

Abstractions: The Heart of Computer Science

Fundamental Abstractions

Abstract Implementations

Declarative Abstractions

Computational Abstractions

Jeffrey D. Ullman
Stanford University/Infolab



Abstractions In Computer Science

- An *abstraction* consists of:
 1. A data model.
 2. A “programming language” for manipulating data.
- It is the second part that distinguishes abstractions of CS from abstractions in mathematics or other disciplines.

Example: The Dictionary Abstraction

- **Data model:** a universal set U and a set $S \subset U$.
- **Programming language:** Sequences of three operations:
 1. **Insert**(x): $S := S \cup \{x\}$.
 2. **Delete**(x): $S := S - \{x\}$.
 3. **Lookup**(x): true if $x \in S$ else false.
- **Example:** U = character strings, S = currently valid variable names in a program (used for compiler's symbol table).

Taxonomy of Abstractions

- Based on their purpose.
- **Fundamental Abstractions**: To study algorithms for commonly occurring computations.
- **Abstract Implementations**: To support fundamental abstractions, not for their own sake.
- **Declarative Abstractions**: Tell what you want, not how to do it.
- **Computational Abstractions**: Directly executable or close to it.

Fundamental Abstractions

- Two kinds:
 - **Stand-alone** – studied in isolation.
 - **Examples**: Dictionary, Stack, Queue, Priority queue,...
 - Intended to be **embedded** in a general-purpose language.
 - Essentially classes + their methods, but without implementation.
 - **Examples**: many kinds of graphs (directed, undirected, labeled, multigraphs, unrooted trees, rooted trees,...)
 - Why embedded? Say a graph abstraction has a “find neighbors” method. You need a loop outside to go over each neighbor.

Abstract Implementations

- Data structures useful for implementing many fundamental abstractions.
- **Examples:** hash tables, search trees, linked lists.
- Admit different implementations at the machine level.
 - **Example:** a hash table could be implemented in a RAM model of a computer or in a secondary-storage model, with very different notions of running time.
 - Disk-oriented model counts only disk I/O's for running time, not machine instructions.

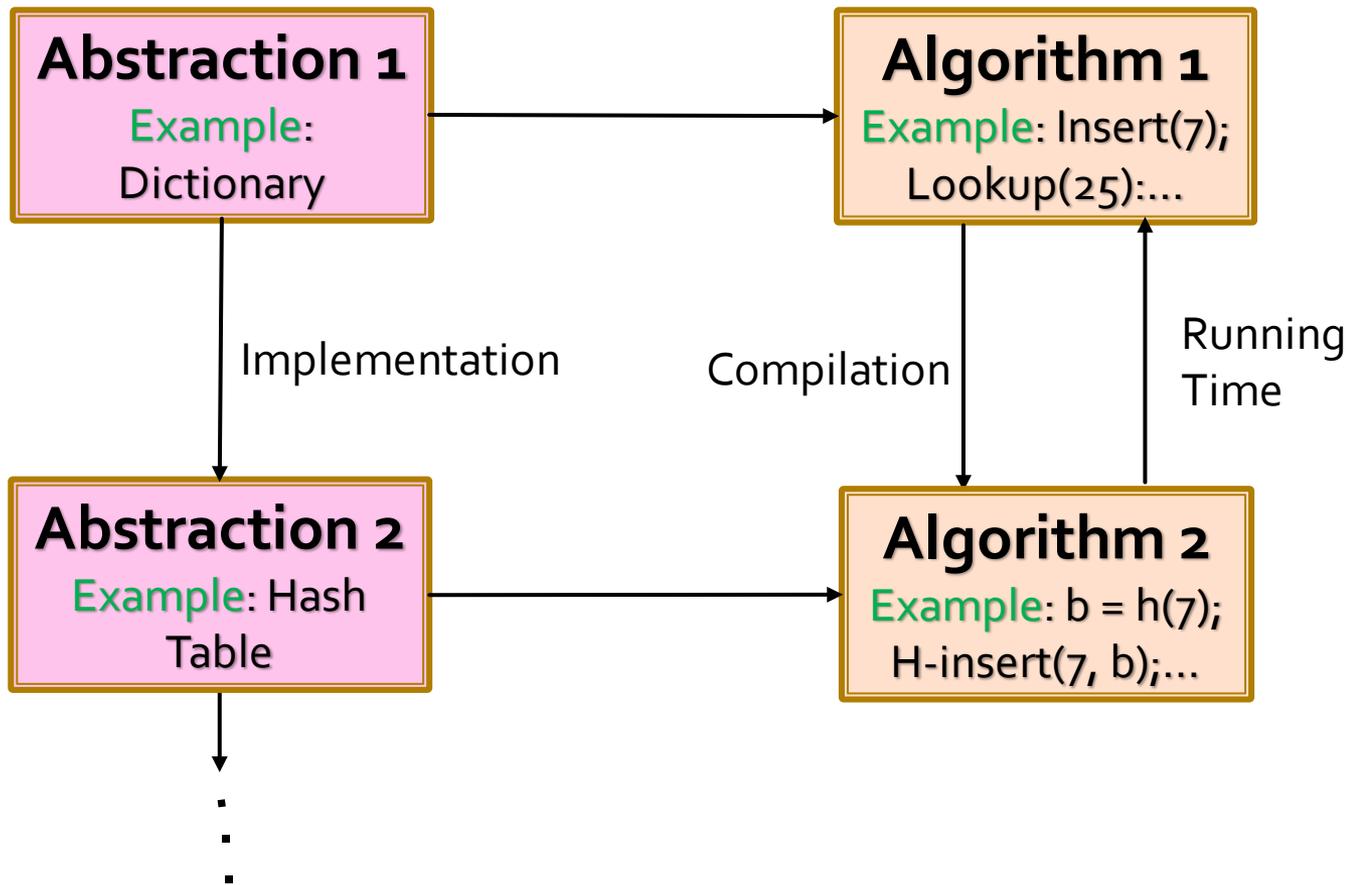
Declarative Abstractions

- Sufficiently high-level that serious optimization is needed to be executable.
 - Valuable because they make programming much easier.
- Never Turing-complete.
- **Examples:** regular expressions, context-free grammars, linear algebra, relational algebra.

