5.3 Recursive definitions and structural induction

Recall factorial function:

$n! = n\,(n-1)\,\ldots\,1$, for $n \geq 0$ and $0! = 1$

Then $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$

$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$

5.3 Recursive definitions and structural induction

Recall factorial function:

$f(n) = n! = n\,(n-1)\,\ldots\,1$, for $n \geq 0$ and $f(0) = 0! = 1$

Then $f(4) = 4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$

$f(5) = 5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$

Recursive definition of factorial function:

$f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition

$f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

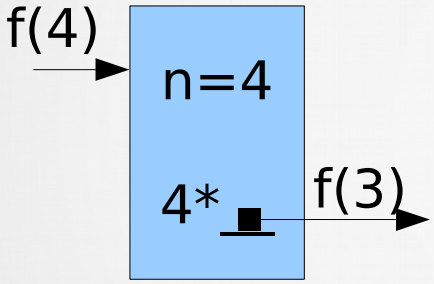
5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition

$f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$



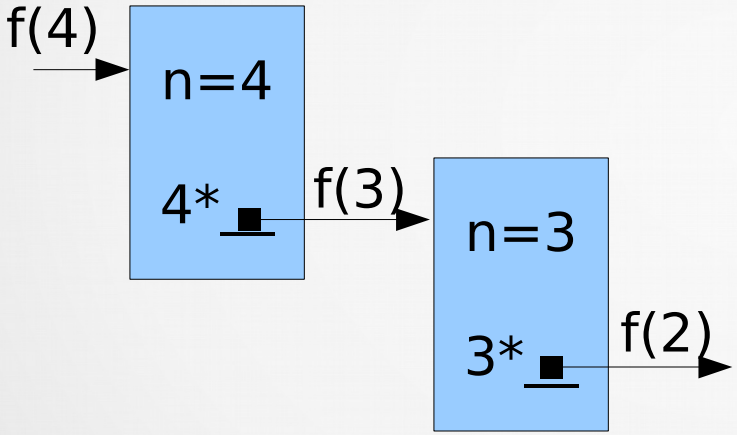
5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition

$f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$



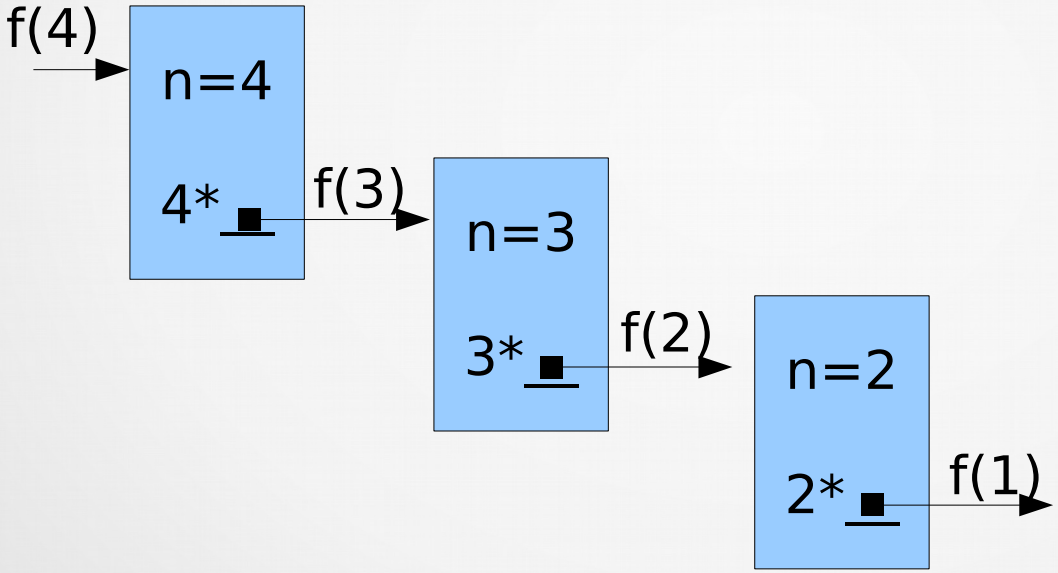
5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition

$f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

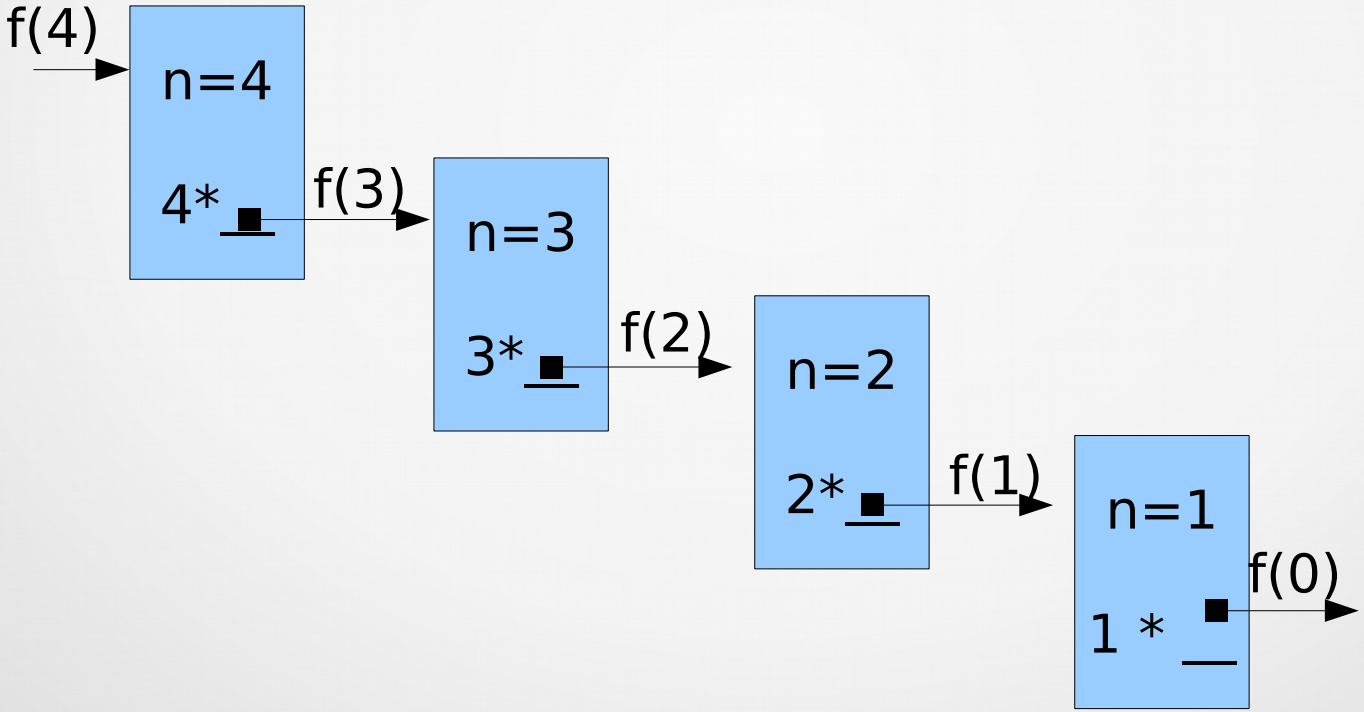


5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition $f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

