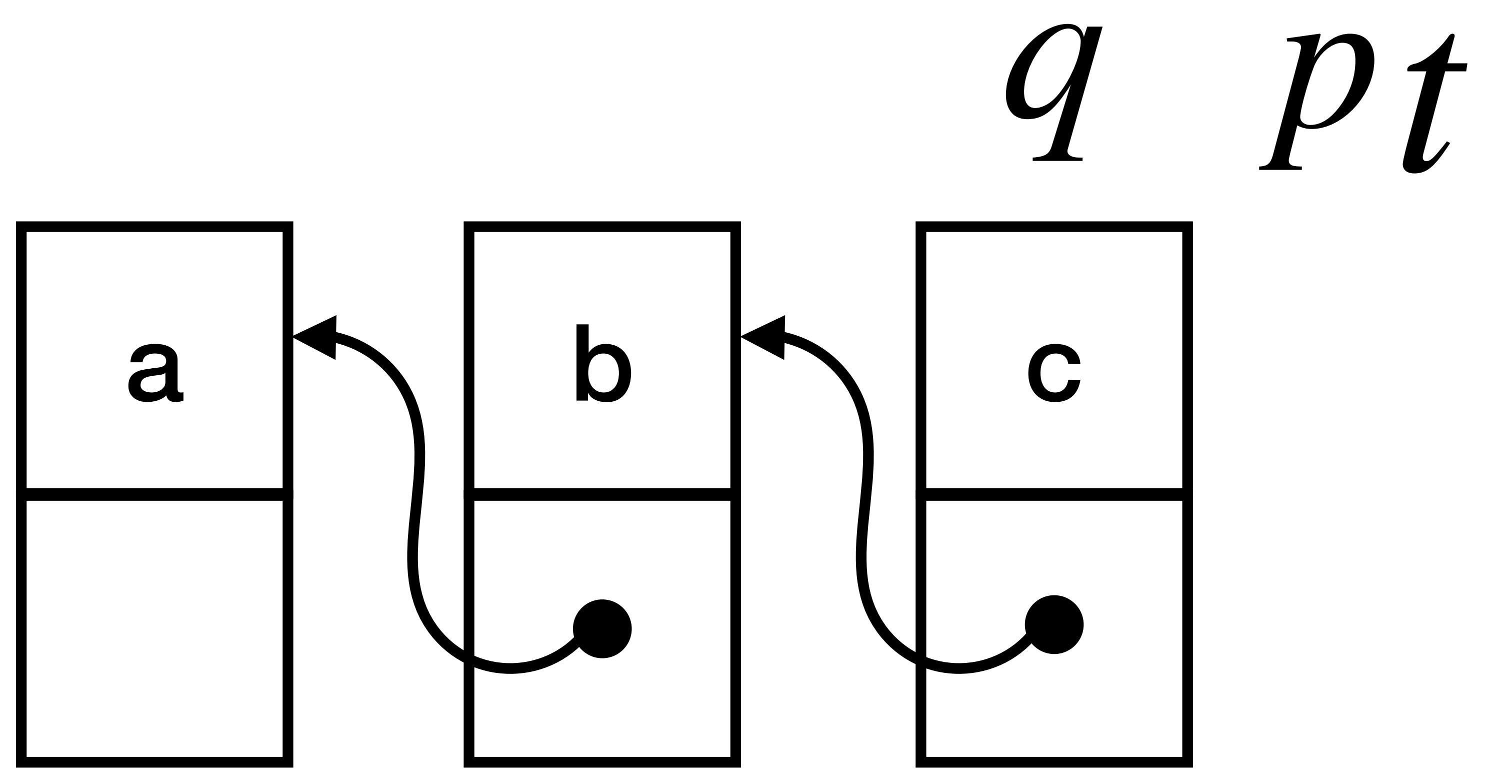
$\quad p := t$

$)$



Example

$$
\begin{array}{l}
q := \mathsf{nil}; \\
\mathsf{while}\ p \neq \mathsf{nil}\ \mathsf{do}\ (\\
\quad t := [p+1]; \\
\quad [p+1] := q; \\
\quad q := p; \\
\quad p := t \\
)
\end{array}
$$



Example

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

invariant?

reversal

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

an inductive list predicate:

points to

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

Example

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

invariant?

reversal

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

would fail if lists overlap!

Example

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

invariant?

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

do not overlap

an inductive reachability predicate:

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

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

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

invariant?

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

Example

separating conjunction!

$q := \mathsf{nil};$

$\mathsf{while}\ p \neq \mathsf{nil}\ \mathsf{do}\ ($ $P \triangleq \exists \alpha, \beta\,.\ (\mathsf{list}(\alpha, p) * \mathsf{list}(\beta, q)) \wedge \alpha_0^\dagger = \alpha^\dagger \cdot \beta$

$\quad t := [p + 1];$

$\quad [p + 1] := q;$

$\quad q := p;$

$\quad p := t$

$)$

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

$$\{\mathsf{true}\}$$

$$[x] := 1;$$

$$[y] := 2;$$

$$[z] := 3;$$

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

is it valid?

we must exclude aliasing!

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

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

$[x_1] := 1;$

$[x_2] := 2;$

$\ldots$

$[x_n] := n;$

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

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

Example

ownership of heap cell at x

separating conjunction!

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

n conjuncts!

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

separating conjunction!

