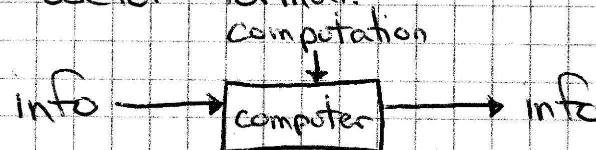
Formal Methods for Computer Science I

Computational thinking is a way of solving problems, designing systems, and understanding human behaviour that draws on concepts fundamental to computer science.
(CS includes programming)

Computer = programmable machine that receives input, stores and manipulates information, and provides output in a useful format.



CS = systematic study of information and computation.

Boolean Algebra

$\wedge$ and conjunction	$\vee$ or disjunction	$\neg$ not negation	⇒ complete operator system
* &	+ |	− !	(alternative forms)

neutral elements, zero elements, idempotence, commutativity, associativity, distributivity, negation

De Morgan's Laws

$\neg(a \wedge b) = (\neg a) \vee (\neg b)$ NAND

$\neg(a \vee b) = (\neg a) \wedge (\neg b)$ NOR

exclusive or (XOR), implication $(\neg a \vee b)$, equivalence $(a == b)$

Tautology (always true), Contradiction (always false), Satisfiable

Predicate Logic

propositional logic : True, False, propositions

quantifier, universe, predicate $\Uparrow$

$\forall$ universal quantifier (for all)
$\exists$ existential quantifier (exists)

$\forall$ distributive with $\wedge$, not with $\vee$
$\exists$ distributive with $\vee$, not with $\wedge$

De Morgan's Axioms

$\neg\exists: \neg(\exists x : U(x) : F(x)) \Leftrightarrow \forall x : U(x) : \neg F(x)$
$\neg\forall: \neg(\forall x : U(x) : F(x)) \Leftrightarrow \exists x : U(x) : \neg F(x)$

Well Formulated Formulas (WFF)

bounded variables: specific value or quantified
free variables: not bound, not specified

You can't just swap $\forall$ and $\exists$ if they're mixed!
If P doesn't contain x as a variable, P can be outsic

$Min(x) : U(x) : T(x)$ smallest value of terms $T(x)$
$Max(x) : U(x) : T(x)$ greatest value of terms $T(x)$
$Anz(x) : U(x) : F(x)$ number/count of all objects for which $F(x)$

Logic reloaded

Logic = the study of the principles of correct reasoning

Sets

set = a group of objects $\{0, 12, 32\}$

$\emptyset$ empty set $\qquad U$ universe

$\in$ membership $\qquad \notin$ non-membership

$A \subseteq B$ subset
$A \subset B$ proper subset
$A \cup B$ union $(x \in A \vee x \in B)$
$A \cap B$ intersection $(x \in A \wedge x \in B)$
$A \setminus B, A'$ complements $(x \in A \wedge x \notin B)$

Proprieties like Boolean Algebra.

$A \subseteq A$ reflexivity
$A \subseteq B \wedge B \subseteq A \Leftrightarrow A = B$ anti-symmetry
$A \subseteq B \wedge B \subseteq C \Leftrightarrow A \subseteq C$ transitivity

$A \times B$ cartesian product (all combinations)

N-ary relation, binary relation

Relations

N-ary relation $R \subseteq A_1 \times A_2 \times \ldots \times A_n$

Binary relation $R \subseteq A_1 \times A_2$

$(a,b) \in R$ $\quad aRb$

domain (R) all a that satisfy R with a b

range (R) all b that satisfy R with an a

- reflexive relation $\forall x : x \in A : xRx$

  every element x of A is in relation R with itself

- symmetric relation $\forall x,y : x,y \in A : xRy \Rightarrow yRx$

  if there is a relation between x and y, then there is also a relation between y and x

- transitive relation $\forall x,y,z : x,y,z \in A : (xRy \land yRz) \Rightarrow xR$

Equivalence class $[x]_R = \{y \mid xRy\}$ f.ex. $[1]_= = \{1\}$

Transitive closure : relation R^+ that contains all possible transitive relations over all elements

- irreflexive relation $\forall x : x \in A : \neg(xRx)$

  no element x of A is in relation R with itself

- antisymmetric relation $\forall x,y : x,y \in A : (xRy \land yRx) \Rightarrow x=y$

  if there is a relation between x and y and one between y and x, then x equals y

- asymmetric relation $\forall x,y : x,y \in A : xRy \Rightarrow \neg(yRx)$

  xRy and yRx cannot hold at the same time

- non-symmetric relation $\forall x,y: x,y \in A: (xRy) \wedge \neg(yRx)$
  a relation that is not symmetric
- total relation $\forall x,y: x,y \in A: xRy \vee yRx$
  R is defined on the entire A
- acyclic relation
  there are no elements with transitive closure to thems

Partial order: reflexive, transitive, antisymmetric
Total order: partial order, total relation
Strict partial order: reflexive, transitive

Trees

nodes, edges; root, leaf; parent, child; subtree, path
left, right
depth = level $\Rightarrow$ distance from root
height $\Rightarrow$ distance to leaf
degree $\Rightarrow$ number of children of a node

ordered tree: leftmost, rightmost
isomorphic trees (same structure)