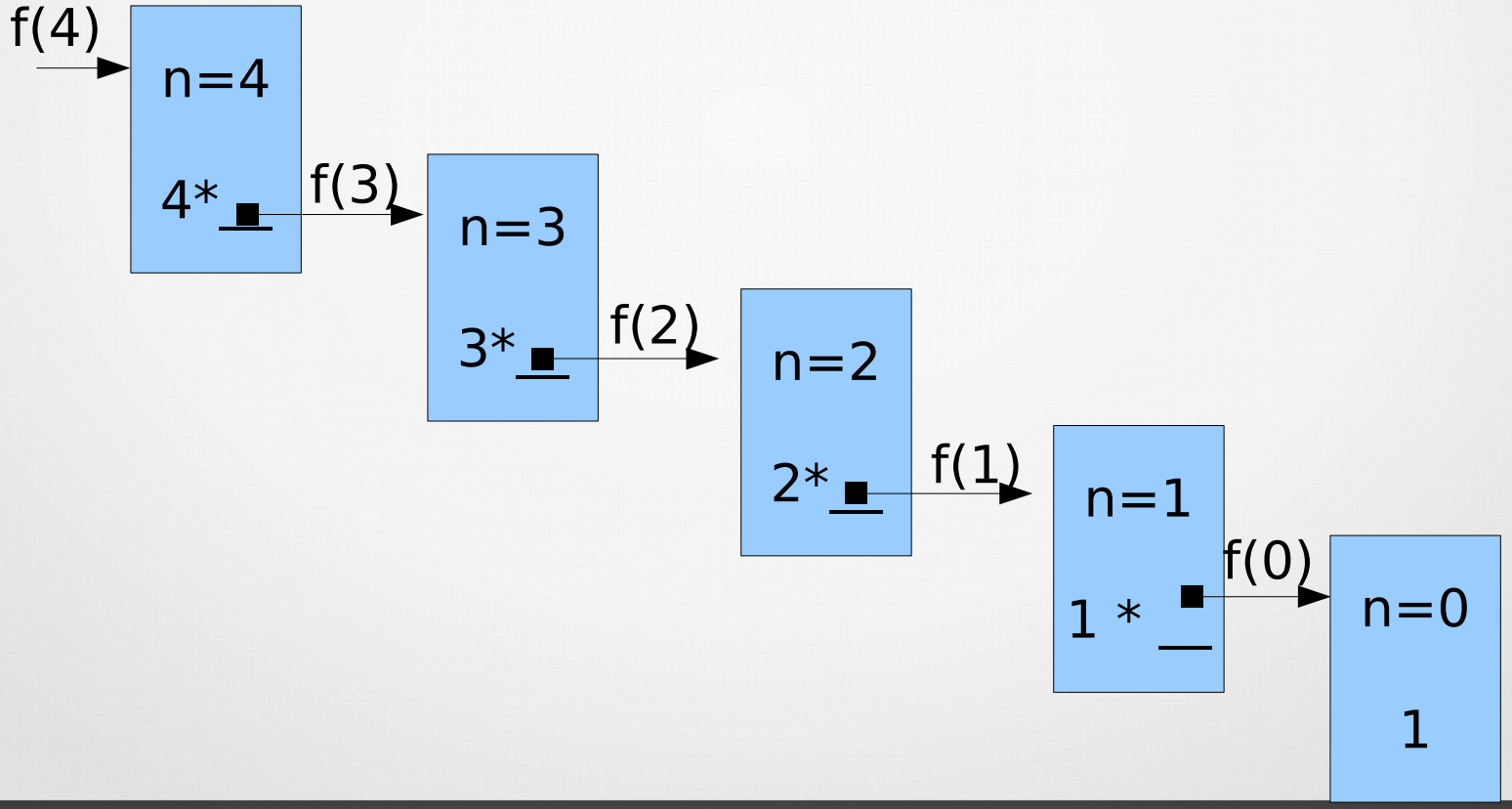


5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition $f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