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

Heap manipulating atomic commands

Stores, heaps and states

store

set of variables

set of values

$$s : X \rightharpoonup_{\text{fin}} \mathbb{Z}$$

heap

set of locations

set of values

$$h : \mathbb{N} \rightharpoonup_{\text{fin}} \mathbb{Z}_{\perp}$$

null=-1

special value (deallocated)

set of all stores

$$\text{Stores} \triangleq \{s : X \rightharpoonup_{\text{fin}} \mathbb{Z}\}$$

set of all heaps

$$\text{Heaps} \triangleq \{h : \mathbb{N} \rightharpoonup_{\text{fin}} \mathbb{Z}_{\perp}\}$$

state

store-heap pair

$$\sigma = \langle s, h \rangle$$

concrete domain

$$\wp(\text{States})$$

set of all states

$$\text{States} \triangleq \text{Stores} \times \text{Heaps}$$

Notation

$\mathrm{dom}(f)$ is the domain of definition of a heap/store function

$h_1 \# h_2 \triangleq \mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \varnothing$

$h_1 \bullet h_2$ is the union of functions with disjoint domains
(undefined if $\neg(h_1 \# h_2)$)

$f[x \mapsto n]$ is the partial function like f except that x goes to n

Regular commands

regular command

atomic command

choice

Kleene star

$$
\begin{aligned}
r ::= \;& e \\
\mid \;& r_1 ; r_2 \\
\mid \;& r_1 + r_2 \\
\mid \;& r^\star
\end{aligned}
$$

$$
\begin{aligned}
e ::= \;& \mathsf{skip} \\
\mid \;& b? \\
\mid \;& x := a \\
\mid \;& x := [a] && \text{// read} \\
\mid \;& [a_1] := a_2 && \text{// write} \\
\mid \;& x := \mathsf{alloc}(\,) \\
\mid \;& \mathsf{free}(x) \\
\mid \;& x := \mathsf{cons}(a_0, \ldots, a_k)
\end{aligned}
$$

Assertion language

Assertion language

assertion

$$
\begin{aligned}
P ::= \;& \mathsf{true} \mid \mathsf{false} \mid a_1 < a_2 \mid a_1 = a_2 \mid \ldots \\
\mid \;& \neg P \mid P_1 \wedge P_2 \mid \exists x\,.\,P \mid \ldots \\
\mid \;& \mathsf{emp} \\
\mid \;& a_1 \mapsto a_2 \\
\mid \;& P_1 * P_2
\end{aligned}
$$

Boolean and classical assertions

also called pure assertions

empty heap

structural assertions

ownership

and separately

Satisfaction: classical

$$\langle s, h \rangle \vDash P$$

the assertion P holds
for the given state $\langle s, h \rangle$

$$\langle s, h \rangle \vDash a_1 < a_2 \quad \text{iff } s \vDash a_1 < a_2$$

$$\langle s, h \rangle \vDash P_1 \wedge P_2 \quad \text{iff } \langle s, h \rangle \vDash P_1 \text{ and } \langle s, h \rangle \vDash P_2$$

$$\langle s, h \rangle \vDash \forall x \,.\, P \quad \text{iff } \forall v \in \mathbb{Z} \,.\, \langle s[x \mapsto v], h \rangle \vDash P$$

…

Satisfaction: structural

$$\langle s, h \rangle \vDash P$$

the assertion P holds
for the given state $\langle s, h \rangle$

$\langle s, h \rangle \vDash a_1 \mapsto a_2$ iff $\mathrm{dom}(h) = \{[\![a_1]\!]s\}$ and $h([\![a_1]\!]s) = [\![a_2]\!]s$

$\langle s, h \rangle \vDash \mathsf{emp}$ iff h is the empty map $[\,]$

$\langle s, h \rangle \vDash P_1 * P_2$ iff $\exists h_1, h_2 .\ \langle s, h_1 \rangle \vDash P_1$ and $\langle s, h_2 \rangle \vDash P_2$ and $h = h_1 \bullet h_2$

splits the heap
but not the store!

Example

$\text{dom}(h) = \{11{,}42\}$

$s(x) = 11 \qquad h(11) = 42$

$s(y) = 42 \qquad h(42) = 11$

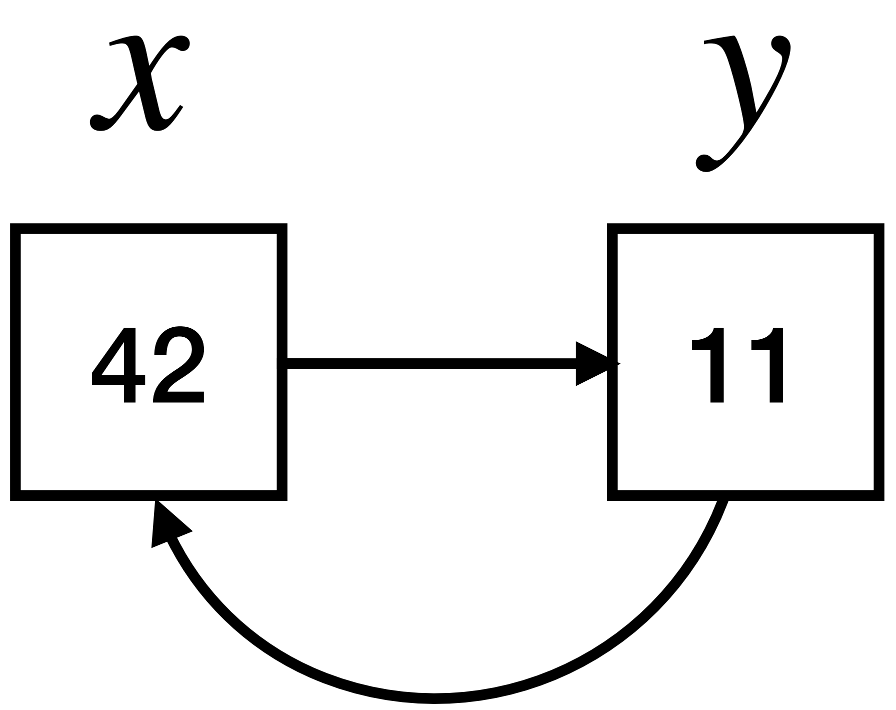
$h = h_1 \bullet h_2$

$\text{dom}(h_1) = \{11\}$

$h_1(11) = 42$

$\text{dom}(h_2) = \{42\}$

$h_2(42) = 11$



$\langle s, h \rangle \vDash \text{emp}$?