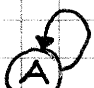
binary tree (all nodes have a left and/or right child)
complete binary tree (all nodes have a left AND a right child)

Breadth-first Traversal (BFS) $\Rightarrow$ Level-order

Preorder (visit, left, right)
Inorder (left, visit, right)
Postorder (left, right, visit)

examples: hierarchical structure
binary search tree (⚠ balance)
mathematical expressions (operators, operands)

Graphs

Graps are made of Vertices and Edges
$G = (V, E)$ $E = \{ \{u, v\} \mid u, v \in V \}$
Representation as Adiacency matrix
- directed VS undirected graphs
- degree of a node = number of connections
- weighted graphs = edges have a "value" (weight)
- complete VS not complete graph
  (all possible connections) (not all possible connections)
- Bipartite graph (vertices can be divided in two disjoint sets U and V such that every edge connects a vertex in U to one in V)
  → has no odd-length cycles
- path
- cycle (returns to starting point)
- loop (returns to itself, as in )
- eulerian path (visits every edge exactly once)
- hamiltonian path (visits every vertex exactly once)
  } cycle if return to start
- spanning tree (tree without cycles)
- components
- critical node
- critical edge
  (grap gets separated/divided if this is eliminated)
- biconnected components/graph (removal of a vertex doesn't break the graph)
  (→ multiple ways from V_1 to V_2)
- subgraph (connected nodes and all related, connecting edges)

- weakly reachable (exists undirected path)
- strongly reachable (exists directed path)

- Diskstra algorithm
- Floyd-Warshall algorithm
- Traveling salesman problem

Complexity

Computation requires resources: memory, bandwidth, time, ..

Big-O Notation (worst-case) $O(n)$
Ω Notation (best-case) $\Omega(n)$
Θ Notation (average-case) $\Theta(n)$

$O(1)$ constant, $O(\log(n))$ logarithmic, $O(n)$ linear,
$O(n^2)$ quadratic, $O(n^3)$ cubic, $O(n^k)$ polynomial, $O(k^n)$ expone

- ignore constants $O(c*f) = O(f)$
- take the maximum term $O(f)+O(g) = \max(O(f), O(g))$
- multiplicate fully $O(f*g) = O(f) * O(g)$

solvable in $O(n^k)$ $P \subset NP$ verifiable in $O(n^k)$
but $\overset{?}{\neq}$

NP-complete problems: NP, and proven one cannot do bette
fex. Hamiltonian path, Traveling salesman, Graph coloring, Subs

Halting problem: will this program terminate? $\Rightarrow$ undecidal

Program verification and testing

testing: run the program with a set of inputs and check the output for defects (meaningful input)

verification: formally prove that the program has no defect

precondition $\{P\}$, postcondition $\{Q\}$

- Partial correctness $\{P\}\ S\ \{Q\}$
- Total correctness $[P]\ S\ [Q]$
- Skip $\{Q\}$ Skip $\{Q\}$
- Abort $\{P\}$ Abort $\{False\}$
- Assignment $\{Q[x/E]\}\ x := E\ \{Q\}$
- Sequence $\{P\}\ S_1\ \{Q\},\ \{Q\}\ S_2\ \{R\}$
- Conditional $\{P \land B\}\ S_1\ \{Q\},\ \{P \land \neg B\}\ S_2\ \{Q\}$
- While loop $\Rightarrow$ Loop invariant (true before and after every loop
- Weakest precondition $wp(S,Q)$

  $\forall \{P\}\ S\ \{Q\}\ ::\ P\ \ wp(S,Q)$

$\Rightarrow$ Verification of $\{P\}\ S\ \{Q\}$

1. Compute $wp(S,Q)$
2. Prove $P\ \ wp(S,Q)$

- Assignment $wp(x := A, Q) = Q_{x \leftarrow A}$
- Array Assignment $wp(a[x] := A, Q) = Q_{a \leftarrow a'}$
- Sequence $wp(S_1; S_2, Q) = wp(S_1, wp(S_2, Q))$
- Conditional $(B \Rightarrow wp(S_1,Q)) \land (\neg B \Rightarrow wp(S_2,Q))$
- While loop $wp(L,Q) = I \land ((I \land B) \Rightarrow wp(S,I))$

  $\land ((I \land \neg B) \Rightarrow Q)$

Models and languages

A model is a simplification of the subject, and its purpo
is to answer some particular questions aimed at the subj

Meta-Model = a model that makes statements about what can be expressed in valid models

⇒ you need to know the language!

- BNF (Backus-Naur-Form) ⇒ notation used to describe the syntax of languages used in computing
- EBNF (Extended BNF)

syntactically correct sentences do not necessarily have a valid meaning!

Programming Languages

A language is a set of sequences of symbols that we interpret to attribute meaning.

programming language ⇒ communicating software designs

programming = modeling

statements, expressions, variables, literals, control constructs, functions, comments,

- Imperative (data + algorithms)
- Object-Oriented (objects + messages)
- Functional (stateless + pure functions)
- Logic (Facts + rules)