Concepts of Programming Languages, CSCI 305, Fall 2021
Logic Languages, Chapter 12, Oct. 13

Logic programming – programming paradigm largely based on formal logic

Logic program – set of sentences, in a logical form, expressing facts and rules about some problem domain

	Imperative	**Functional**	**Logic**
Example languages	Fortran, Pascal, C, Java, scripting most that we use	Scheme, LISP, ML Haskel, Single Assignment C	Prolog, Mercury (very few)
Basis	Turing machines (Alan Turing)	Lambda calculus (Alonzo Church)	Mathematical logic (Aristotle)
Computes principally using	Iteration and side effects	Substitution of parameters into functions	Resolution of logical statements, driven by the ability to unify variables and terms
Characteristics	Mirrors underlying hardware and can be "tweaked" for high performance	Avoids the semantic complexity of side effects. Particularly good for symbolic manipulation. Since referentially transparent, easier to reasoning about, good for parallel processing.	Well suited for problems that emphasize relationships and search. Can be considered "runnable specifications" Naturally parallel Used for formal specification, expert systems, theorem proving, and sophisticated control systems
Power (aside from hardware imposed restrictions)	Full power of Turing machines (Turing complete)	Full power of Lambda calculus (Turing complete)	Less than full generality of resolution theorem proving (Turing complete)
Programming constructs added that weren't in the basic model	Nothing needed	I/O and precision Some languages add assignment	I/O, true arithmetic, imperative control flow, high-order predicates for self-inspection and modification

Commonalities: Each of the above languages needs to be scanned, parsed and analyzed semantically. Each has naming and scoping issues, types, expressions and the control-flow concepts of selection and recursion.