$\langle s, h \rangle \vDash x \mapsto y$?

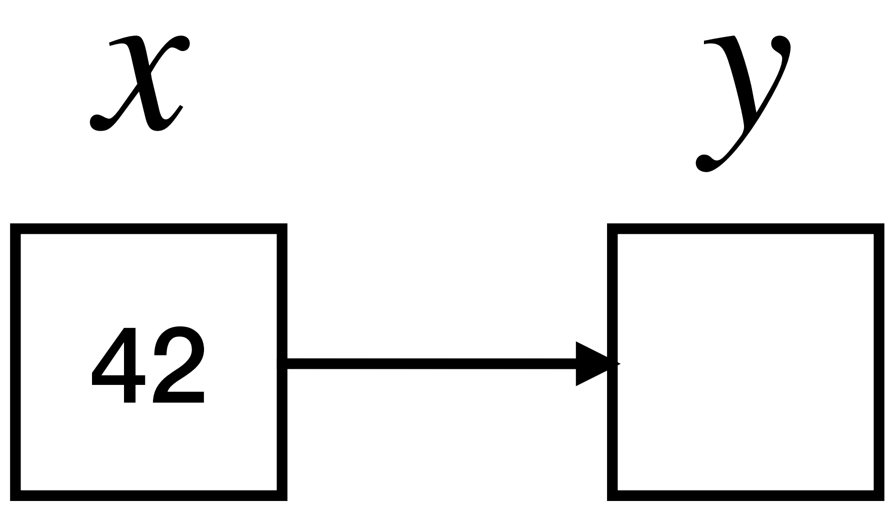
$\langle s, h \rangle \vDash x \mapsto y * y \mapsto x$?

Example

$\mathrm{dom}(h_1) = \{11\}$

$s(x) = 11 \qquad h_1(11) = 42$

$s(y) = 42$



$\langle s, h_1 \rangle \vDash \mathsf{emp}$?

$\langle s, h_1 \rangle \vDash x \mapsto y$?

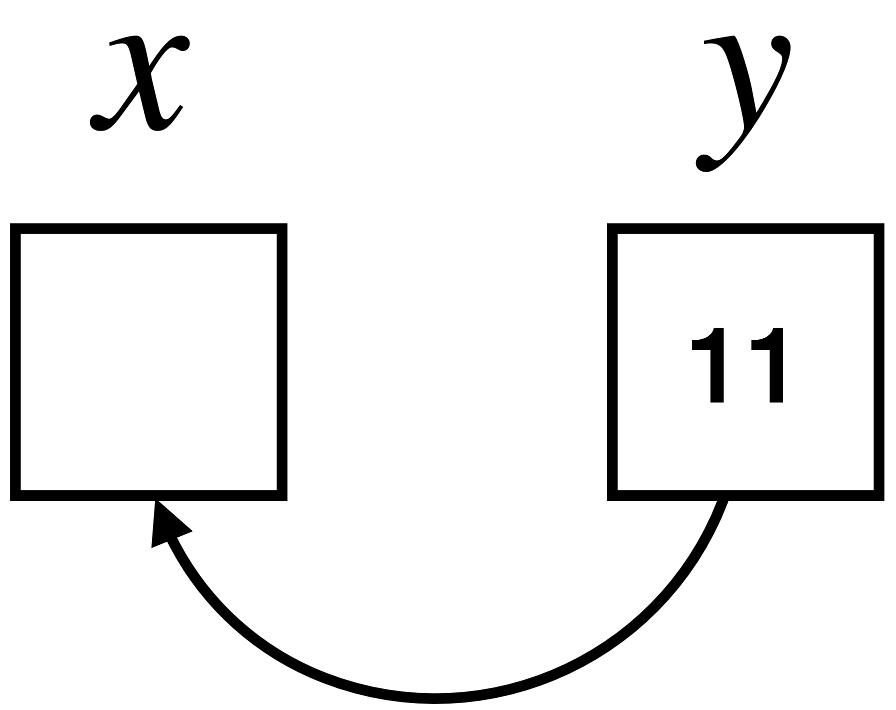
$\langle s, h_1 \rangle \vDash x \mapsto y * y \mapsto x$?

Example

$\mathrm{dom}(h_2) = \{42\}$

$s(x) = 11$

$s(y) = 42 \qquad h_2(42) = 11$



$\langle s, h_2 \rangle \vDash \mathsf{emp}$?

$\langle s, h_2 \rangle \vDash y \mapsto x$?

$\langle s, h_2 \rangle \vDash x \mapsto y * y \mapsto x$?

