Synthesis, Analysis, and Verification

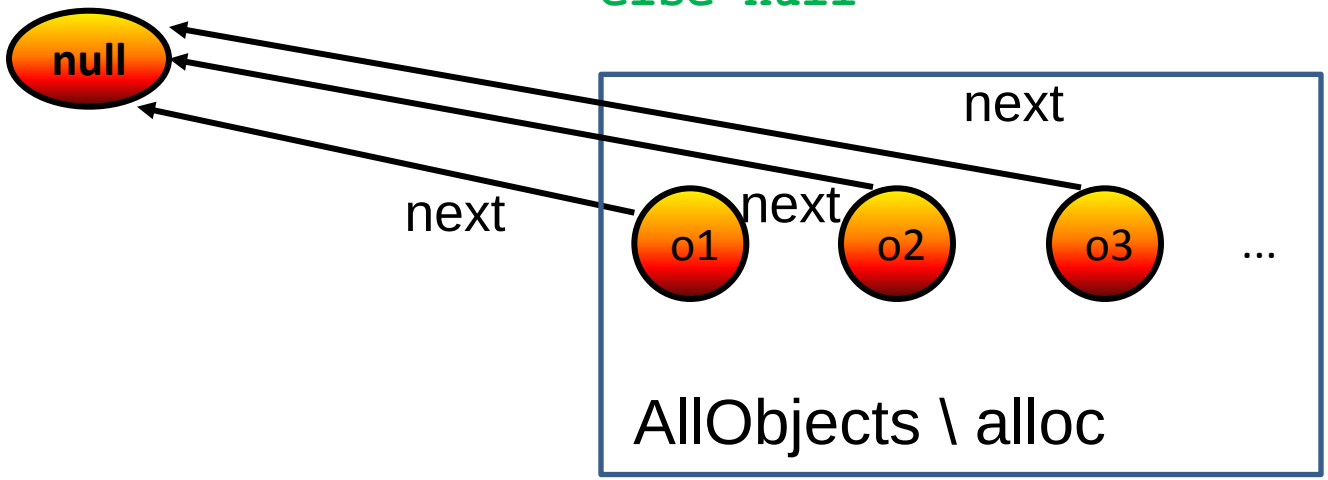
Lecture 13

Dynamic Allocation

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

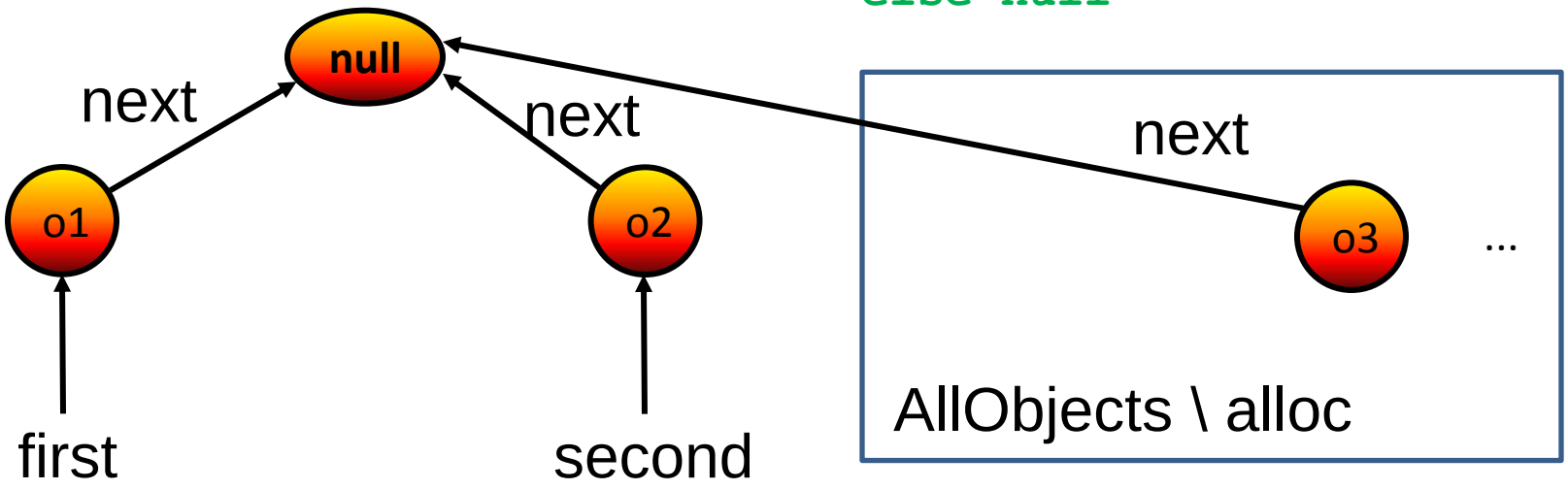
Linked List Example

```
class List { List next; }
public static void main(){ //alloc={}, next=λx.null
  List first = new List(); //alloc={o1}, next=λx.null
  List second = new List();//alloc={o1,o2}, next=λx.null
  first.next = second;     //alloc={o1,o2},
                             next=λx.if(x==o1) o2 else null
  second.next = first;     //alloc={o1,o2},
                             next=λx.if(x==o2) o1
                                    else if (x==o1) o2
                                    else null
}
```