Computational Abstractions

- Actual machine or directly translatable.
- **Serial-machine examples:** C++, Python, RAM.
- **Parallel/distributed-machine examples:** PRAM, MapReduce.
- Abstractions for computer-like things that are not “conventional.”
 - **Examples:** Turing machine, Boolean circuits, Quantum circuits.

Compilation and Running Time



Abstractions and Algorithms for Compiler Design

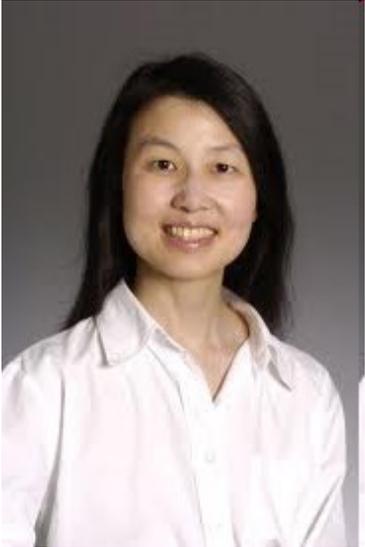
Computation Thinking
Compiler Framework
Abstractions and Algorithms
Impact

Alfred V. Aho
Columbia University



COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

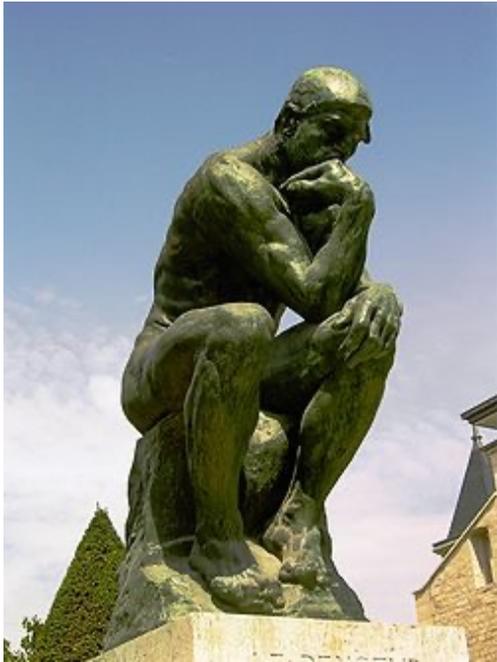
Computational Thinking - 1



Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability.

Jeannette M. Wing
Computational Thinking
CACM, vol. 49, no. 3, pp. 33-35, 2006

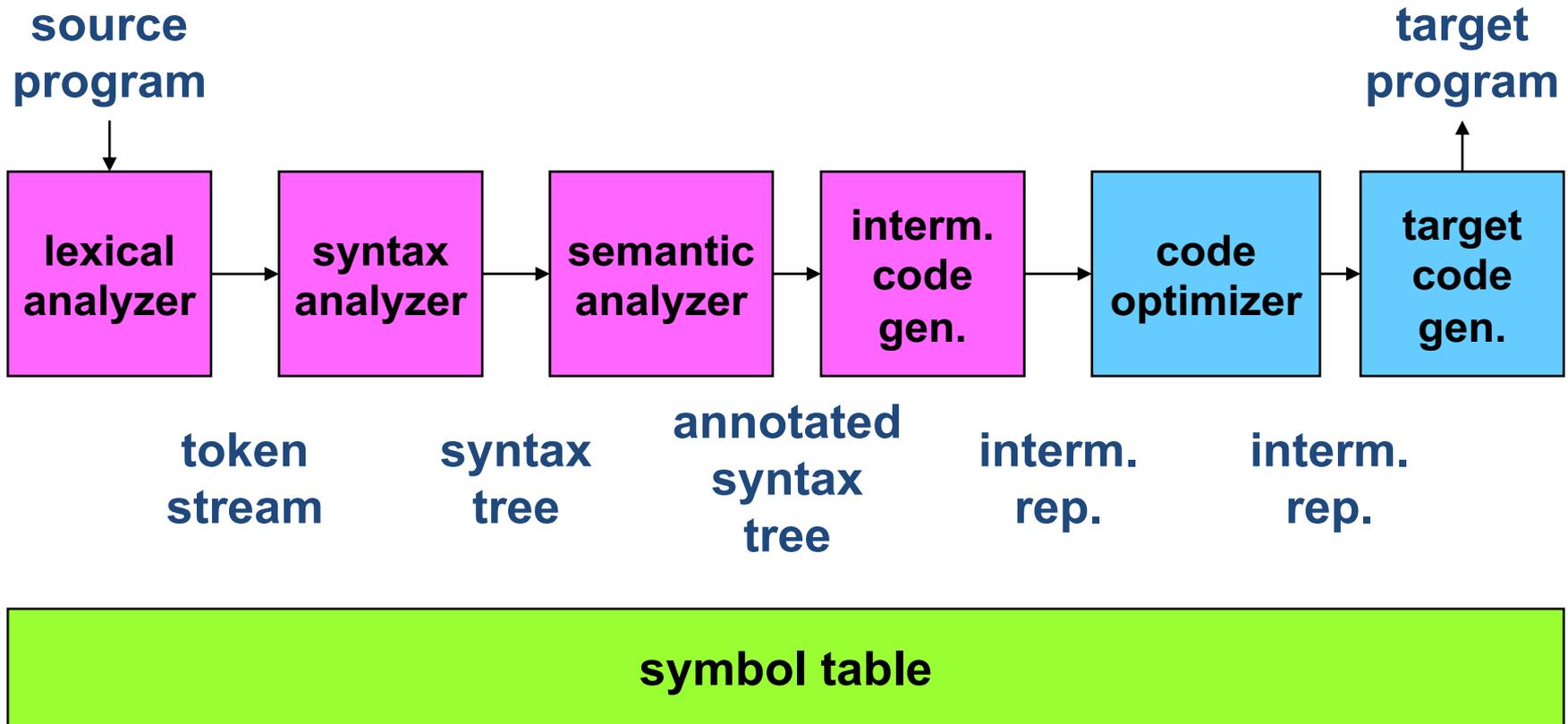
Computational Thinking - 2



The thought processes involved in formulating problems using **abstractions** so their solutions can be represented as computational steps and **algorithms**.