Example

$\mathrm{dom}(h_2) = \{42\}$

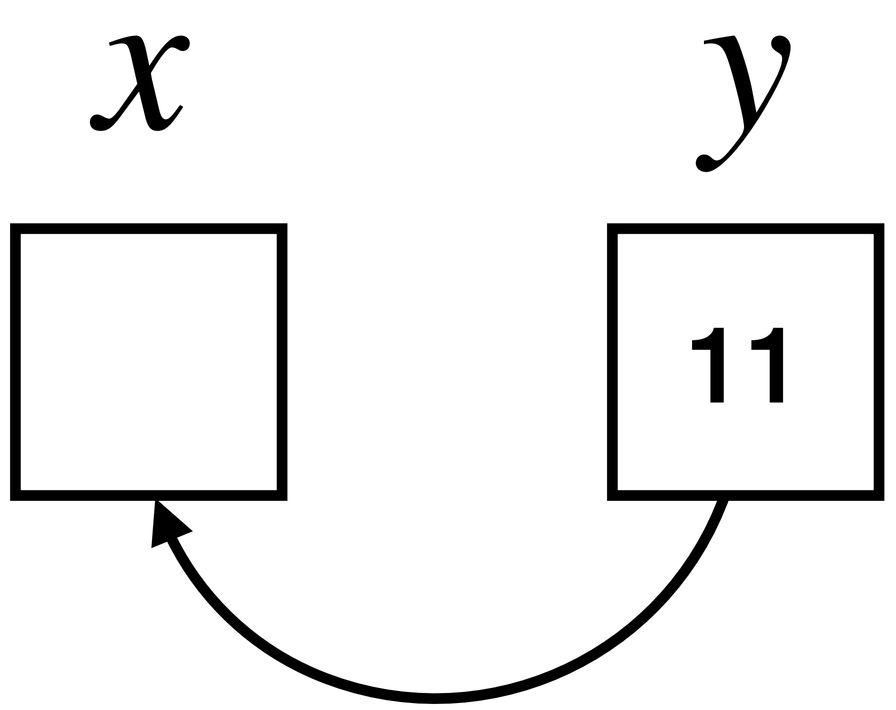
$s(x) = 11$

$s(y) = 42 \qquad h_2(42) = 11$

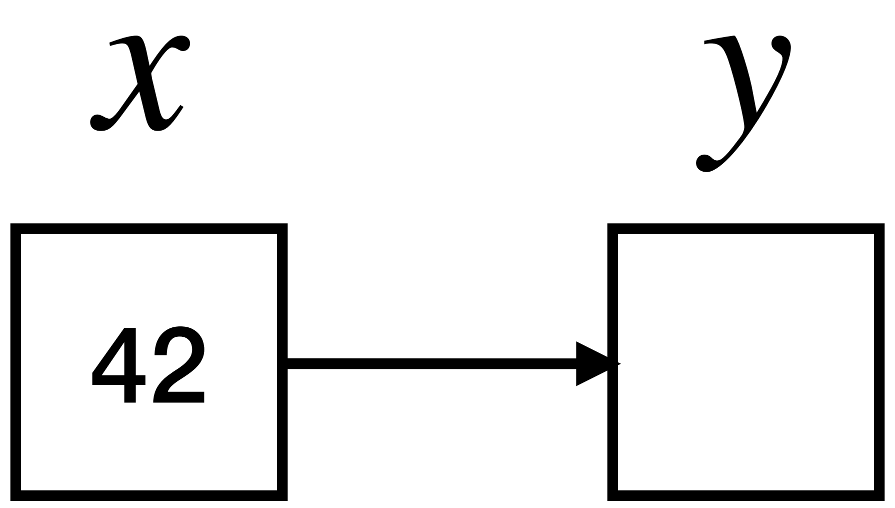
$\mathrm{dom}(h_1) = \{11\}$

$s(x) = 11 \qquad h_1(11) = 42$

$s(y) = 42$



$h = h_1 \bullet h_2$



$\langle s, h \rangle \models x \mapsto y * y \mapsto x$?

Some subtleties

any assertion

pure assertion

$P * \mathsf{emp} \equiv P$ $\qquad$ $P \land \mathsf{emp} \not\equiv P$

$x \mapsto v * \mathsf{true} \not\equiv x \mapsto v$ $\qquad$ $x \mapsto v * P \not\equiv x \mapsto v \land P$

$x \mapsto v * x \mapsto w \equiv \mathsf{false}$ $\qquad$ $x \mapsto v \land x \mapsto w \equiv x \mapsto v \land (v = w)$

$(x = y) * (x = y) \equiv (x = y)$ $\qquad$ $P * P \equiv P$

$x \mapsto v * y \mapsto w \not\Rightarrow x \mapsto v$ $\qquad$ $x \mapsto v \land y \mapsto w \Rightarrow x \mapsto v$

$x \mapsto v \not\Rightarrow x \mapsto v * y \mapsto w$ $\qquad$ $x \mapsto v \not\Rightarrow x \mapsto v \land y \mapsto w$