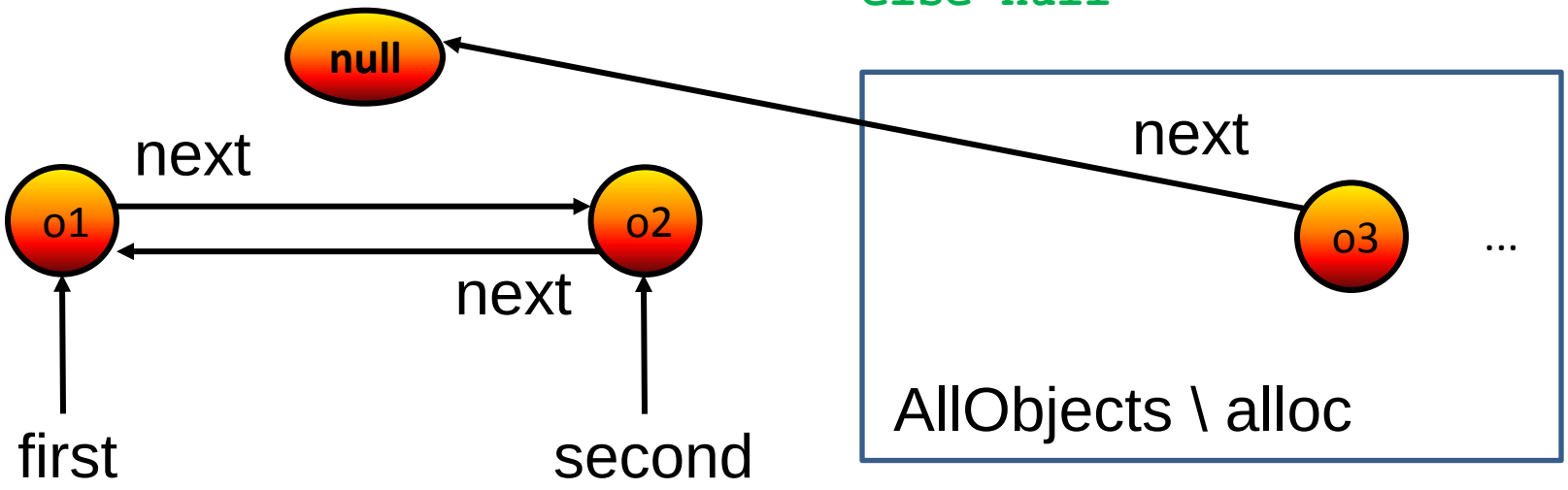
Linked List Example

```
class List { List next; }
public static void main(){ //alloc={}, next=λx.null
  List first = new List(); //alloc={o1}, next=λx.null
  List second = new List();//alloc={o1,o2}, next=λx.null
  first.next = second;     //alloc={o1,o2},
                             next=λx.if(x==o1) o2 else null
  second.next = first;     //alloc={o1,o2},
                             next=λx.if(x==o2) o1
                                     else if (x==o1) o2
                                     else null
}
```