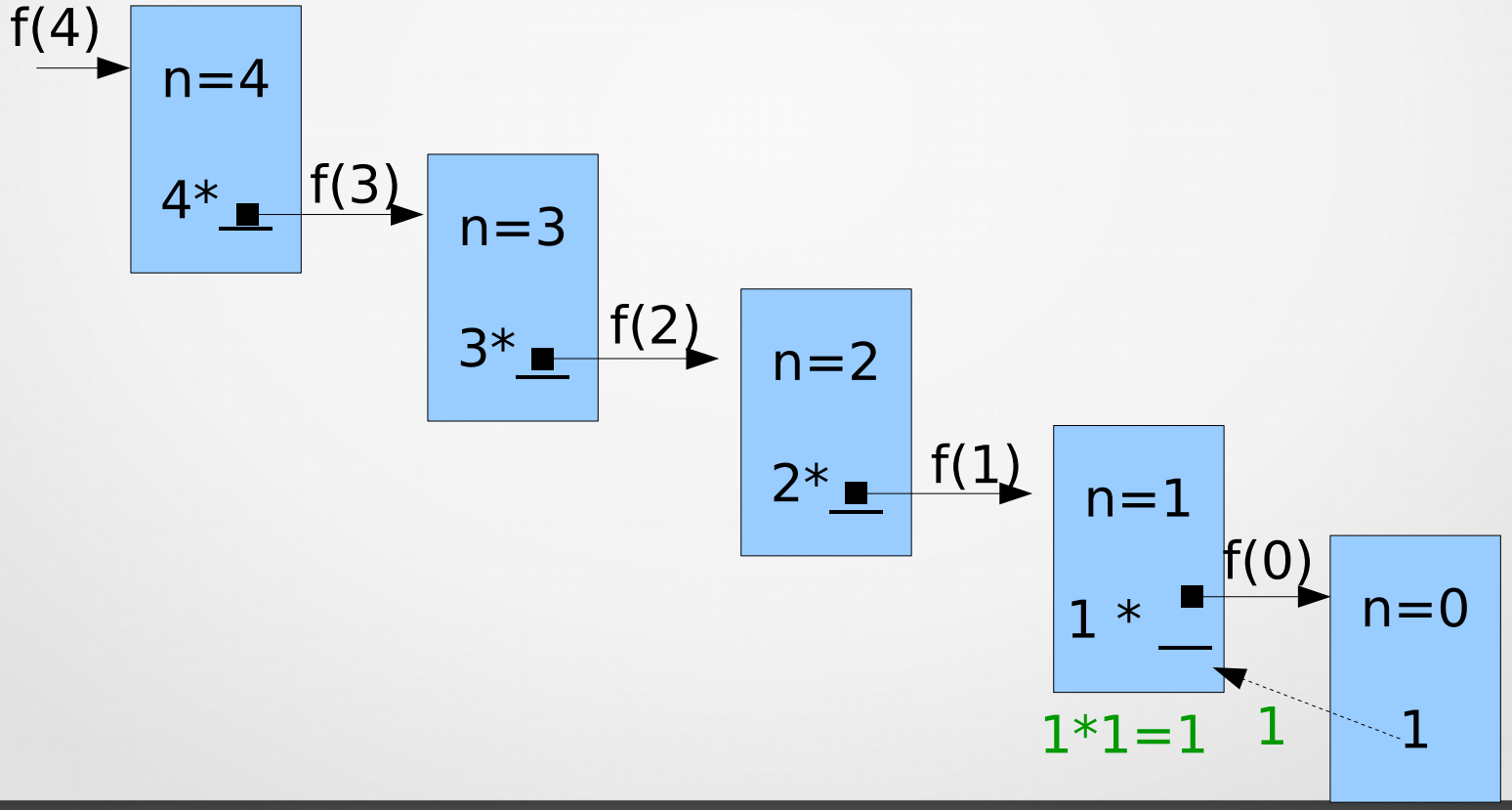


5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition $f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

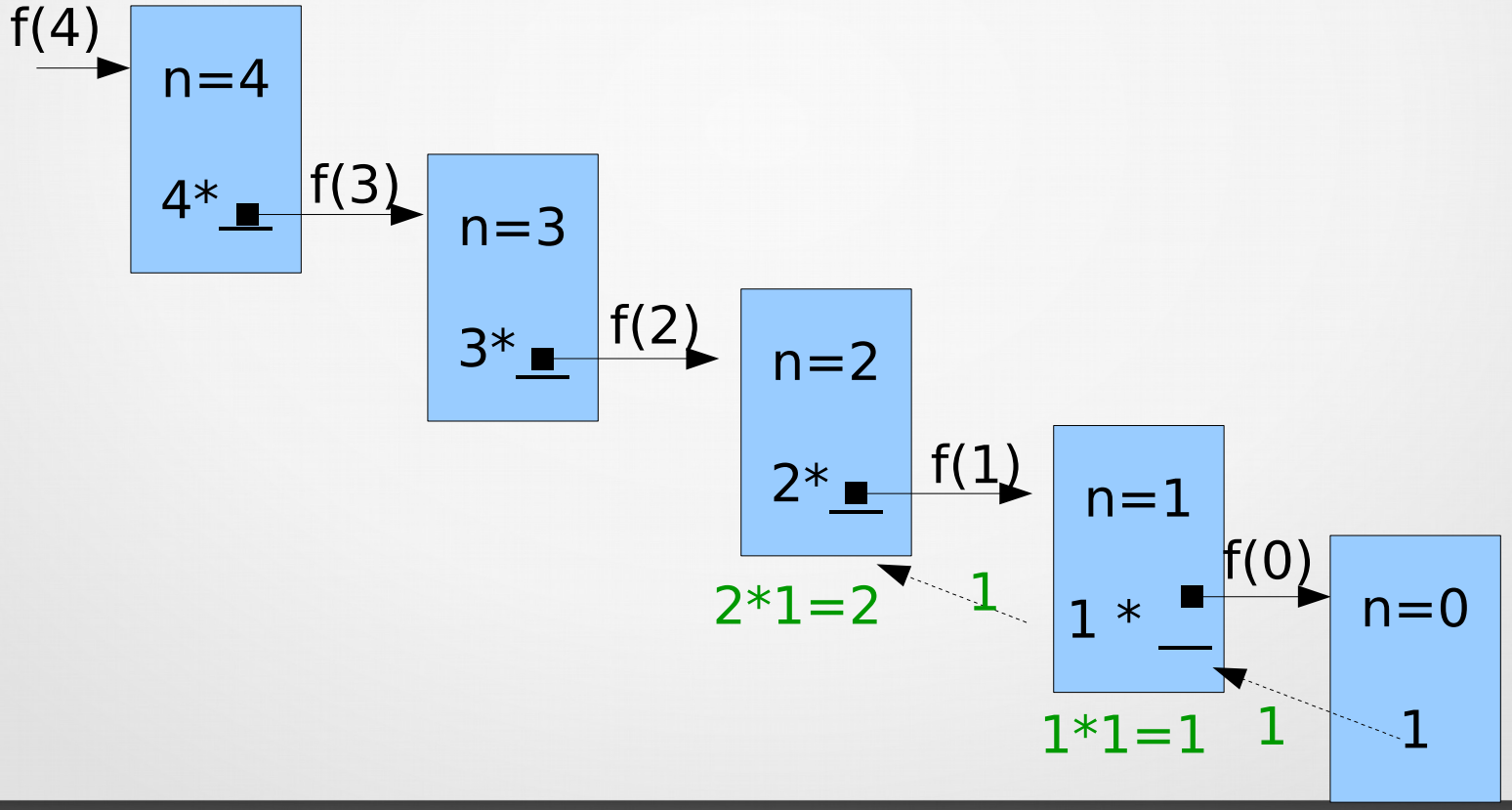


5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition $f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$