Example

Let us define the following inductive predicate for list segments

$$\mathsf{ls}(a_1, a_2, 0) \triangleq a_1 = a_2 \wedge \mathsf{emp}$$

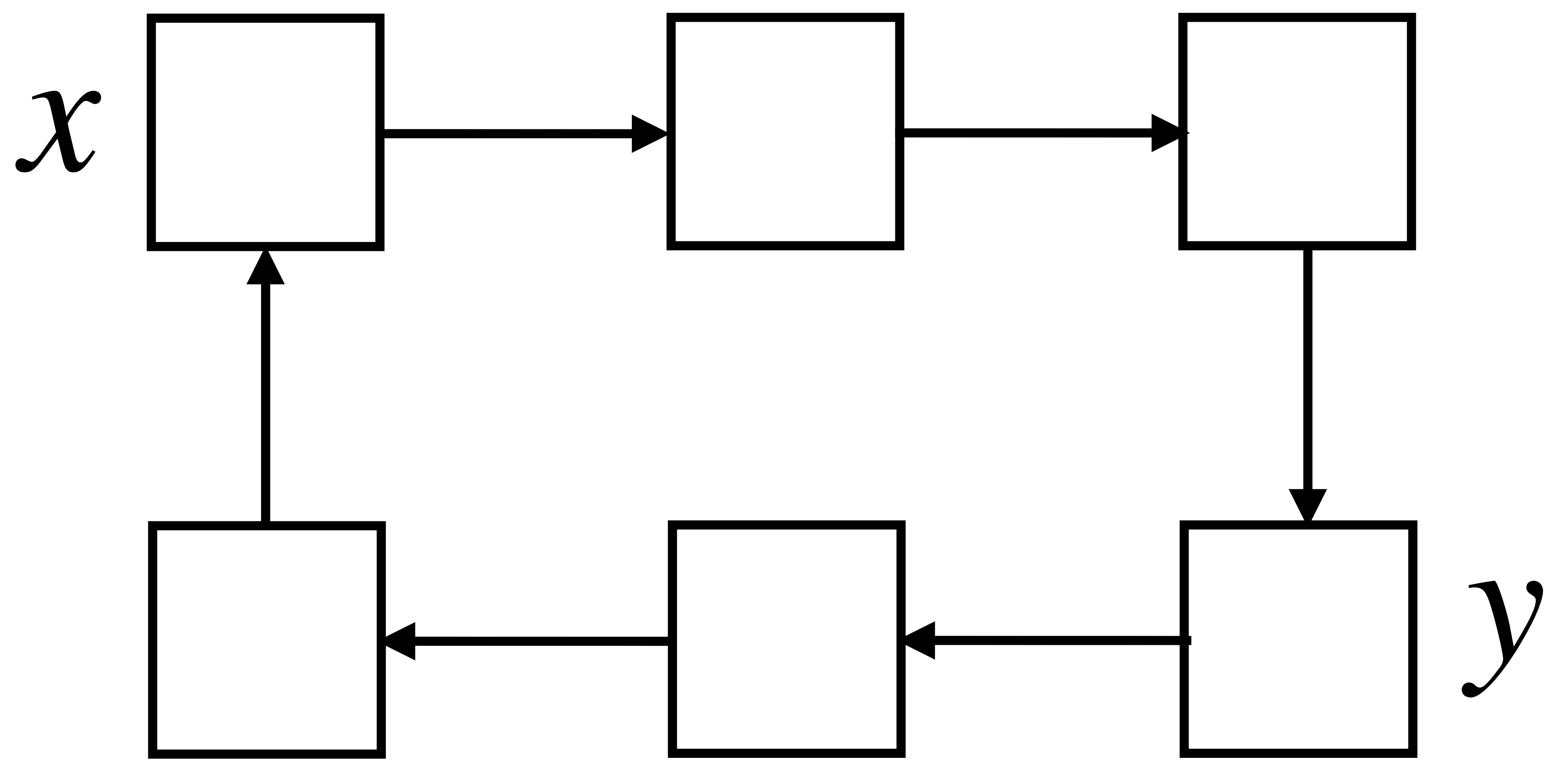
$$\mathsf{ls}(a_1, a_2, n+1) \triangleq a_1 \neq a_2 \wedge \exists x \,.\, a_1 \mapsto x * \mathsf{ls}(x, a_2, n)$$

and let $\mathsf{ls}(a_1, a_2) \triangleq \exists n \,.\, \mathsf{ls}(a_1, a_2, n)$ and $\mathsf{list}(a) \triangleq \mathsf{ls}(a, \mathsf{nil})$

Does the model in the figure satisfy $\mathsf{ls}(x, x)$? ❌

and $\mathsf{ls}(x, y) * \mathsf{ls}(y, x)$? ✅

and $\mathsf{list}(x)$? ❌



Separation Logic (SL)

CSL 2001

Local Reasoning about Programs that Alter Data Structures

Peter O'Hearn[1], John Reynolds[2], and Hongseok Yang[3]

[1] Queen Mary, University of London
[2] Carnegie Mellon University
[3] University of Birmingham and University of Illinois at Urbana-Champaign

Abstract. We describe an extension of Hoare's logic for reasoning about programs that alter data structures. We consider a low-level storage model based on a heap with associated lookup, update, allocation and deallocation operations, and unrestricted address arithmetic. The assertion language is based on a possible worlds model of the logic of bunched implications, and includes spatial conjunction and implication connectives alongside those of classical logic. Heap operations are axiomatized using what we call the "small axioms", each of which mentions only those cells accessed by a particular command. Through these and a number of examples we show that the formalism supports local reasoning: A specification and proof can concentrate on only those cells in memory that a program accesses.

This paper builds on earlier work by Burstall, Reynolds, Ishtiaq and O'Hearn on reasoning about data structures.

1 Introduction

Pointers have been a persistent trouble area in program proving. The main difficulty is not one of finding an in-principle adequate axiomatization of pointer operations; rather there is a mismatch between simple intuitions about the way that pointer operations work and the complexity of their axiomatic treatments. For example, pointer assignment is operationally simple, but when there is aliasing, arising from several pointers to a given cell, then an alteration to that cell may affect the values of many syntactically unrelated expressions. (See [20, 2, 4, 6] for discussion and references to the literature on reasoning about pointers.)

We suggest that the source of this mismatch is the global view of state taken in most formalisms for reasoning about pointers. In contrast, programmers reason informally in a local way. Data structure algorithms typically work by applying local surgeries that rearrange small parts of a data structure, such as rotating a small part of a tree or inserting a node into a list. Informal reasoning usually concentrates on the effects of these surgeries, without picturing the entire memory of a system. We summarize this local reasoning viewpoint as follows.

> To understand how a program works, it should be possible for reasoning and specification to be confined to the cells that the program actually accesses. The value of any other cell will automatically remain unchanged.

