$\rightarrow$ Answers to fundamental questions of what is doable with computing machines and how, in principle, can it be done

$\rightarrow$ Study of simple theoretical models to this effect

$\rightarrow$ Examples of things that are doable $\leadsto$ monitoring | controlling complex technical equipment $\leadsto$ production and power plants | air and ground traffic control systems | airplanes | cars | robots | communication systems (internet)

$\leadsto$ complex numerical | symbolic computations (scientific computing) | logical reasoning | maintaining and managing huge data bases

$\leadsto$ picture | sound | speech analysis | interpretation

$\leadsto$ structure analysis $\leadsto$ bio-computing

$\rightarrow$ not doable $\leadsto$ completely meaning-preserving translation | interpretation of natural languages

$\leadsto$ generally all activities requiring natural intelligence

$\leadsto$ creating consistent formal systems, theorems etc. | inventing machinery | architecting large technical systems ...

or even simpler things like $\leadsto$ building a bird's nest from the branches of a tree | solving puzzles | conjecturing the age of a person from a picture analysis ...

$\rightarrow$ Computers cannot do anything on there own $\rightarrow$ merely provide platforms (mechanisms, operations, ressources) for

$\rightsquigarrow$ state– | term–transforming processes

$\rightarrow$ what the processes are supposed to do (and how it is to be done) is specified by means of algorithms

$\rightsquigarrow$ derive problem solutions from problem specifications by partially or fully ordered sequences of state or term transformations

$\rightsquigarrow$ realize functions or relations

$\rightsquigarrow$ are based on calculi

$\rightarrow$ Calculus $\rightsquigarrow$ defined by a finite set of syntactical constructs and by a finite set of re-write rules which transform syntactical figures of the calculus into others

$\rightarrow$ Algorithms | calculi | basic machine models for computations are the concepts which give answers to the basic question $\rightsquigarrow$ what is doable with computers

$\rightarrow$ Computer architecture $\rightsquigarrow$ how the machines are built | performance | capacities play only a secondary role

$\rightsquigarrow$ a personal computer can - in principle - do the same as a huge main frame

$\rightsquigarrow$ relevant are merely computational completeness of the basic operational and control mechanisms

$\rightsquigarrow$ the operational mechanism prescribes how operators are supplied with (applied to) operands

$\rightsquigarrow$ the control mechanism prescribes how the computation is step by step driven ahead

Correctness of algorithms

$\rightarrow$ sources of algorithmic errors are
$\rightsquigarrow$ erroneous construction $\rightsquigarrow$ violation of syntactical rules
$\rightsquigarrow$ easy to detect
$\rightsquigarrow$ erroneous interpretation | violation of semantic rules
$\rightsquigarrow$ algorithmic errors $\rightsquigarrow$ algorithms do not realize the intended functions | relations over their domains
$\rightsquigarrow$ termination with erroneous results | no termination at all for certain input values (arguments) from the domain
$\rightsquigarrow$ non-terminating loops | recursive function | procedure calls $\rightsquigarrow$ oscillating | diverging
$\rightarrow$ partial correctness $\rightsquigarrow$ if an algorithm terminates for input values from within the intended domain, then it also terminates with correct results
$\rightarrow$ total correctness $\rightsquigarrow$ an algorithm terminates for all input values from the intended domain, and it does so with correct results $\rightsquigarrow$ the algorithm realizes the intended function | relation

$\rightarrow$ the fundamental problem $\rightarrow$ is there a mechanizable method to prove these properties ?
$\rightarrow$ it can be shown that such verifiers cannot be constructed $\rightarrow$ the problem is not computable | not decidable
$\rightsquigarrow$ can only be solved by natural intelligence | intuition on a case by case basis
$\rightsquigarrow$ if an algorithm is correct, then this can also be verified (proved), but not vice versa.