Linked List Example

```
class List { List next; }
public static void main(){ //alloc={}, next=λx.null
  List first = new List(); //alloc={o1}, next=λx.null
  List second = new List();//alloc={o1,o2}, next=λx.null
  first.next = second;     //alloc={o1,o2},
                             next=λx.if(x==o1) o2 else null
  second.next = first;     //alloc={o1,o2},
                             next=λx.if(x==o2) o1
                                    else if (x==o1) o2
                                    else null
}
```



Memory Allocation in Java

```
x = new C();
y = new C();
assert(x != y);   // fresh object references-distinct
```

Why should this assertion hold?

How to give meaning to ‘new’ so we can prove it?

A View of the World

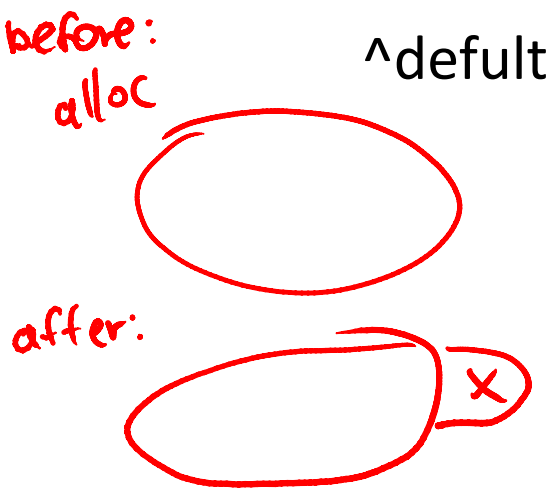
Everything exists, and will always exist.
(It is just waiting for its time to become allocated.)

It will never die (but may become unreachable).

alloc : Obj → Boolean i.e. alloc : Set[Obj]

x = new C(); → havoc(x);
assume(x ∉ alloc);
alloc = alloc ∪ {x};

^defult constructor



New Objects Point Nowhere

class C { int f; C next; C prev; }

this should work:

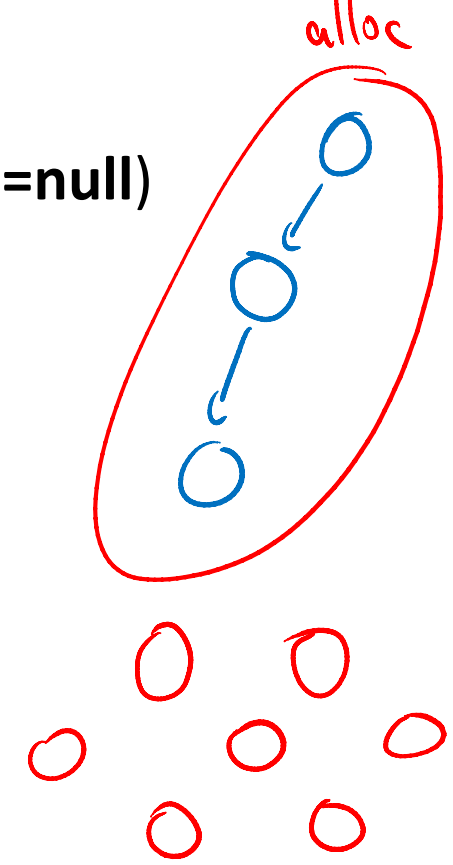
```
x = new C();
assert(x.f==0 && x.next==null  && x.prev==null)
```

x = new C(); →

1) use assigument
   f := f(x := 0)

2) use assume

```
[ havoc(x)
  assume (x ∉ alloc)
  alloc = alloc ∪ {x}
]
assume ( f(x)==0 ∧
         next(x)=null ∧
         prev(x)=null);
```