L. Fribourg (Ed.): CSL 2001, LNCS 2142, pp. 1–19, 2001.
© Springer-Verlag Berlin Heidelberg 2001

"it should be possible for reasoning and specification to be confined to the cells that the program actually accesses. The value of any other cell will automatically remain unchanged"



Some abbreviations

$$a \mapsto \_ \triangleq \exists v \,.\, a \mapsto v \qquad \text{(for } v \text{ fresh)}$$

$$a_1 \doteq a_2 \triangleq (a_1 = a_2) \wedge \mathsf{emp}$$

$$a \mapsto \langle a_0, \ldots, a_k \rangle \triangleq (a \mapsto a_0) * \ldots * (a + k \mapsto a_k)$$

Local axioms: write

$$\frac{}{\{a_1 \mapsto \_\}\ [a_1] := a_2\ \{a_1 \mapsto a_2\}}$$

$$\{x \mapsto \_\}\ [x] := y\ \{x \mapsto y\}$$

Local axioms: read

$$\frac{}{\{a \mapsto v \wedge x = x'\}\ x := [a]\ \{x = v \wedge a[x'/x] \mapsto v\}}$$

$$\{y \mapsto v\}\ x := [y]\ \{x = v \wedge y \mapsto v\}$$

Local axioms: allocation

$$\frac{}{\{\mathsf{emp}\}\ x := \mathsf{alloc}(\,)\ \{x \mapsto \_\}}$$

Local axioms: dispose

$$\overline{\{a \mapsto \_\}\ \mathsf{free}(a)\ \{\mathsf{emp}\}}$$

$$\{x \mapsto \_\}\ \mathsf{free}(x)\ \{\mathsf{emp}\}$$

Local axioms: allocation

$$\frac{}{\{x \doteq x'\}\ x := \mathsf{cons}(a_1, \ldots, a_k)\ \{x \mapsto \langle a_1[x'/x], \ldots, a_k[x'/x]\rangle\}}$$

Frame rule

in-place reasoning

$$\frac{\{P\}\ r\ \{Q\}}{\{P * R\}\ r\ \{Q * R\}}\ \mathsf{mod}(r) \cap \mathsf{free}(R) = \varnothing$$

variables possibly modified by r

Example

$$\{x \mapsto \_ * y \mapsto \_ * z \mapsto \_\}$$

frame rule

write axiom

$$\{x \mapsto \_\}$$

$$[x] := 1;$$

$$\{x \mapsto 1\}$$

$$\{x \mapsto 1 * y \mapsto \_ * z \mapsto \_\}$$

$$[y] := 2;$$

$$[z] := 3;$$

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

Example

$$\{\mathsf{list}(x) \wedge x \neq \mathsf{nil}\} \equiv \{x \mapsto v * \mathsf{list}(v)\}$$

$$\{x \mapsto v * \mathsf{list}(v)\}$$

frame $\quad \{x \mapsto v\}\ t := [x];\ \{x \mapsto v \wedge t = v\}$

$$\{x \mapsto t * \mathsf{list}(t)\}$$

frame $\quad \{x \mapsto t\}\ \mathsf{free}(x);\ \{\mathsf{emp}\}$

$$\{\mathsf{emp} * \mathsf{list}(t)\} \equiv \{\mathsf{list}(t)\}$$

$$\{\mathsf{list}(t)\}$$

$\mathsf{ls}(a_1, a_2) \triangleq (a_1 = a_2 \wedge \mathsf{emp}) \vee (a_1 \neq a_2 \wedge \exists v .\ a_1 \mapsto v * \mathsf{ls}(v, a_2))$

$\mathsf{list}(a) \triangleq \mathsf{ls}(a, \mathsf{nil})$

Example

$\{x \mapsto v * \text{list}(v) * \text{list}(y)\}$

$t := x;$

$n := [t];$

while $n \neq$ nil do (

$t := n;$

$n := [t];$

)

$[t] := y;$

$\{\text{list}(x)\}$

$\text{ls}(a_1, a_2) \triangleq (a_1 = a_2 \wedge \text{emp}) \vee (a_1 \neq a_2 \wedge \exists v\,.\; a_1 \mapsto v * \text{ls}(v, a_2))$

$\text{list}(a) \triangleq \text{ls}(a, \text{nil})$

Example

$\{x \mapsto v * \mathsf{list}(v) * \mathsf{list}(y)\}$

$\{x \mapsto v * \mathsf{list}(v)\}$

$t := x;$

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

$\{t \mapsto v\}\ n := [t];\ \{t \mapsto v \wedge n = v\}$

frame

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

invariant

while $n \neq$ nil do (

$\{\mathsf{ls}(x,t) * t \mapsto n * \mathsf{list}(n) \wedge n \neq \mathsf{nil}\} \equiv \{\mathsf{ls}(x,t) * t \mapsto n * n \mapsto w * \mathsf{list}(w)\}$

$t := n;$

$\{\mathsf{ls}(x,t') * t' \mapsto t * t \mapsto w * \mathsf{list}(w) \wedge t = n\}$

$\{t \mapsto w \wedge t = n\}\ n := [t];\ \{t \mapsto w \wedge n = w\}$

frame

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

$\{\mathsf{ls}(x,t) * t \mapsto n * \mathsf{list}(n) \wedge n = \mathsf{nil}\} \Rightarrow \{\mathsf{ls}(x,t) * t \mapsto \_\}$

$\{t \mapsto \_\}\ [t] := y;\ \{t \mapsto y\}$

frame

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

frame

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

$\mathsf{ls}(a_1,a_2) \triangleq (a_1 = a_2 \wedge \mathsf{emp}) \vee (a_1 \neq a_2 \wedge \exists v\,.\ a_1 \mapsto v * \mathsf{ls}(v,a_2))$

