Predicates (outputs true/false)

Function symbols (outputs a domain from the range e.g. numbers)

A constant is a function with arity zero





Predicate symbols: P, O, R, C, H

Terms: X, y





Predicates: = (arity 2)

Terms: x, y

Function symbols: + (arity 2)



Predicates: prime (arity 1), ∈ (arity 2)

Terms: n, x

Function symbols: ㅡ (arity 2), > (arity 2), ℕ (arity 0)

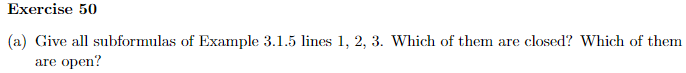


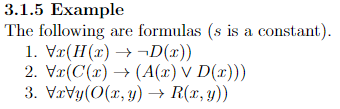
Predicates: B (arity 1), A (arity 0) , R (arity 1 & 2) 

Function symbols: f (arity 1), g (arity 2), σ (arity 2), (arity 1), ε (arity 1), a (arity 0),

bc (single constant, arity 0)

Terms: x, y, z





Note: Every formula is a subformula of itself

Closed: Original formula 1

Open (contians free variables): ㄱD(x), H(x), D(x), H(x)→ㄱD(x)

Closed: Original formula 2

Open (contains free vaiables): C(x), A(x), D(x), A(x) ⋁ D(x), C(x) → (A(x) ⋁ D(x))

Closed: Original formula 3

Open (contains free vaiables): O(x,y), R( x,y ), O(x,y) → R( x,y )





UA = ℕ

RA = { (w, v, w) | w < v }00

BA = { (3, 3), (12, 12)}



Random junk:

A model is a way to assign true or false to a formula

A formula is a logical consequence of another formula

K-ary means how many arguments a predicate or function symbol has.

Predicates cannot appear inside another predicate. But functions can appear inside funtions. Funtions casn also appear inside predicates.

Constants genereally appear at the beginning of the alphabet. Variables near the end.

3.2.1 definition

A (pronounced “script A”) is a strucure. There are always 2 things to a structure. UAand IA.

UA is some set that is not empy. It is a ground set / universe.

IA is a function that maps:

* K-ary Predicate Symbols to a k-ary relationship with elements in UA (if IA is defined for that predicate symbol)
* K-ary Function Symbols to a k-ary function on elements in UA (if IA is defined for the function symbols)
  + Contstants are mapped to a single element in UA
* Variables to elements in UA (If IA is defined for that variable)

IA(P) = PA

A is suitable for a fomula if it is defined for all predicates, functions, and free variables. Bound variables are ignored in suitability.

In the example:

For the constants, just pick something in UA.

* At this stage it does not matter what we pick.

For the functions, create some sort of mapping like in this example:

* sA : n ↦ n + 2

For the predicates:

* For unary predicate symbols, just give some subset of the universe.
* For binary predicate symbols, just give some set of pairs under the universe. One way is to define with relations. For example, the set of all pairs (n, k), where n is less than k. { (n, k) | n < k }