Alfred V. Aho
Computation and Computational Thinking
The Computer Journal, vol. 55, no. 7, pp. 832-835, 2012

Phases of a Compiler



Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman
Compilers: Principles, Techniques, & Tools
Addison Wesley, 2007

Regular Expression Abstraction

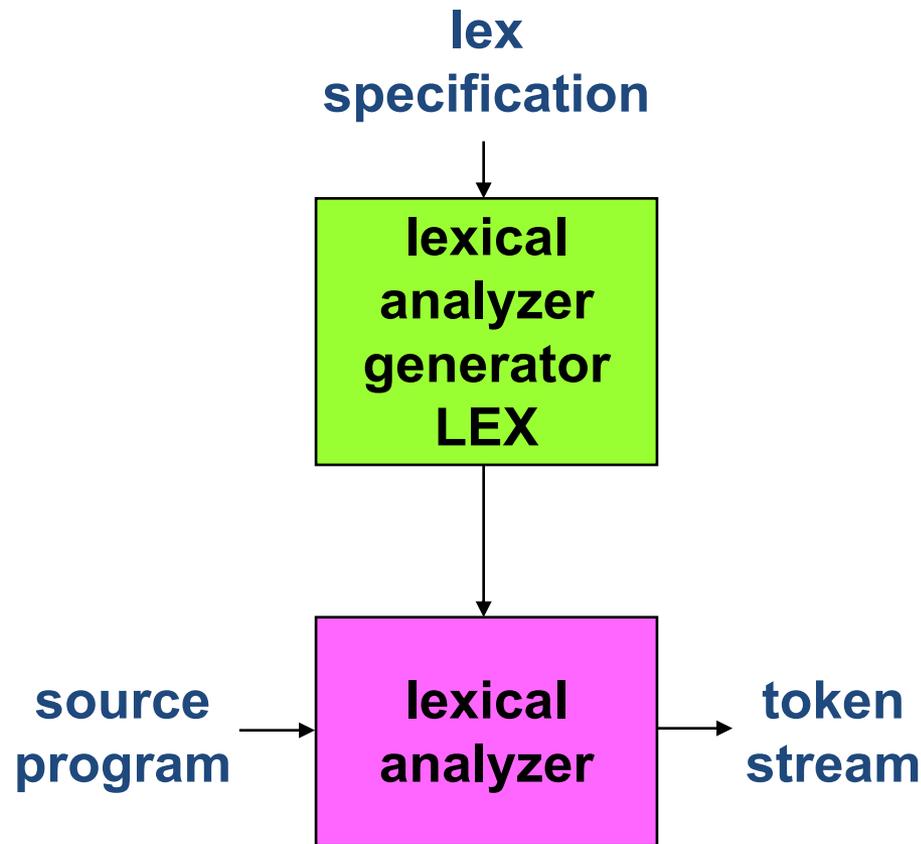
- Data model
 - Sets of strings
- Operations
 - Union
 - Concatenation
 - Kleene star
- Examples
 - integer: $[0-9][0-9]^*$
 - identifier: $[A-Za-z][A-Za-z0-9]^*$

Regular Expression Matchers

- Nondeterministic finite automata
 - McNaughton-Yamada-Thompson construction:
 $O(|RE| \times |input|)$ time
- Deterministic finite automata
 - Subset construction converting NFA to DFA:
 $O(2^{|RE|} + |input|)$ -time algorithm
 - Dynamic construction of DFA transitions:
 $O(|RE| + |input|)$ time observed

Alfred V. Aho
Algorithms for Finding Patterns in Strings
Handbook of Theoretical Computer Science, Volume A, pp. 255-300, 1990

Lexical Analyzer Generator



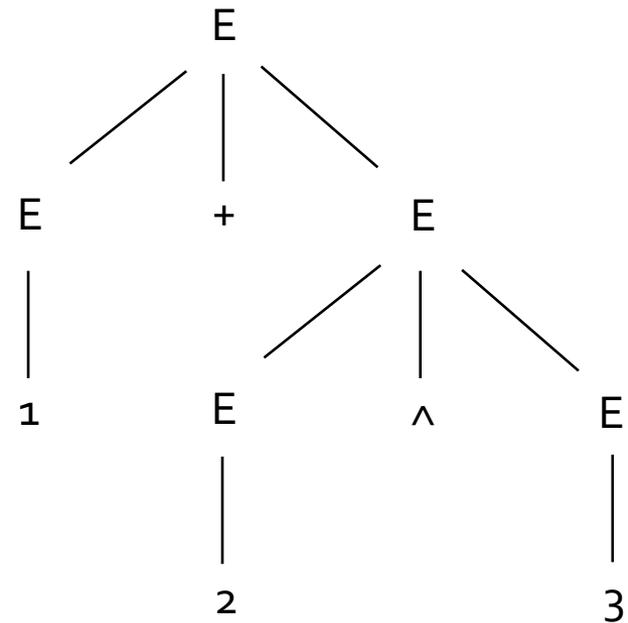
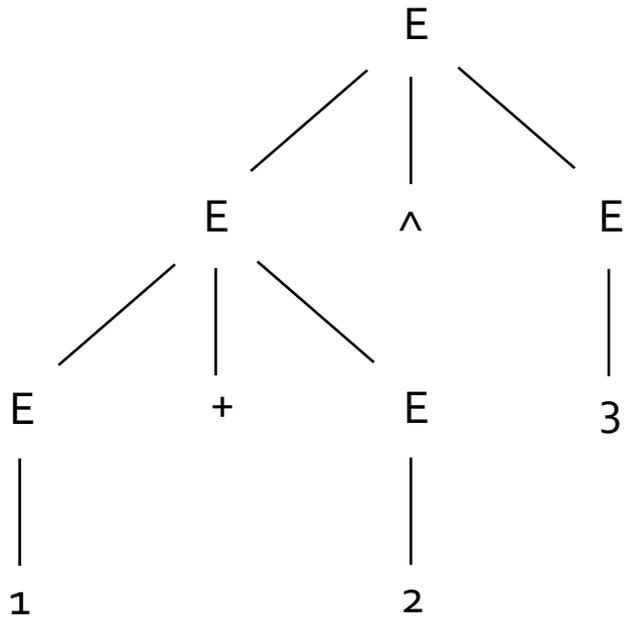
Michael E. Lesk and Eric Schmidt
Lex – A Lexical Analyzer Generator
CSTR 39, Bell Labs 1975