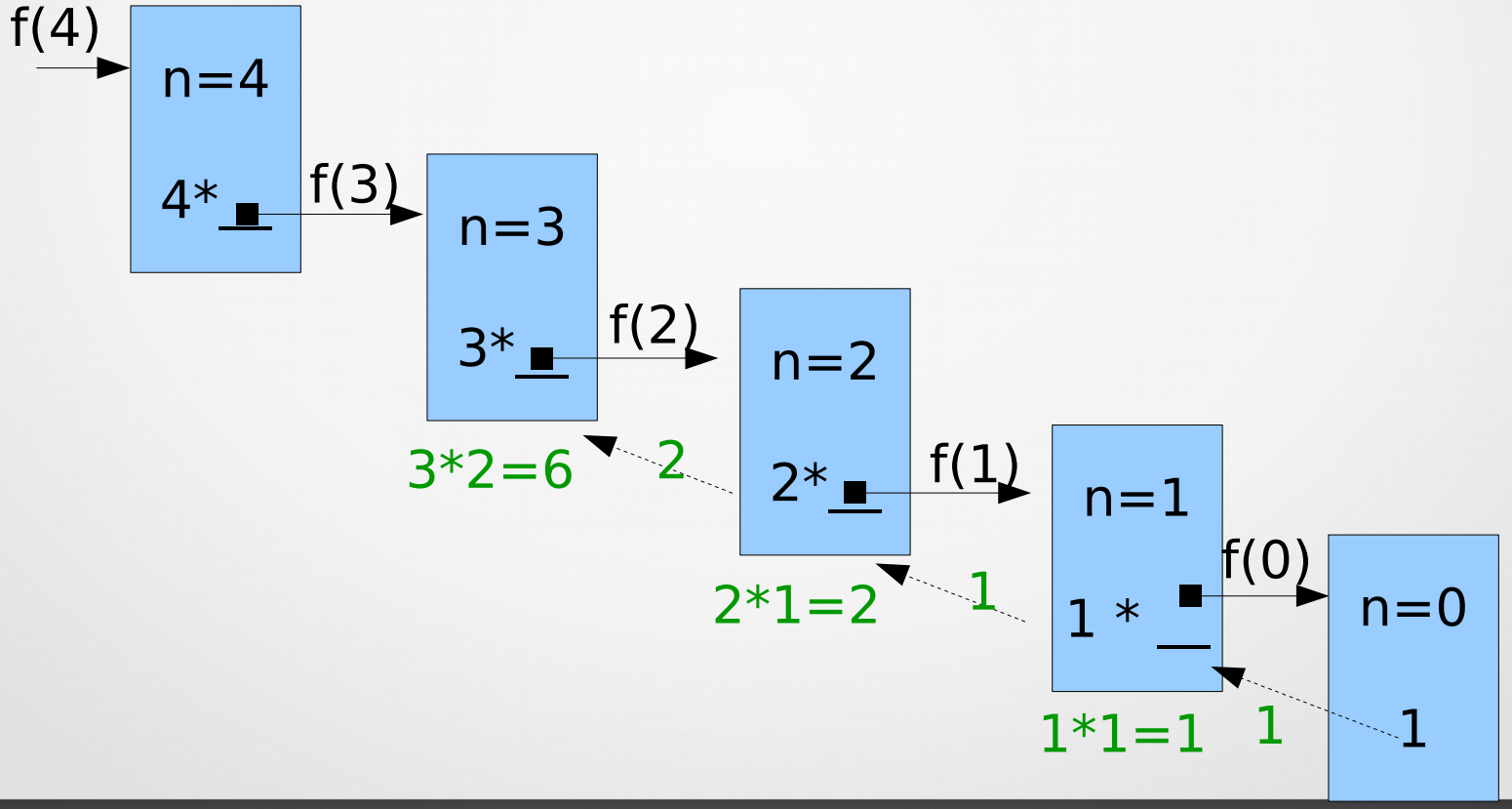
5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition

$f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

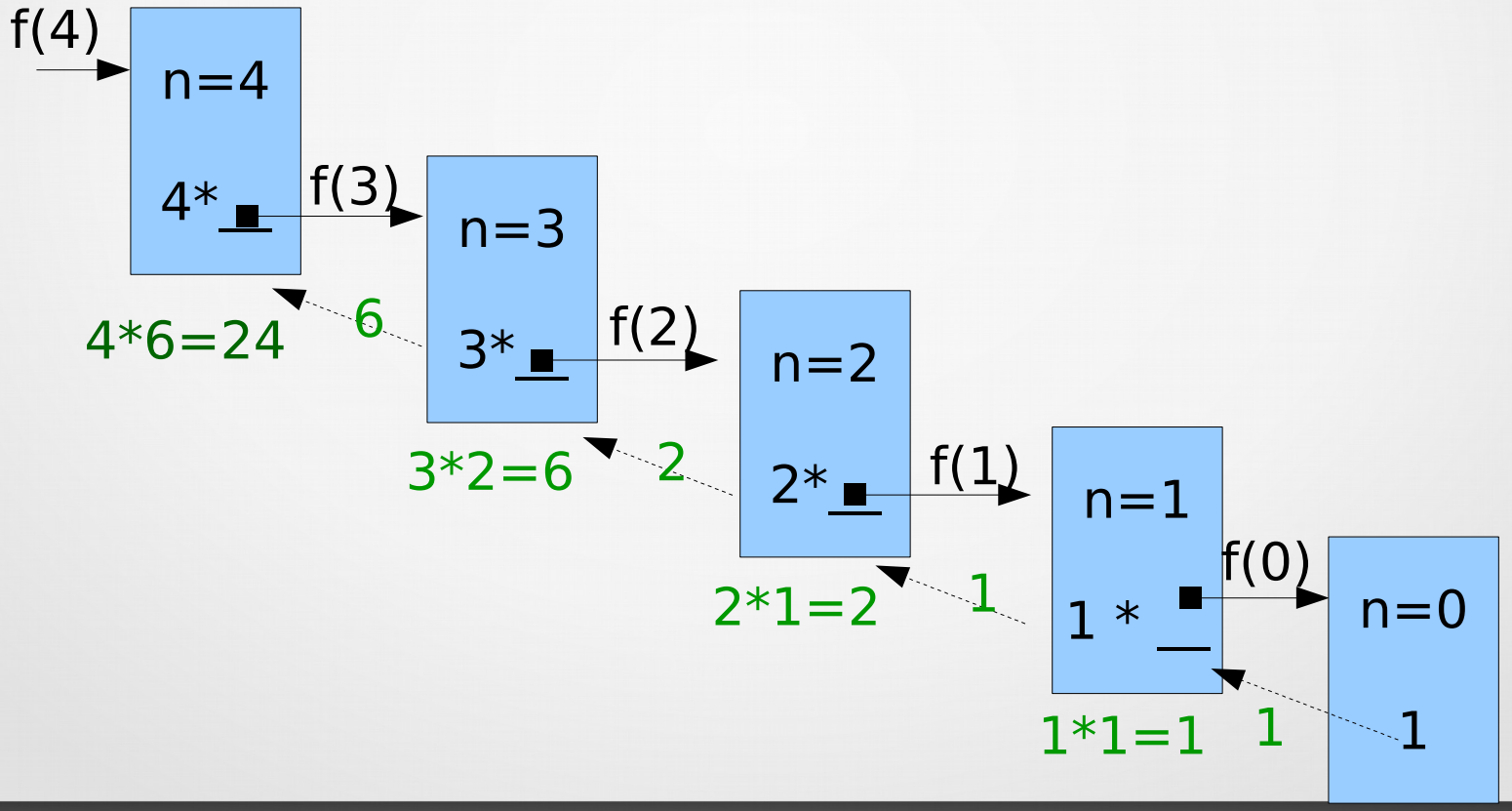


5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

Inductive definition $f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$

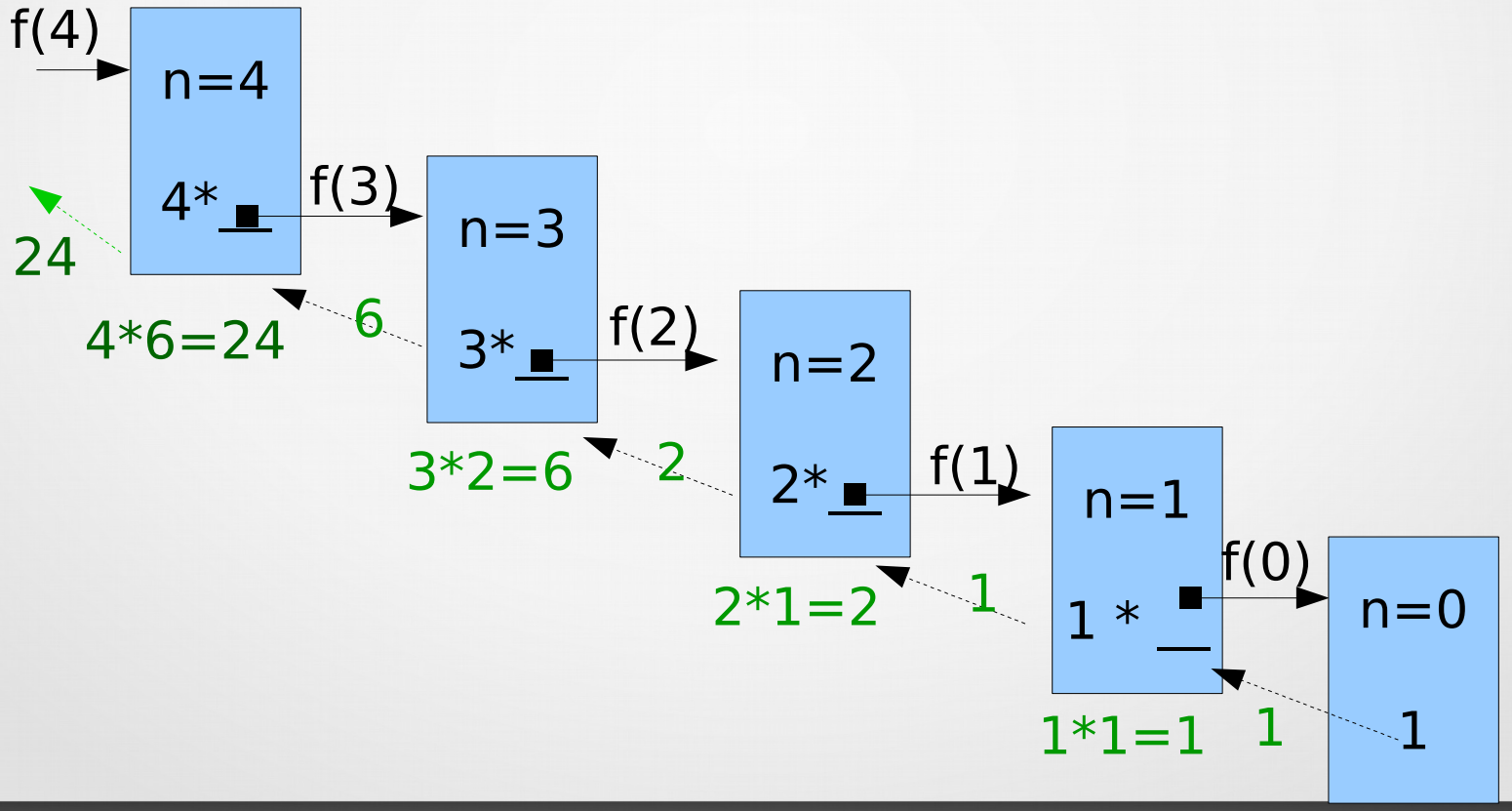


5.3 Recursive definitions and structural induction

Recursive definition of factorial function:

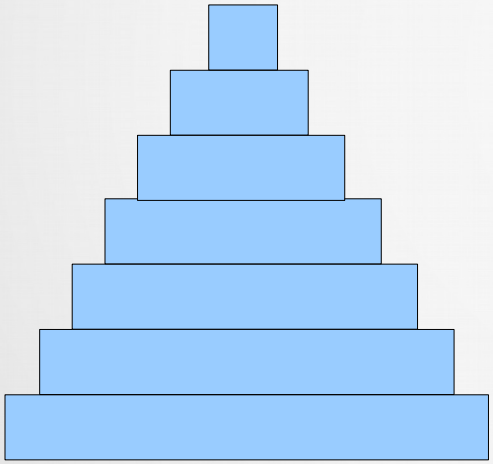
Inductive definition $f(0) = 1$

$f(n) = n\, f(n-1)$, for $n \geq 1$



5.3 Recursive definitions and structural induction

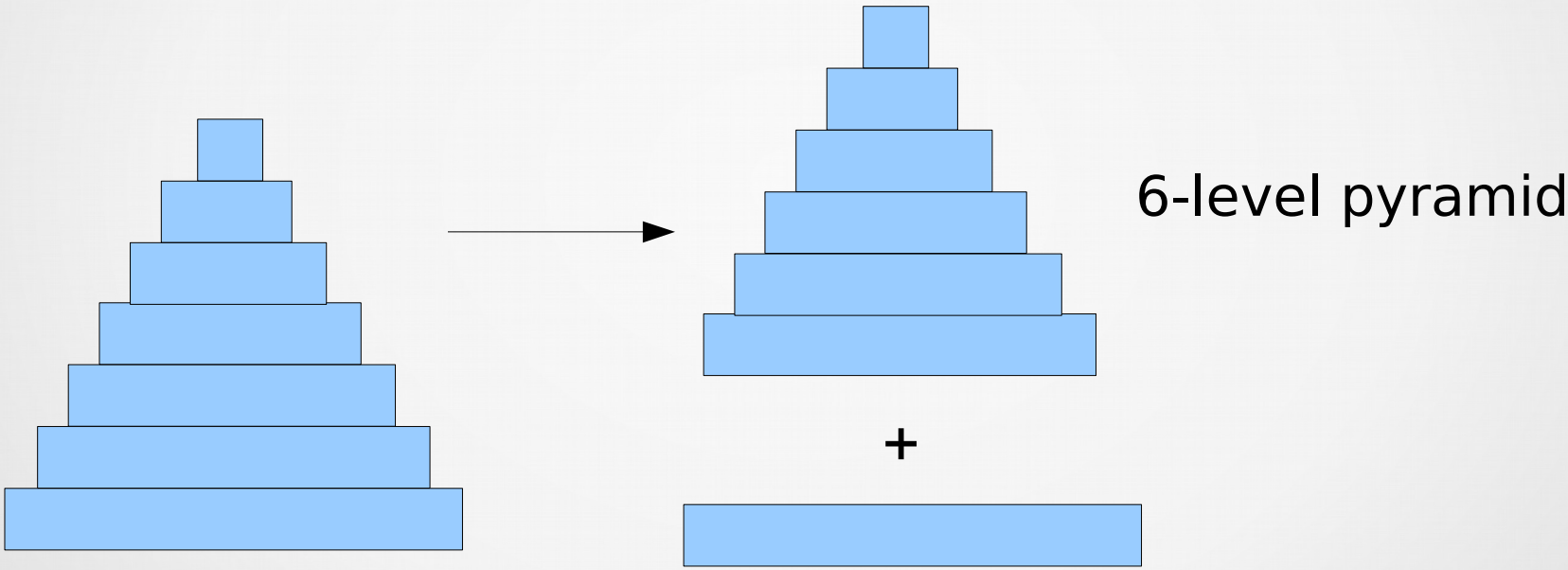
Structural recursion examples:



