Context and Hierarchy

Essential Concepts of the Course:

Complementarity:
the intimate relationship between data structure, algorithm and computational architecture.
Abstraction hierarchies:
from conceptualization through mathematics to implementation.
Programming paradigms:
the languages of design, modeling and implementation
Implementation hierarchies:
trading off between design and implementation efficiencies

Abstraction Hierarchy

conceptualization	(real world specific)
mathematical model	(symbolic)
implementation model	(software specific)
process model	(hardware specific)

Implementation Hierarchy

conceptualization/design (quasi)-language
very-high-level (task specific) programming tool
high-level programming language
low-level programming language
opcodes and machine language
high-level synthesis
low-level synthesis

Naming Domains

data types
constants/grounds
operators (functions, predicates)
program execution types (memory location, signal transitions)
resources (memory, operator circuits, i/o devices)
constraints (equations)

Data Structures

bit	array (eg byte, word)
string	queue
stream	linked list
struct	object

Programming Paradigms

procedural	C, Pascal, COBOL
functional	LISP, ML
recursive	LISP, Prolog
logical/declarative	Prolog
constraint-based	Prolog III
object-oriented	Smalltalk, Java, C++
rule-based	OPS5
mathematical	Mathematica

Models of Computation

- table lookup
- register manipulation
- predicate calculus
- lambda calculus, combinators
- recursive function theory
- term-rewriting
- graph-rewriting
- matrix algebra
- relational database
- cellular automata

Mathematical Structures

- propositional calculus (boolean algebra)
  - truth symbols
  - propositional symbols (binary variables)
  - connectives (and, or, not)
  - interpretations
- predicate calculus
  - truth symbols
  - constant symbols
  - variable symbols
  - function symbols
  - predicate symbols (relations)
  - quantifiers
- equality and orderings
- non-negative integers
- sets, bags (multi-sets)
- strings, trees, lists
- tuples (structs)
- graphs

Mathematical Abstractions

Relations

base
atom
compound
structure

reflexive	all x | (x,x) inR
symmetric	if (x,y) inR, then (y,x) inR
transitive	if (x,y) inR and (y,z) inR, then (x,z) inR
antisymmetric	if (x,y) inR and (y,x) inR, then x = y
trichotomy	(x,y) inR xor (y,x) inR xor x=y
irreflexive	not reflexive
asymmetric	not symmetric

Functions (binary relations with existence and uniqueness)

base
compound
structure

identity	Id op A = A op Id = A
inverse	A op iA = iA op A = Id
associative	(A op B) op C = A op (B op C)
commutative	A op B = B op A
distributive	A op1 (B op2 C) = (A op1 B) op2 (A op1 C)
idempotent	A op A = A

Equations (equivalence relations)

theorems (proved)
axioms (assumed)
generate
    base, atom, compound
unique
    base, compound