Grammar Abstraction

Context-free grammar for arithmetic expressions with addition and exponentiation:

$$E \rightarrow E + E \mid E \wedge E \mid 1 \mid 2 \mid 3$$

Shift-Reduce Parsing $1+2^3$



Shift-reduce parsing actions showing (stack, input):

$(\varepsilon, 1+2^3) \Rightarrow_s (1, +2^3) \Rightarrow_r (E, +2^3) \Rightarrow_s (E+, 2^3) \Rightarrow_s (E+2, \wedge 3) \Rightarrow_r (E+E, \wedge 3)$: Here we have reached a **shift-reduce conflict**.

Unambiguous Grammar

- Unambiguous grammar for arithmetic expressions:

$$E \rightarrow E + T \mid T$$

$$T \rightarrow F \wedge T \mid F$$

$$F \rightarrow 1 \mid 2 \mid 3$$

- This grammar makes the addition operator + left associative, the exponentiation operator ^ right associative and gives the operator ^ a higher precedence than +.

Augmented Grammar

- Shift-reduce parsable ambiguous grammar with disambiguating rules:

%left '+'

%right '^'

$E \rightarrow E + E \mid E \wedge E \mid 1 \mid 2 \mid 3$

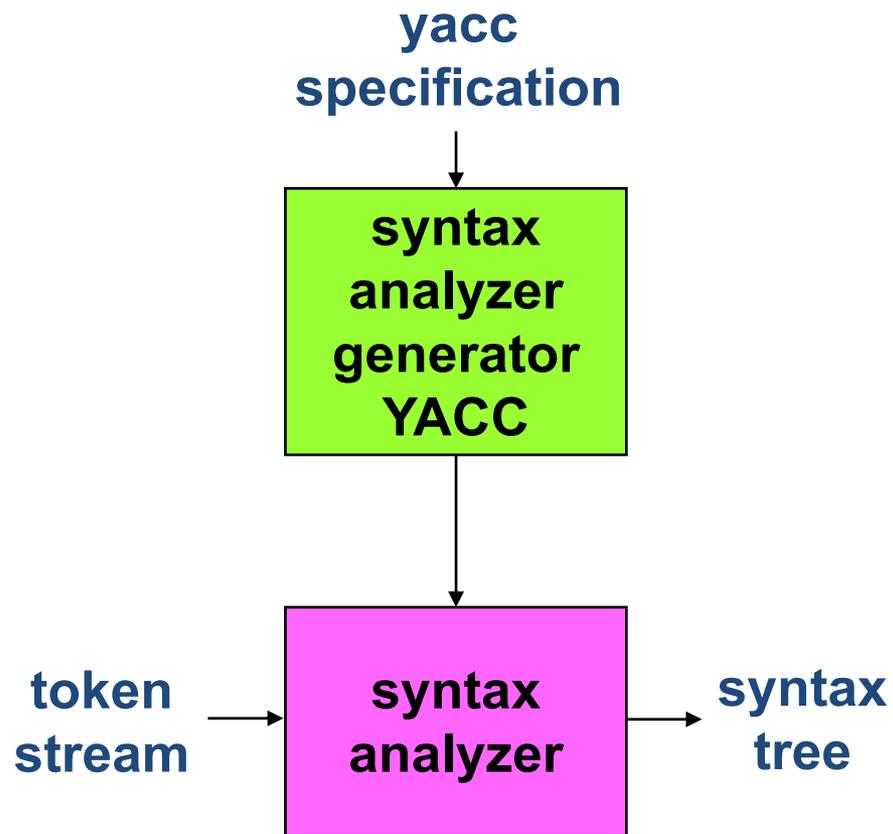
Shift-reduce parsing actions:

$(\varepsilon, 1+2^3) \Rightarrow_s (1, +2^3) \Rightarrow_r (E, +2^3) \Rightarrow_s (E+, 2^3) \Rightarrow_s (E+2, ^3) \Rightarrow_r$

$(E+E, ^3) \Rightarrow_s (E+E^, 3) \Rightarrow_s (E+E^3, \varepsilon) \Rightarrow_r (E+E^E, \varepsilon) \Rightarrow_r (E+E, \varepsilon) \Rightarrow_r (E, \varepsilon)$

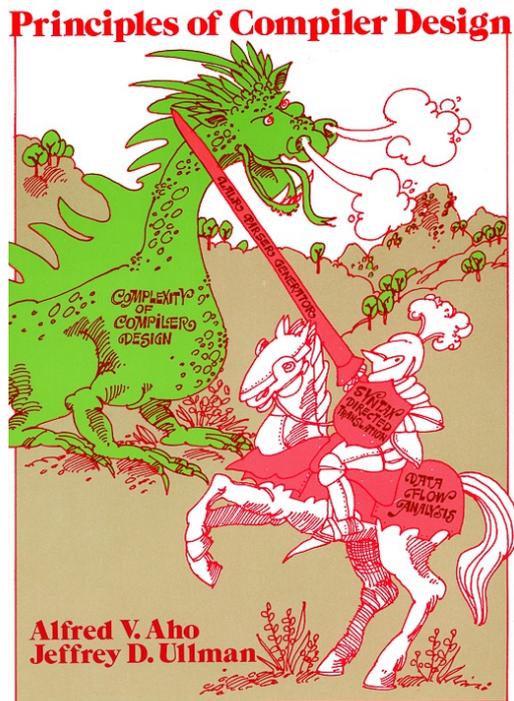
A. V. Aho, S. C. Johnson, and J. D. Ullman
Deterministic parsing of ambiguous grammars
ACM POPL, pp. 1-21, 1973