If you are new, you are known by few

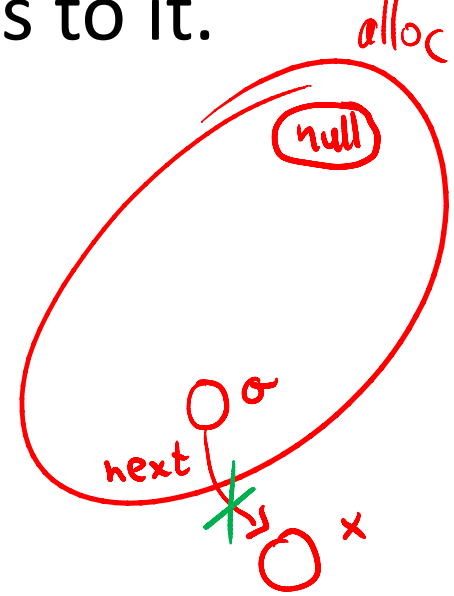
class C { int f; C next; C prev; }

Assume C is the only class in the program

Lonely object: no other object points to it.

Newly allocated objects are lonely!

x = new C(); →



∀σ. σ ∈ alloc → next(σ) ∈ alloc ∧ prev(σ) ∈ alloc

Remember our Model of Java Arrays

```
class Array {
  int length;
  data : int[]
}
```

length : Array -> int

data : Array -> (Int -> Int)

or simply: Array x Int -> Int

a[i] = x ➔

assert (a ≠ null);
assert (0 ≤ i ∧ i < length(a));
data= data((a,i):= x)

y = a[i] ➔

assert (a ≠ null);
assert (0 ≤ i ∧ i < length(a))
y = data((a, i))

Allocating New Array of Objects

```
class oArray {
  int length;
  data : Object[]
}
x = new oArray[100] →
```

n

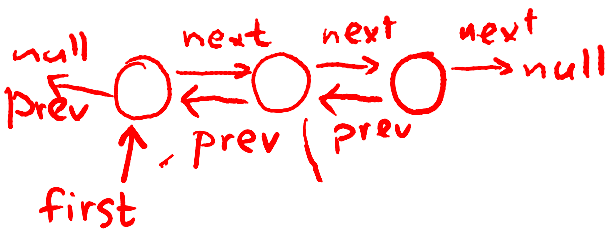
length : Object → int

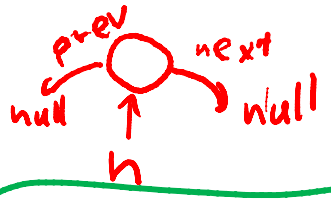
data : Object × int → Object

havoc(x); assume(x ∉ alloc);
alloc = alloc ∪ {x};
length = length(x := 100);
data = λ(σ, i). if (σ = x ∧ 0 ≤ i < 100)
null
else data(σ, i)

∀σ. next(σ) ≠ x

D-Linked List






```
assume P;
if (first == null) {
  first = n;
  n.next = null;
  n.prev = null;
} else {
  n.next = first;
  first.prev = n;
  n.prev = null;
  first = n;
}
assert Q;
```

$n \neq null \wedge$
$next(n) = null \wedge prev(n) = null \wedge$
$prev(first) = null \ \wedge \ Q$

P

$\wedge$ first $\neq$ null

$\{P\}\ r\ \{Q\}$
$\forall x,x'.\ P(x) \wedge r(x,x') \rightarrow Q(x')$

assume(first≠null)

$next = \lambda x.\ if(x = n)\ first\ else\ next(x)$
$= next(n := first)$

$prev = prev(first := n)$
$prev = prev(n := null)$
$first = n$

$\wedge$

$next' = next(n := first) \wedge$
$prev' = (prev(first := n))(n := null)$
$first' = n$

$\rightarrow$

$f(p := v) = \lambda x.\ if(x = p)\ v\ else\ f(x)$

$\forall x.\forall y.prev'(x) = y \ \rightarrow$
$(y \neq null \rightarrow next'(y) = x) \wedge$
$(y = null \wedge x \neq null \rightarrow (\forall z.next'(z) \neq x))$

Q

How to prove such verification conditions automatically?