7-level pyramid

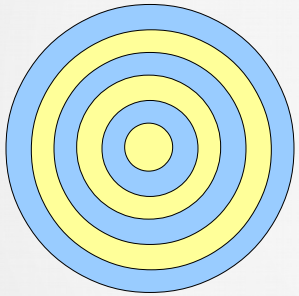
5.3 Recursive definitions and structural induction

Structural recursion examples:



5.3 Recursive definitions and structural induction

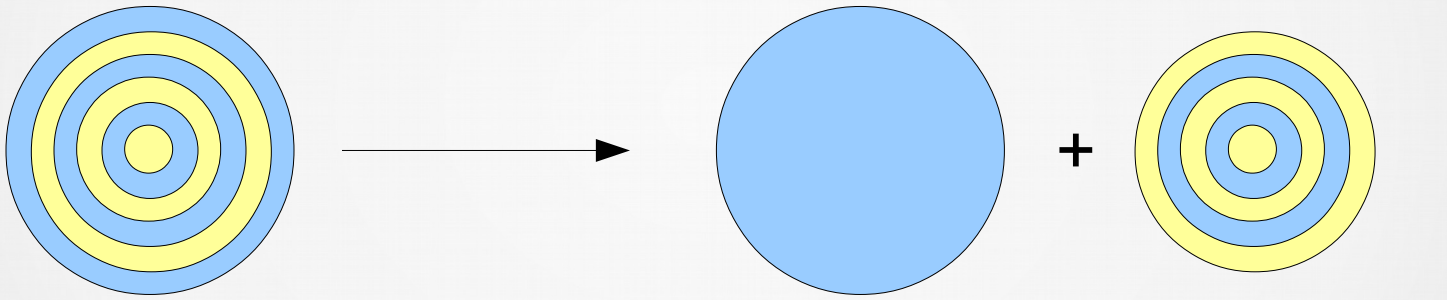
Structural recursion examples:



a 6-rings bullseye

5.3 Recursive definitions and structural induction

Structural recursion examples:



a 6-rings bullseye

1 ring

a 5-rings bullseye

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

domain: non-negative integers

Basis step: specify the value of the function at zero

Recursive step: give a rule for finding its value at an integer from its values at smaller integers

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

domain: non-negative integers

Basis step: specify the value of the function at zero

Recursive step: give a rule for finding its value at an integer from its values at smaller integers

Recursively defined functions are well defined, i.e. for every positive integer the value of a function at this integer is determined in an unambiguous way.

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

1) Give a recursive definition of x^n,
   where $x,n \in Z$ and $n \geq 0$, $x \neq 0$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

1) Give a recursive definition of x^n,
   where $x,n \in Z$ and $n \geq 0$, $x \neq 0$

Answer:

$x^0 = 1$

$x^{n+1} = x^n \cdot x$, for $n = 0, 1, 2, \ldots$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

2) Find the recursive definition that matches the non-recursive definition of the function $f(n)=\sum_{i=0}^{n} i^2$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

2) Find the recursive definition that matches the non-recursive definition of the function $f(n)=\sum_{i=0}^{n} i^2$

Solution: $f(n)=\sum_{i=0}^{n} i^2=0^2+1^2+3^2+\ldots+n^2$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

2) Find the recursive definition that matches the non-recursive definition of the function $f(n)=\sum_{i=0}^{n} i^2$

Solution: $f(n)=\sum_{i=0}^{n} i^2=0^2+1^2+3^2+\ldots+n^2$

when n = 0, $f(0)=\sum_{i=0}^{0} i^2=0^2=0$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

2) Find the recursive definition that matches the non-recursive definition of the function $f(n)=\sum_{i=0}^{n} i^2$

Solution: $f(n)=\sum_{i=0}^{n} i^2=0^2+1^2+3^2+\ldots+n^2$

when n = 0, $f(0)=\sum_{i=0}^{0} i^2=0^2=0$

for n+1: $f(n+1)=\sum_{i=0}^{n+1} i^2$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

2) Find the recursive definition that matches the non-recursive definition of the function $f(n)=\sum_{i=0}^{n} i^2$

Solution: $f(n)=\sum_{i=0}^{n} i^2=0^2+1^2+3^2+\ldots+n^2$

when n = 0, $f(0)=\sum_{i=0}^{0} i^2=0^2=0$

for n+1: $f(n+1)=\sum_{i=0}^{n+1} i^2=\sum_{i=0}^{n} i^2+(n+1)^2=f(n)+(n+1)^2$

5.3 Recursive definitions and structural induction

Recursively(Inductively) defined functions

Examples:

2) Find the recursive definition that matches the non-recursive definition of the function $f(n)=\sum_{i=0}^{n} i^2$

Solution: $f(n)=\sum_{i=0}^{n} i^2=0^2+1^2+3^2+\ldots+n^2$

when n = 0, $f(0)=\sum_{i=0}^{0} i^2=0^2=0$

for n+1: $f(n+1)=\sum_{i=0}^{n+1} i^2=\sum_{i=0}^{n} i^2+(n+1)^2=f(n)+(n+1)^2$

Answer:

f(0)=0

f(n+1) = f(n) + $(n+1)^2$, for n = 0, 1, 2, …

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Certain kinds of sets are most naturally specified with recursive definitions.

A recursive definition of a set shows how to construct elements in the set by putting together smaller elements.