$\mathsf{list}(a) \triangleq \mathsf{ls}(a,\mathsf{nil})$

Correctness and (in)completeness

Relational semantics

$[\![\mathsf{skip}]\!] \triangleq \{(\sigma, \sigma)\}$
$[\![b?]\!] \triangleq \{(\sigma, \sigma) \mid \sigma = \langle s, h\rangle \wedge s \vDash b\}$
$[\![x := a]\!] \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto [\![a]\!]s], h\rangle)\}$

$[\![x := [a]]\!] \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto v], h\rangle) \mid v = h([\![a]\!]s) \in \mathbb{Z}\}$

$[\![[a_1] := a_2]\!] \triangleq \{(\langle s, h\rangle, \langle s, h[[\![a_1]\!]s \mapsto [\![a_2]\!]s]\rangle) \mid h([\![a_1]\!]s) \in \mathbb{Z}\}$

$[\![x := \mathsf{alloc}(\,)]\!] \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto n], h[n \mapsto v]\rangle) \mid v \in \mathbb{Z} \wedge (n \notin \mathsf{dom}(h) \vee h(n) = \bot\,)\}$

$[\![\mathsf{free}(x)]\!] \triangleq \{(\langle s, h \bullet [s(x) \mapsto v]\rangle, \langle s, h\rangle) \mid s(x) \in \mathbb{N} \wedge v \in \mathbb{Z}\}$

$[\![x := \mathsf{cons}(a_0, \ldots, a_k)]\!] \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto n], h[n \mapsto [\![a_0]\!]s, \ldots, (n+k) \mapsto [\![a_k]\!]s]\rangle)$
$\mid (\forall i \in [n, n+k]\,.\; i \notin \mathsf{dom}(h) \vee h(i) = \bot\,)\}$

Correctness

Th. [*correctness*]
If $\{P\}\ r\ \{Q\}$ then $[\![r]\!]P \subseteq Q$

Proof. By induction on the derivation.

Incompleteness

Th. *[incompleteness]*
There exists valid SL triples that are not provable

Proof. Misses footprint theorem: see slides on ISL

Final considerations on SL

Simple Proofs for Simple Programs

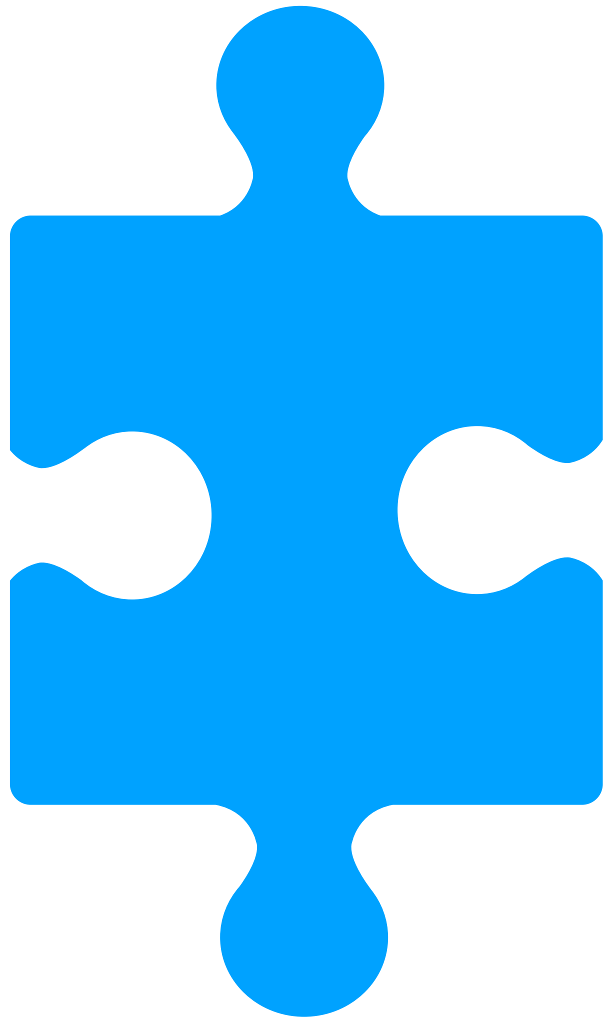
SL address resource manipulation

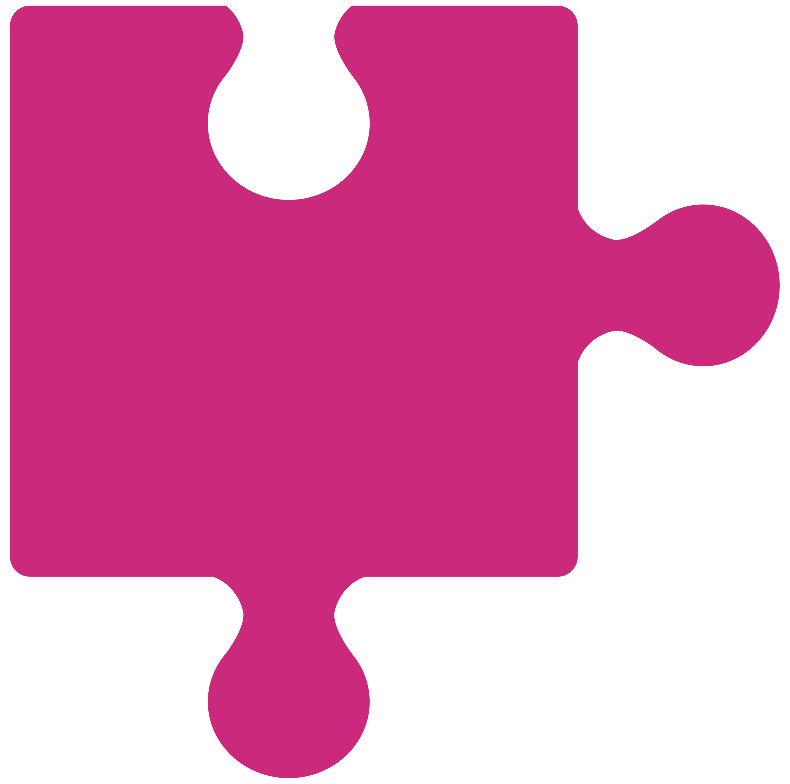
separating conjunction for in-place reasoning