Syntax Analyzer Generator



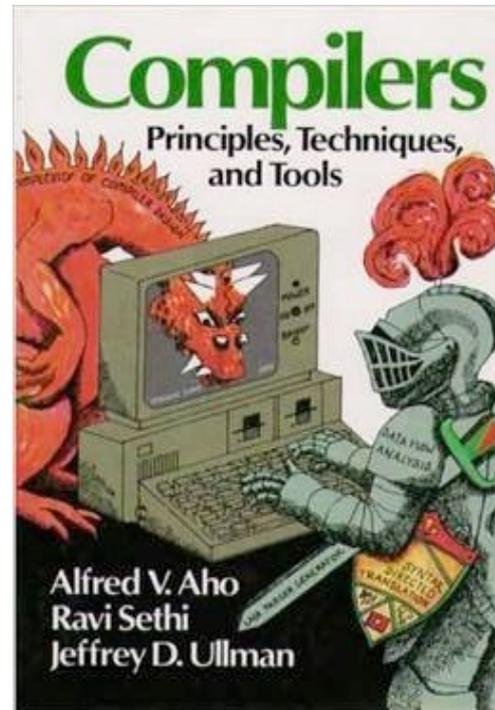
Stephen C. Johnson
Yacc-Yet Another Compiler Compiler
CSTR 32, Bell Labs, 1975

The Dragon Books



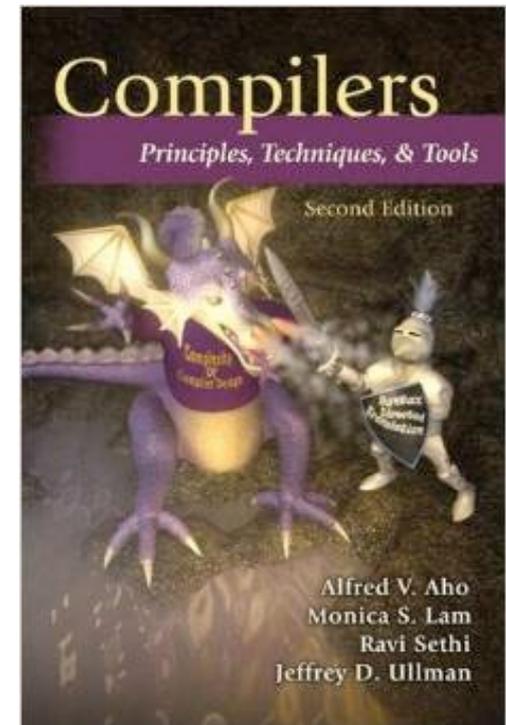
1977

finite automata
grammars
lex & yacc
syntax-directed translation



1986

type checking
run-time organization
automatic code
generation



2007

garbage collection
optimization
parallelism
interprocedural analysis

Impact

- Early programming languages created with Lex and Yacc include
 - ampl for mathematical programming
 - awk for data processing and transformation
 - eqn for typesetting mathematics
 - pcc, a portable C compiler
 - pic for typesetting figures
- The dragon books have been used for decades in universities around the worldwide in compiler courses
- A small team of students can create a new programming language and build a compiler for it in a one-semester compiler course

Conclusion

Abstractions and algorithms have transformed the field of compiler design from an art to a science.

Abstractions for Large-Scale Data

Secondary-Storage Abstraction

The Relational Model

The MapReduce Abstraction

Jeffrey D. Ullman
Stanford University/Infolab



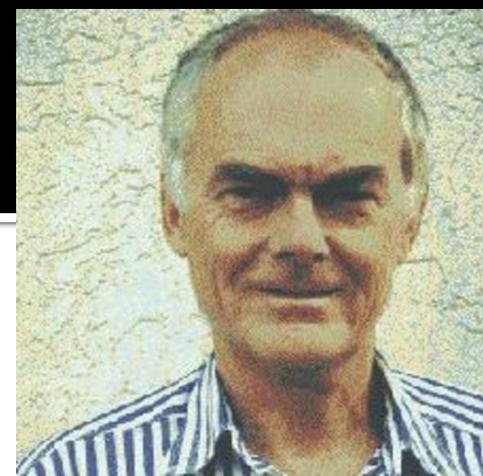
The Abstractions of Database Systems

- **Key Computational Abstraction:** secondary-storage model.
 - Addresses real issues regarding efficiency of database algorithms.
 - Yields different results for some problems.
 - **Example:** Balanced search trees (B-trees) and MergeSort look very good in this model.
- **Key Declarative Abstraction:** relational model.
- **Parallel-computing** abstractions.
 - **Examples:** shared-nothing, bulk-synchronous, MapReduce.

Secondary-Storage Abstraction

- When data is so large, it has to reside primarily on disk, you have a Computational Abstraction with a different measure of running time.
- Access to any data element requires moving its entire disk block to main memory.
 - Takes more time than what you typically do with the data once it is in main memory.
- **Leads to:** running time = number of disk I/O's.

B-Trees

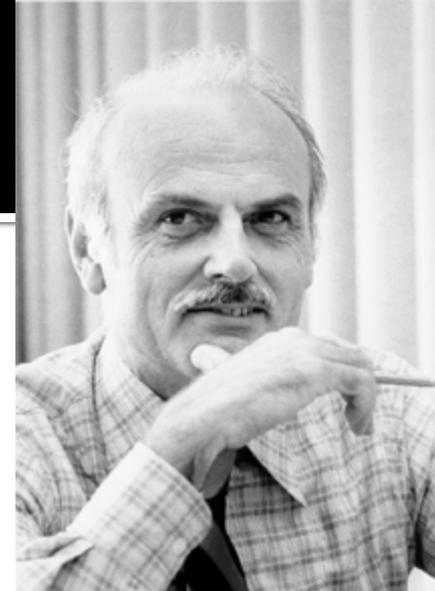


- Balanced search tree whose nodes are entire disk blocks.
- Nodes of a binary search tree consist of two pointers, to left and right children plus a value that separates items found to left and right.
- In a B-tree, a node has pointers to n children and $n - 1$ values that separate items found at each of the children.
 - n can be in the thousands, so 3 levels of B-tree are sufficient for billions or trillions of items.

