

Software Architecture

Abstract Data Types

Mathematical description

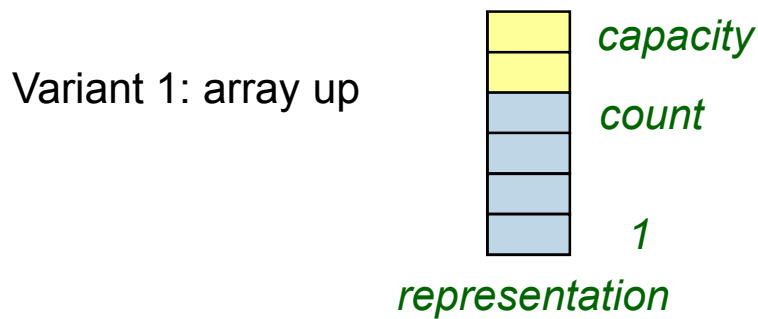
- An ADT is a mathematical specification
- Describes the properties and the behavior of instances of this type
- Doesn't describe implementation details (therefore its *abstract*)
- An example: STACK

Stack

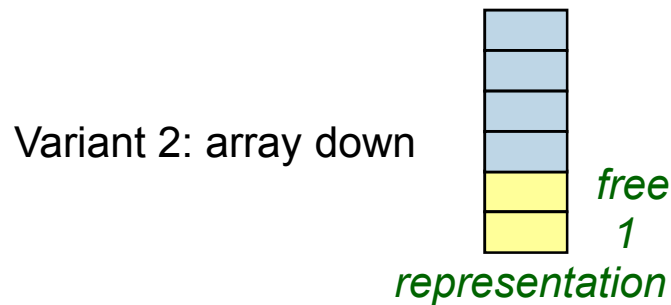
- LIFO Queue (last in, first-out)
- Operations:
 - put: Put something onto the STACK (Command)
 - remove: Remove the top element of the STACK (Command)
 - item: Return the value of the top item (Query)
 - empty: Is the STACK empty? (Query)

Abstract data types

One data type – many implementations

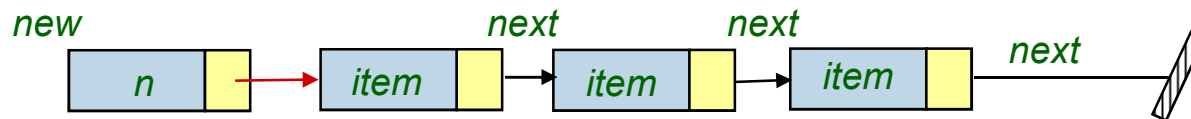


“Push” x on stack *representation*:
 $representation[*count*] := x$
 $count := count + 1$



“Push” x on stack *representation*:
 $representation[*free*] := x$
 $free := free - 1$

Variant 3: linked list



“Push” operation:
 $new(n)$
 $n.item := x$
 $n.next := first$
 $first := new$

Formal description of a STACK

Types:

The type(s) that are described by the ADT.

Functions:

The functions that can be applied to the ADT.

Preconditions:

Preconditions that need to be fulfilled to apply a feature.

Axioms:

Axioms that the ADT fulfills.

Abstract data types

Creators:

Create a new instance of an ADT.

{OTHERS} → ADT

Queries:

Functions that have a return value and do not change the instance.

ADT {x OTHERS} → OTHERS

Commands:

Functions without return value that change the instance.

ADT {x OTHERS} → ADT

Partial function ↗

There are cases where no valid return value can be given for a function
e.g.. division by 0

Abstract data types

Types

STACK [G]

-- G: Formal generic parameter

Functions

put: STACK [G] \times G \rightarrow STACK [G]

remove: STACK [G] \rightarrow STACK [G]

item: STACK [G] \rightarrow G

empty: STACK [G] \rightarrow BOOLEAN

new: STACK [G]

Abstract data types

Preconditions

remove (s: STACK [G]) **require** not empty (s)

item (s: STACK [G]) **require** not empty (s)

Axioms For all x: G, s: STACK [G]

1. item (put (s, x)) = x

2. remove (put (s, x)) = s

3. empty (new)

4. not empty (put (s, x))

Well-formed and correct terms

Well-formed: all functions get a right number of arguments of right types

Correct: preconditions of all functions are satisfied

empty (item (put (new, 3))) **ill-formed**

item (put (new, 3)) **3**

item (remove (put (new, 3))) **incorrect**

empty (remove (put (new, 7))) **True**

item (put (put (remove (put (new, 4)), 3), 2)) **2**

Structural induction

Goal: Prove that a property P is valid for all correct terms T of the ADT.

Induction basis (step 0):

Prove that P holds for all creators of the ADT.

Induction hypothesis (step $n-1$):

Assume that P holds for any correct term T_{sub} .

Induction step (step n):

Prove for all commands that can be applied to T_{sub} that P will still hold afterwards.

Sufficient completeness

An ADT is sufficiently complete if and only if:

For every term you can determine whether it is **correct** or not using the axioms of the ADT.

Every correct term where **the outermost function is a query** of the ADT can be reduced, using the axioms of the ADT, into a **term not using any function of the ADT**.

Important for the assignment

- For every **partial function** you need to specify a **precondition**.
- Don't forget function **new**!
- Because we distinguish between **withdraw** and **deposit**, you need to make sure that **no negative** figures can be deposited or withdrawn.
- Task 3: add a new type...