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Certain kinds of sets are most naturally specified with recursive definitions.

A recursive definition of a set shows how to construct elements in the set by putting together smaller elements.

Example: consider a set of all positive odd integers

$S = \{1, 3, 5, 7, \ldots.\}$

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Certain kinds of sets are most naturally specified with recursive definitions.

A recursive definition of a set shows how to construct elements in the set by putting together smaller elements.

Example: consider a set of all positive odd integers

$$S = \{1, 3, 5, 7, \ldots.\}$$

Recursive definition:

$1 \in S$

if $x \in S$ then $x+2 \in S$ and $x-2 \in S$

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Certain kinds of sets are most naturally specified with recursive definitions.

A recursive definition of a set shows how to construct elements in the set by putting together smaller elements.

Example: consider a set of all positive odd integers

$$S = \{1, 3, 5, 7, \ldots.\}$$

Recursive definition:

$1 \in S$

if $x \in S$ then $x+2 \in S$ and ~~$x-2 \in S$~~

5.3 Recursive definitions and structural induction

Components of a recursive definition of a set:

- a basis explicitly states that one or more specific elements are in the set
- a recursive rule shows how to construct larger elements in the set from elements already known to be in the set.
  (can be more than one recursive rule)
- an exclusion statement states that an element is in the set only if it is given in the basis or can be constructed by applying the recursive rules repeatedly to elements given in the basis

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: consider a set of all positive odd integers

$$S = \{1, 3, 5, 7, \ldots.\}$$

Recursive definition:

$1 \in S$

if $x \in S$ then $x+2 \in S$

Exclusion statement: $k \in S$ only if it is given in the basis or can be calculated by applying the recursive rule to elements of set S.

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Recursive definitions play an important role in the *study of strings*.

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Recursive definitions play an important role in the *study of strings*.

Example: assume that $\sum$ is a finite/infinite sequence of symbols, called alphabet

a string over an alphabet $\sum$ is a finite sequence of symbols from $\sum$

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Recursive definitions play an important role in the *study of strings*.

Example: assume that $\sum$ is a finite/infinite sequence of symbols, called alphabet

a string over an alphabet $\sum$ is a finite sequence of symbols from $\sum$

The set $\sum^*$ of strings over the alphabet $\sum$ is defined recursively by:

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Recursive definitions play an important role in the *study of strings*.

Example: assume that $\sum$ is a finite/infinite sequence of symbols, called alphabet

a string over an alphabet $\sum$ is a finite sequence of symbols from $\sum$

The set $\sum^*$ of strings over the alphabet $\sum$ is defined recursively by:

lambda

Basis step: $\lambda \in \sum^*$ (where λ is the empty string, i.e.no symbols)

Recursive step: if $w \in \sum^*$ and $x \in \sum^*$, then $wx \in \sum^*$

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: let alphabet $\sum = \{0,1\}$

Basis step: $\lambda \in \sum^*$

Recursive step: if $x \in \sum^*$ then $x0 \in \sum^*$ and $x1 \in \sum^*$

Bit-strings

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: let alphabet $\sum = \{0,1\}$

Basis step: $\lambda \in \sum^*$

Recursive step: if $x \in \sum^*$ then $x0 \in \sum^*$ and $x1 \in \sum^*$

$10101 \in \sum^*$

$101 \in \sum^*$

$1 \in \sum^*$

Bit-strings

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: let alphabet $\sum = \{0,1\}$

Basis step: $\lambda \in \sum^*$
Recursive step: if $x \in \sum^*$ then $x0 \in \sum^*$ and $x1 \in \sum^*$

$10101 \in \sum^*$
$101 \in \sum^*$
$1 \in \sum^*$

How about 001? Does $001 \in \sum^*$?

Bit-strings

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: properly nested parentheses

Basis: The sequence () is properly nested.
Recursive rules: If u and v are properly-nested sequences of parentheses then:

- (u) is properly nested.
- uv is properly nested.

Exclusion statement: a string is properly nested only if it is given in the basis or can be constructed by applying the recursive rules to strings in the basis.

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: properly nested parentheses

Basis: The sequence () is properly nested.
Recursive rules: If u and v are properly-nested sequences of parentheses then:

- (u) is properly nested.
- uv is properly nested.

Exclusion statement: a string is properly nested only if it is given in the basis or can be constructed by applying the recursive rules to strings in the basis.
(()())

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: properly nested parentheses

Basis: The sequence () is properly nested.
Recursive rules: If u and v are properly-nested sequences of parentheses then:

- (u) is properly nested.
- uv is properly nested.

Exclusion statement: a string is properly nested only if it is given in the basis or can be constructed by applying the recursive rules to strings in the basis.

(()())
((((())())())()())

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: strings concatenation “abc” · “bbc” = “abcbbc”

Let's define concatenation of strings · recursively:

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: strings concatenation “abc” · “bbc” = “abcbbc”
Let's define concatenation of strings · recursively:

Basis step: if $w \in Z^*$, then $w \cdot \lambda = w$, where λ is empty string
Inductive step: if $w_1 \in Z^*$, $w_2 \in Z^*$, and $x \in Z^*$,

then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: strings concatenation “abc” · “bbc” = “abcbbc”
Let's define concatenation of strings · recursively:

Basis step: if $w \in Z^*$, then $w \cdot \lambda = w$, where λ is empty string
Inductive step: if $w_1 \in Z^*$, $w_2 \in Z^*$, and $x \in Z^*$,

then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$

Very often instead of $w_1 \cdot w_2$ we simply write $w_1 w_2$

Example:
If w_1=*lala* and w_2=*land* then $w_1 w_2$ = *lalaland*

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: well-formed formulae in Propositional Logic

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: well-formed formulae in Propositional Logic

Basic step: **T**, **F**, and s (propositional variable) are well-formed formulae

Recursive step: if E and F are well-formed formulae, then $(\neg E)$,$(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$ and $(E \leftrightarrow F)$ are well-formed formulae.

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Example: well-formed formulae in Propositional Logic

Basic step: **T**, **F**, and s (propositional variable) are well-formed formulae

Recursive step: if E and F are well-formed formulae, then $(\neg E)$,$(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$ and $(E \leftrightarrow F)$ are well-formed formulae.