B-Trees – (2)

- In the secondary-storage model, B-trees are a very efficient implementation of Dictionaries.
- Root node can be stored in main memory.
- Lookup in a 3-level B-tree requires only 2 disk I/O's.
- Insert/Delete can usually be done in 3 I/O's.
 - Occasionally, needs a few extra I/O's to rebalance the tree.

Relational Model



- Tables with sets of rows and with named columns.
- Examples:

City	State
Toronto	Ontario
NewYorkCity	NewYork
Mumbai	Maharashtra
Montreal	Quebec
SanJose	California
Chennai	TamilNadu

Cities

State	Country	Pop
California	USA	38.51
Maharashtra	India	114.2
NewYork	USA	19.45
Ontario	Canada	14.57
Quebec	Canada	8.49
TamilNadu	India	67.86

States

Relational-Model Programming Languages

- Codd could have viewed the relational model as a Fundamental Abstraction, like graphs, where the operations are simple navigations among rows or columns.
 - In fact, earlier database abstractions had exactly that kind of programming language.
- Rather, the relational model has always been viewed as a Declarative Abstraction, with a very high-level programming language.

What Programming Language?

- It depends on what year it is.
- Codd suggested two different, but equivalent, approaches:
 - Relational algebra.
 - Two forms of nonrecursive first-order logic.
- Original SQL added:
 - Bag and list options for data.
 - Grouping, aggregation, sorting operations.
- Datalog added recursion.



Five Relational-Algebra Operations

- Union, Set-Difference, Selection, Projection, Join.
- **Selection**: Take only the rows that meet a condition.
 - **Example**: Pop \geq 20.

State	Country	Pop
California	USA	38.51
Maharashtra	India	114.2
New York	USA	19.45
Ontario	Canada	14.57
Quebec	Canada	8.49
TamilNadu	India	67.86

Projection

- Remove columns and eliminate duplicates.
- **Example:** Project States onto Country.

State	Country	Pop
California	USA	38.51
Maharashtra	India	114.2
NewYork	USA	19.45
Ontario	Canada	14.57
Quebec	Canada	8.49
TamilNadu	India	67.86

Join – Applies to Two Relations

- Consider all pairs of rows, one from each relation.
- Construct a row from each pair that meets a condition.
- **Example:** join Cities and States on the condition that the State value from each relation is the same.

Example: Join

City	State	State	Country	Pop
Toronto	Ontario	California	USA	38.51
NewYorkCity	NewYork	Maharashtra	India	114.2
Mumbai	Maharashtra	NewYork	USA	19.45
Montreal	Quebec	Ontario	Canada	14.57
SanJose	California	Quebec	Canada	8.49
Chennai	TamilNadu	TamilNadu	India	67.86

Query Optimization

- Because of the declarative nature of the relational model and extensions, optimization is essential.
- Conventional languages like C have “optimizing compilers,” but they don’t change the algorithm written in the source language.
- If I write Bubblesort in C, the compiler won’t change it to Quicksort, because undecidability of Turing-complete languages precludes the compiler detecting Bubblesort or anything else.

Query Optimization – (2)

- Orders of magnitude improvement possible with relational algebra.
- **Example:** Find the population of Mumbai's state.
- **Obvious plan:** join Cities and States, then select for City = 'Mumbai' and project onto Pop.
- **Much better plan:** select on Cities for City = 'Mumbai', find that Mumbai is in Maharashtra state, select on States for State = 'Maharashtra', project onto Pop.

MapReduce Computational Abstraction

- Assumes a computing cluster as “the machine.”
 - Many processors with communication possible but not free.
- **Data model**: Sets of key-value pairs.
- **Programming language**: conventional Turing-complete language, but organized as two functions, **Map** and **Reduce**.



MapReduce – (2)

- Allows parallel computation without “thinking parallel.”
- Apply **Map** function to each input key-value pair.
 - Generates zero, one, or many key-value pairs.
 - Usually of a different type.
- Behind the scenes, all generated pairs are sorted by key and organized into (key, list-of values) pairs.
- Apply **Reduce** function to each (key, list-of-values).
 - The result of the job is the key-value pairs produced by all the applications of **Reduce**.

MapReduce Running Time

- Several interesting considerations when developing good MapReduce algorithms.
- Wall-clock time of the parallel algorithm.
- Rental cost of cloud hardware.
 - Time multiplied by number of processors.
- Communication cost can dominate when Map function produces many pairs.
- Number of jobs needed.
 - Assumes no processor can handle more than a negligible fraction of the data.

Future Directions

- Hardware platforms are evolving.
 - **Special-purpose accelerators**: GPU, TPU, FPGA, etc.
 - **Architectures**: e.g., “serverless” where data is stored in a file system and execution occurs at short-term-rental processors.
- Are there good abstractions like MapReduce for exploiting these and other developments?

Abstractions in Quantum Computing

Postulates of Quantum Mechanics
Superposition and Entanglement
Quantum Circuits
Quantum Algorithms

Alfred V. Aho
Columbia University



COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Quantum Mechanics

The four postulates of quantum mechanics

M. A. Nielsen and I. L. Chuang
Quantum Computation and Quantum Information
Cambridge University Press, 2011

Postulate 1: State Space

The **state space** of an isolated physical system can be modeled by a Hilbert space, a complex vector space with an inner product.

The state of the system is completely described a **unit vector** in this state space.

Qubit: Quantum Bit

The state of a quantum bit, called a **qubit**, can be described by a unit vector in a 2-dimensional state space

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex coefficients called the **amplitudes** of the computational basis states

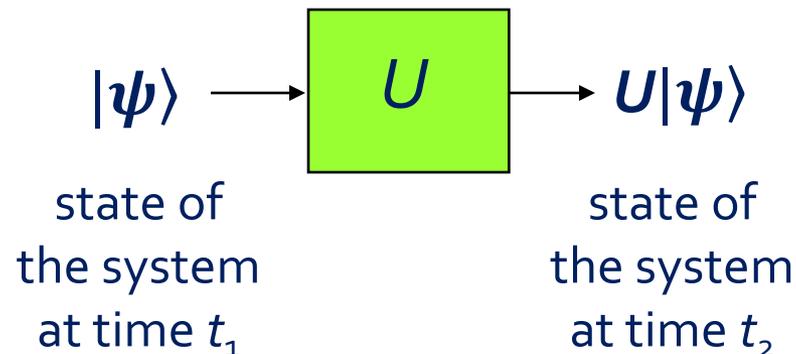
$|0\rangle = (1,0)$ and $|1\rangle = (0,1)$. Note that $|\alpha|^2 + |\beta|^2 = 1$.