pre/post describe local surgeries

Coming next

Let us explore some combination of different deductive systems



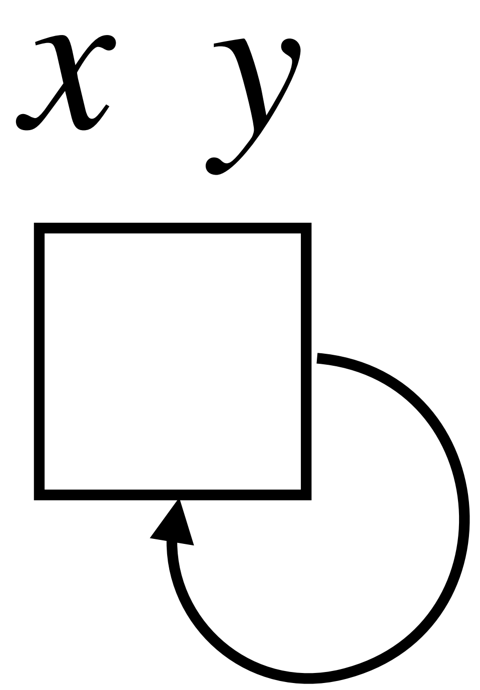


Questions

Question 1

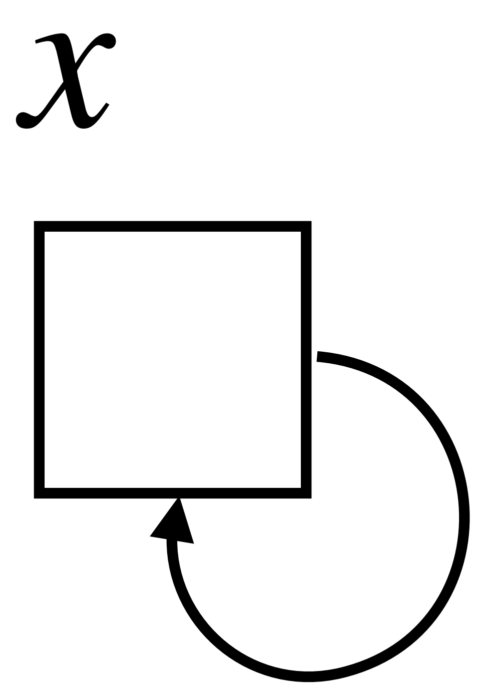
Can you find some state that satisfies the following assertions?

$(x \doteq y) * x \mapsto y$



$(x = y) * x \mapsto y$

$(x = y) \land x \mapsto y$



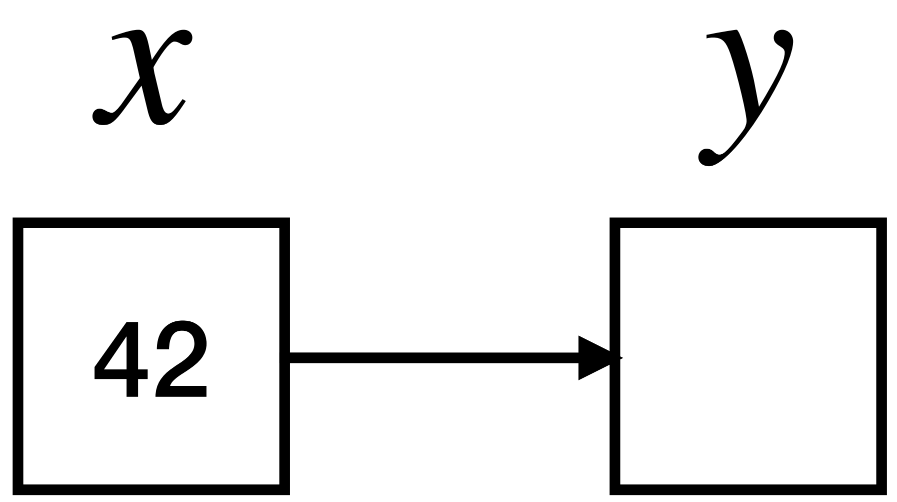
$x \mapsto x$

Question 2

Show a state that satisfies the assertion $(x \mapsto y) * \neg(x \mapsto y)$

$$\mathrm{dom}(h) = \{11\}$$

$$s(x) = 11 \qquad h(11) = 42$$
$$s(y) = 42$$



* Exam 13

Consider the imprecise list segment definition below

$\mathsf{ils}(a_1, a_2) \triangleq (a_1 = a_2 \wedge \mathsf{emp}) \vee (\exists v \,.\, a_1 \mapsto v * \mathsf{ls}(v, a_2))$

Prove that $\mathsf{ils}(a_1, a_2) \not\equiv \mathsf{ls}(a_1, a_2)$ by finding a state that distinguishes $\mathsf{ls}(11,11)$ from $\mathsf{ils}(11,11)$

* Exam 14

Complete the following derivations, if possible

$\{P * x \mapsto \_\}\ [x] := 11\ \{P * ??\}$

$\{\mathsf{true}\}\ [x] := 11\ \{??\}$

$\{P * x \mapsto \_\}\ \mathsf{free}(x)\ \{??\}$

$\{\mathsf{true}\}\ \mathsf{free}(x)\ \{??\}$