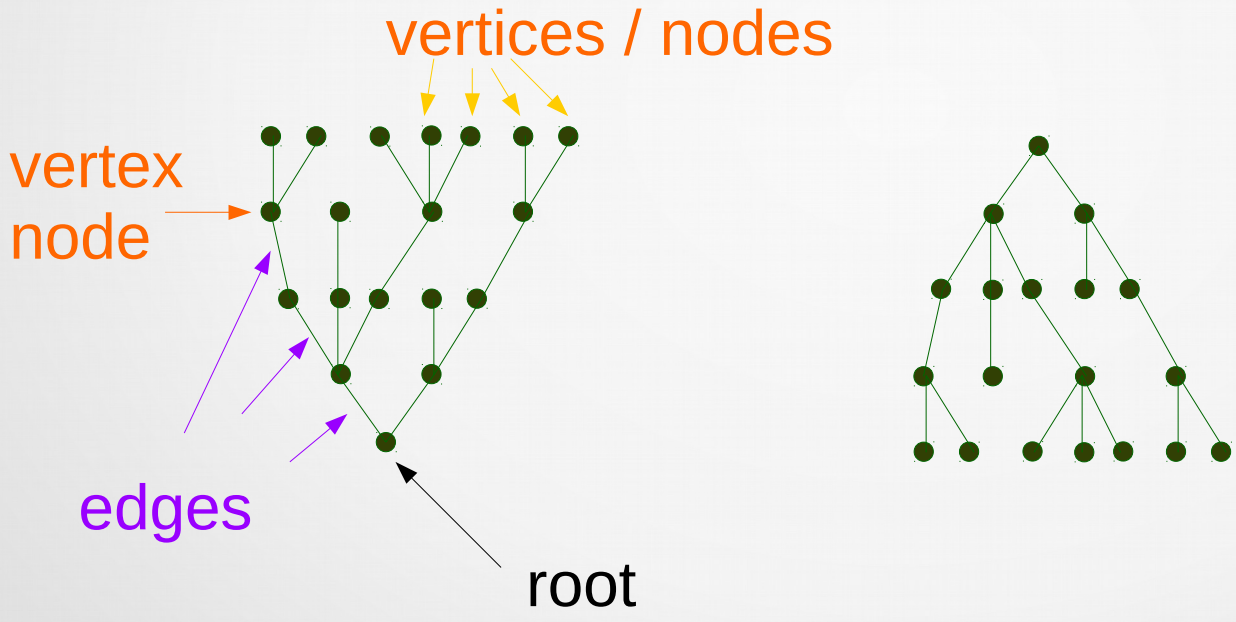
Example: $p, q \rightarrow (p \wedge q) \rightarrow \neg(p \wedge q) \rightarrow (\neg(p \wedge q)) \leftrightarrow q)$

5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Trees (rooted trees)

A trees consist of a root, vertices and edges



5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Trees (rooted trees)

[Def] A rooted tree consist of a set of vertices with a distinguished vertex called the *root*, and edges connecting there vertices, and can be defined recursively:

Basis step: a single vertex r is a *rooted tree*

Inductive step: assume $T_1, T_2, \ldots, T_n$ are disjoint rooted trees with roots $r_1, r_2, \ldots, r_n$, respectively. Then the graph formed by adding an edge from each tree to a vertex r (which is not in any of the trees $T_1, \ldots, T_n$). Vertex r is the *root* of the newly created *rooted tree*.

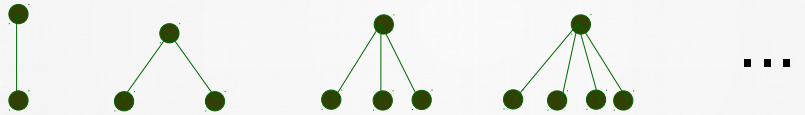
5.3 Recursive definitions and structural induction

Recursively defined sets and structures

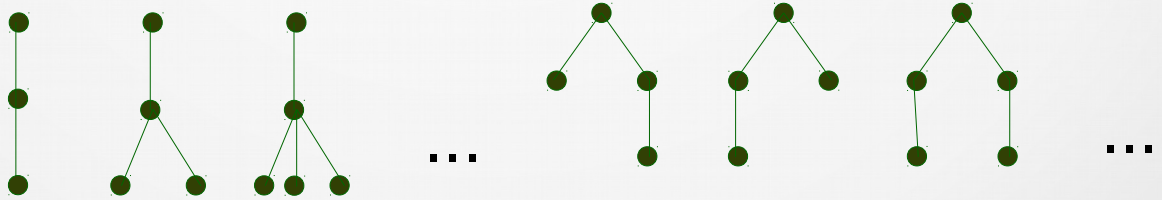
Trees (rooted trees)

Basis step:

Step 1:



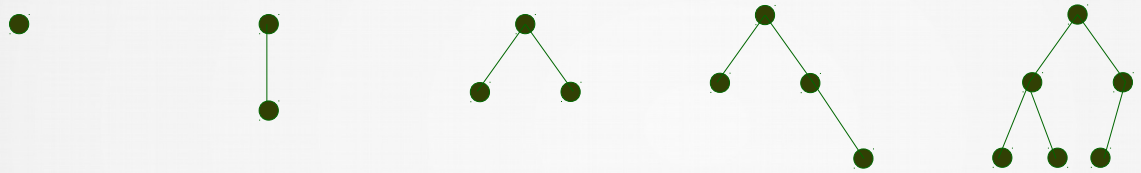
Step 2:

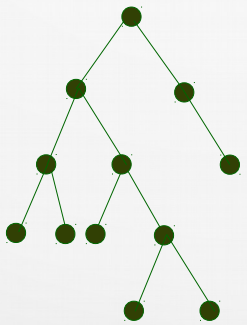


5.3 Recursive definitions and structural induction

Recursively defined sets and structures

Binary Trees





5.3 Recursive definitions and structural induction

Structural Induction

- is a type of induction used to prove theorems about recursively defined sets

- follows the structure of the recursive definition.

5.3 Recursive definitions and structural induction

Structural Induction

Example: Show that the set S defined below is a set of positive integers.

Basis step: $1 \in S$, (1)

Recursive step: $s+t \in S$, if $s,t \in S$ (2)

5.3 Recursive definitions and structural induction

Structural Induction

Example: Show that the set S defined below is a set of positive integers.

Basis step: $1 \in S$, (1)

Recursive step: $s+t \in S$, if $s,t \in S$ (2)

Proof (by structural induction):

Base case: the element 1 from the basis step is a positive integer

- show that the result holds for all elements specified in the basis step of the rec. definition to be in the set

5.3 Recursive definitions and structural induction

Structural Induction

Example: Show that the set S defined below is a set of positive integers.

Basis step: $1 \in S$, (1)

Recursive step: $s+t \in S$, if $s,t \in S$ (2)

Proof (by structural induction):

Base case: the element 1 from the basis step is a positive integer

Inductive/recursive step: assume that two positive integers $k,t \in S$ were constructed by applying a sequence of recursive steps starting with number 1.
Obviously a sum of two positive integers (k+t) is a positive integer, in addition by (2) $(k+t) \in S$ q.e.d.