$|\psi\rangle$ is said to be in a **superposition** of $|0\rangle$ and $|1\rangle$.

Postulate 2: Time Evolution

The evolution of the state of a closed quantum system from one time to another can be described by a **unitary operator**.

A matrix U is **unitary** if $UU^\dagger = I$ where I is the identity. U^\dagger is the conjugate-transpose of the matrix U .



Single-qubit Hadamard Operator

The Hadamard operator has the matrix representation:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

H maps the qubit $\alpha|0\rangle + \beta|1\rangle = (\alpha, \beta)$ to the qubit

$$\frac{\alpha}{\sqrt{2}} (|0\rangle + |1\rangle) + \frac{\beta}{\sqrt{2}} (|0\rangle - |1\rangle) = \left(\frac{\alpha + \beta}{\sqrt{2}}, \frac{\alpha - \beta}{\sqrt{2}} \right)$$

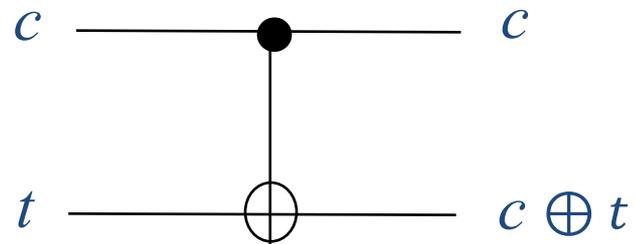
Note that $HH = I$.

Two-qubit CNOT Operator

The two-qubit CNOT
(controlled-NOT) operator
has the matrix
representation on the right:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

CNOT flips the target qubit t
iff the control qubit c has
the value 1:



CNOT gate

The CNOT gate maps

$$\begin{array}{ll} |00\rangle \rightarrow |00\rangle, & |01\rangle \rightarrow |01\rangle \\ |10\rangle \rightarrow |11\rangle, & |11\rangle \rightarrow |10\rangle \end{array}$$

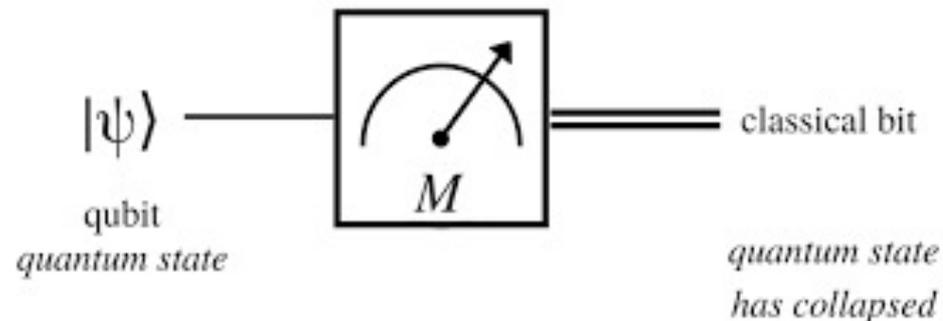
$|xy\rangle$ denotes the tensor product $|x\rangle \otimes |y\rangle$.

Postulate 3: Measurements

- To get information from a closed quantum system, measurements need be applied to the system.
- A measurement returns an outcome with some probability.
- The sum of the probabilities of the possible outcomes is 1.
- A measurement collapses the state of a quantum system.

Measurement of a Single Qubit

- Consider the measurement of a qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in the computational basis.
- The output of the measurement is a classical bit. The probability of getting 0 is $|\alpha|^2$ and the probability of getting 1 is $|\beta|^2$. Note that $|\alpha|^2 + |\beta|^2 = 1$.
- If the outcome is 0, the state after measurement is $|0\rangle$; if the outcome is 1, the state after measurement is $|1\rangle$.



Postulate 4: Composite Systems

The state space of a composite physical system is the **tensor product** space of the state spaces of its component subsystems.

Example: If $A = (a_1, a_2)$, $B = (b_1, b_2)$, $C = (c_1, c_2)$, then $A \otimes B$ is the vector $(a_1b_1, a_1b_2, a_2b_1, a_2b_2)$;

$A \otimes B \otimes C$ is the vector

$$(a_1b_1c_1, a_1b_1c_2, a_1b_2c_1, a_1b_2c_2, \\ a_2b_1c_1, a_2b_1c_2, a_2b_2c_1, a_2b_2c_2)$$

Tensor Product

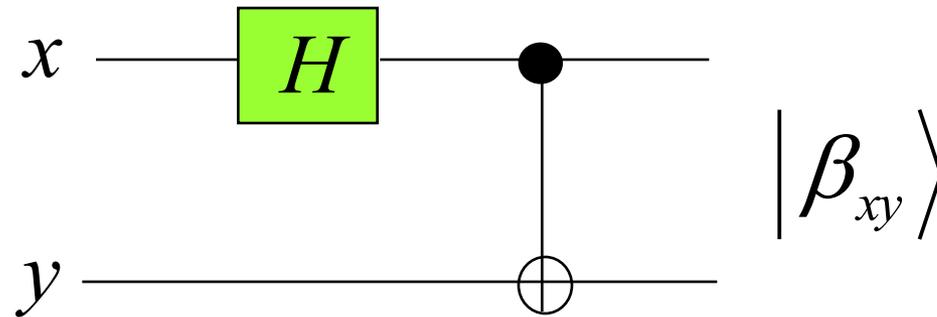
If we combine n single-qubit systems, we get a composite system with a state space of dimension 2^n by taking the tensor product of the state spaces of the n single-qubit systems.

