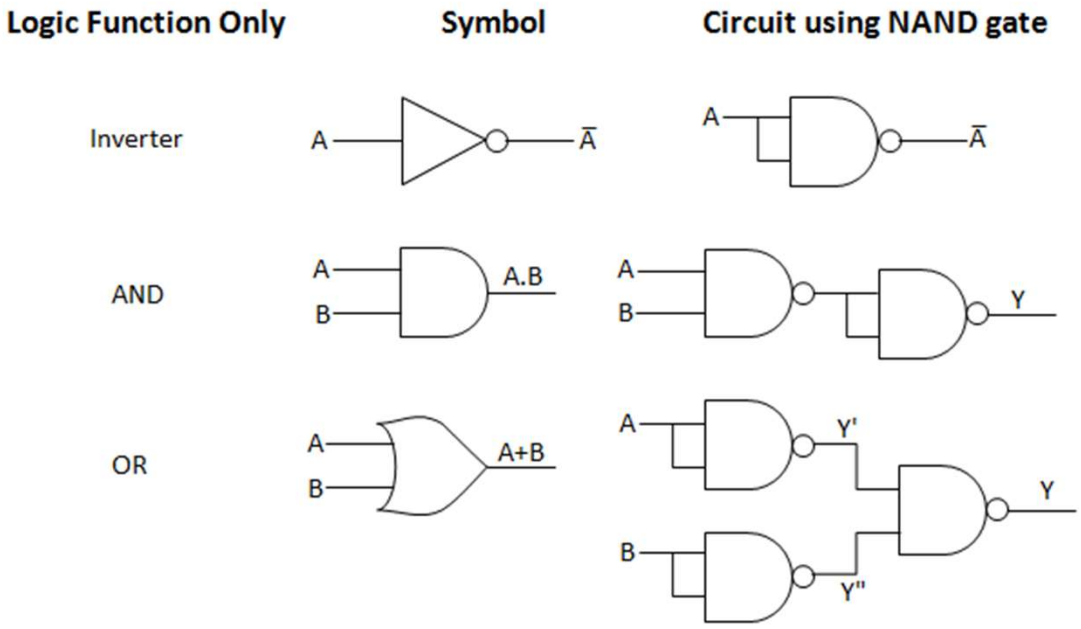
CSE216 Programming Abstractions

Modules and Functors

YoungMin Kwon

Modules and Functors

- We will practice modules and functors
- First: define signatures for IBoolNand and IBool
  - All Boolean operators can be implemented using Nand



Signatures: IBoolNand and IBool

```
(* IBoolNand: signature for nand operator
    - nand is a universal operator
*)
module type IBoolNand= sig
    type t
    val _true:  t
    val _false: t
    val _nand:  t -> t -> t
    val to_str: t -> string
end

(* IBool: signature for boolean operators
    TODO: - include IBoolNand
          - add operators _not, _and, _or, _nor, _imply, _equiv
*)
module type IBool = sig

end
```

Functor: BoolBuilder

- Second: implement BoolBuilder functor
  - Takes IBoolNand type module
  - Returns a module of signature IBool

```
(* BoolBuilder: functor for boolean operators
    TODO: - copy _true, _false, _nand, and to_str from module B
          - implement operators _not, _and, _or, _nor, _imply, _equiv
          - _imply: x -> y == !x \/ y,
            _equiv: x <-> y == (x -> y) /\ (y -> x)
*)
module BoolBuilder (B: IBoolNand): IBool = struct
    type t = B.t


end
```

Bool1 Module: using BoolBuilder

- Third: implement Bool1 module using BoolBuilder

```
(* Bool1: first implementation of IBool
    TODO: - implement _nand
*)
module Bool1 = BoolBuilder (struct
    type t = bool
    let _true     = true
    let _false    = false
    let _nand x y =
    let to_str x  = if x then "true" else "false"
end)
```

Functor: BoolTest

- Fourth: using BoolTest functor and Bool1 module make a tester for Bool1

```
(* BoolTest: functor for test IBool
*)
module BoolTest (B: IBool) = struct
    let test () =
        let open B in
        let t = _true  in
        let f = _false in
…
        assert ("false" = (_and t f   |> to_str));
        assert ("true"  = (_not f     |> to_str));
…
end

(* BT1: first implementation of IBool tester
    TODO: using BoolTest, build a tester for Bool1 and run the test
*)
module BT1 =
let _ = BT1.
```

Bool2 Module (Church Boolean)

- Fifth: implement Bool2 module using BoolBuilder

```
(* Bool2: second implementation of IBool
    TODO: - implement _nand
*)
module Bool2 = BoolBuilder (struct
    type t = B of (t -> t -> t) | S of string
    let dropB = function B x -> x | _ -> assert false
    let dropS = function S x -> x | _ -> assert false

    let _true     = B (fun x y -> x)
    let _false    = B (fun x y -> y)
    let _nand x y =
    let to_str x  = (dropB x) (S "true") (S "false") |> dropS
end)

(* BT2: second implementation of IBool tester
    TODO: using BoolTest, build a tester for Bool2 and run the test
*)
module BT2 =
let _ = BT2.
```