Quantum Circuit Abstraction

- A quantum circuit is an acyclic graph consisting of lines, quantum gates, and measurement gates.
- No fan-in: logical OR is not a unitary operation.
- No fan-out: in quantum mechanics, it is not possible to make a copy of an unknown quantum state (the no-cloning theorem).

Quantum Circuit for EPR States

Quantum circuit to generate Einstein-Podolsky-Rosen states, also known as Bell states:

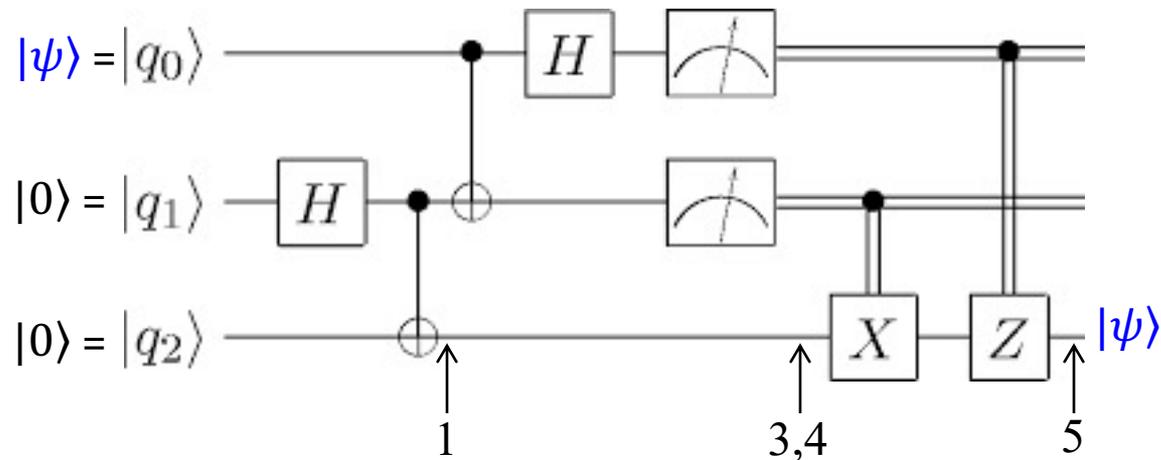


Circuit maps

$$|00\rangle \mapsto \frac{(|00\rangle + |11\rangle)}{\sqrt{2}}, |01\rangle \mapsto \frac{(|01\rangle + |10\rangle)}{\sqrt{2}}, |10\rangle \mapsto \frac{(|00\rangle - |11\rangle)}{\sqrt{2}}, |11\rangle \mapsto \frac{(|01\rangle - |10\rangle)}{\sqrt{2}}$$

Each output is an **entangled state**, one that cannot be written in a product form. (Einstein: “Spooky action at a distance.”)

Quantum Teleportation



1. Alice and Bob generate an EPR pair by entangling $|q_1\rangle$ and $|q_2\rangle$.
2. Alice takes one half of the pair; Bob the other half. Bob moves far away taking his qubit with him.
3. Alice gets her secret qubit $|\psi\rangle$, interacts it with her EPR-half with a CNOT and Hadamard gate, and then measures the two output qubits.
4. Alice sends the two resulting classical measurement bits to Bob.
5. Using the classical measurement bits sent by Alice, Bob interacts his half of the EPR pair with a classically controlled NOT gate and a classically controlled Pauli-Z gate to recreate $|\psi\rangle$.

Shor: Integer Factorization

Problem: Given a composite n -bit integer, find a prime factor.

Best-known deterministic algorithm on a classical computer has time complexity $\exp(O(n^{1/3} \log^{2/3} n))$.

A hybrid classical-computer, quantum-computer can solve this problem in $O(n^2 \log n \log \log n)$ operations.



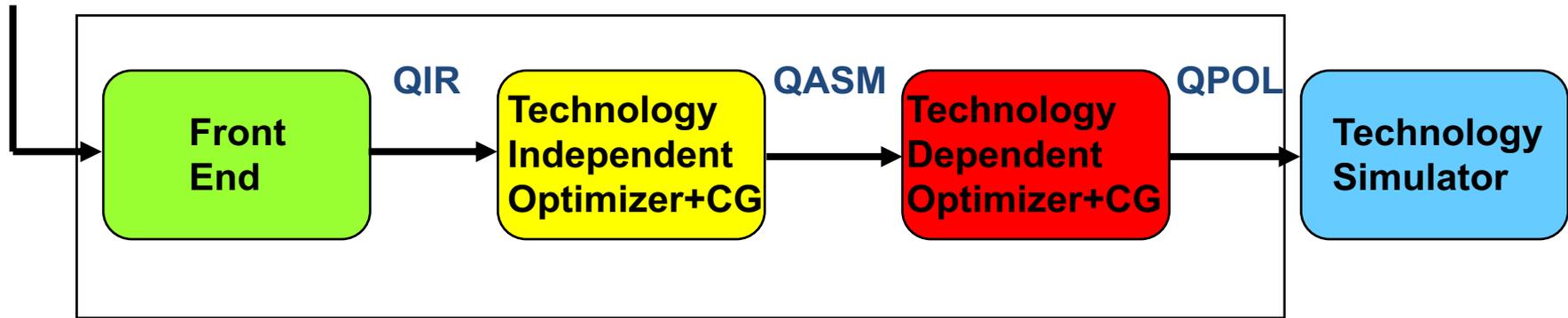
Peter Shor

Algorithms for Quantum Computation: Discrete Logarithms and Factoring
Proc. 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124-134

Quantum Computer Compilers

quantum
source
program

QIR: quantum intermediate representation
QASM: quantum assembly language
QPOL: quantum physical operations language



Quantum Computer Compiler

ABSTRACTIONS

quantum
mechanics

quantum
circuit

quantum
circuit

quantum
device

K. Svore, A. Aho, A. Cross, I. Chuang, I. Markov
A Layered Software Architecture for Quantum Computing Design Tools
IEEE Computer, 2006, vol. 39, no. 1, pp.74-83

Conclusions and Open Problems

- Quantum circuits are a key abstraction for quantum computing programming languages.
- Open problems: useful algorithms, large scalable quantum computers