

The Algol language

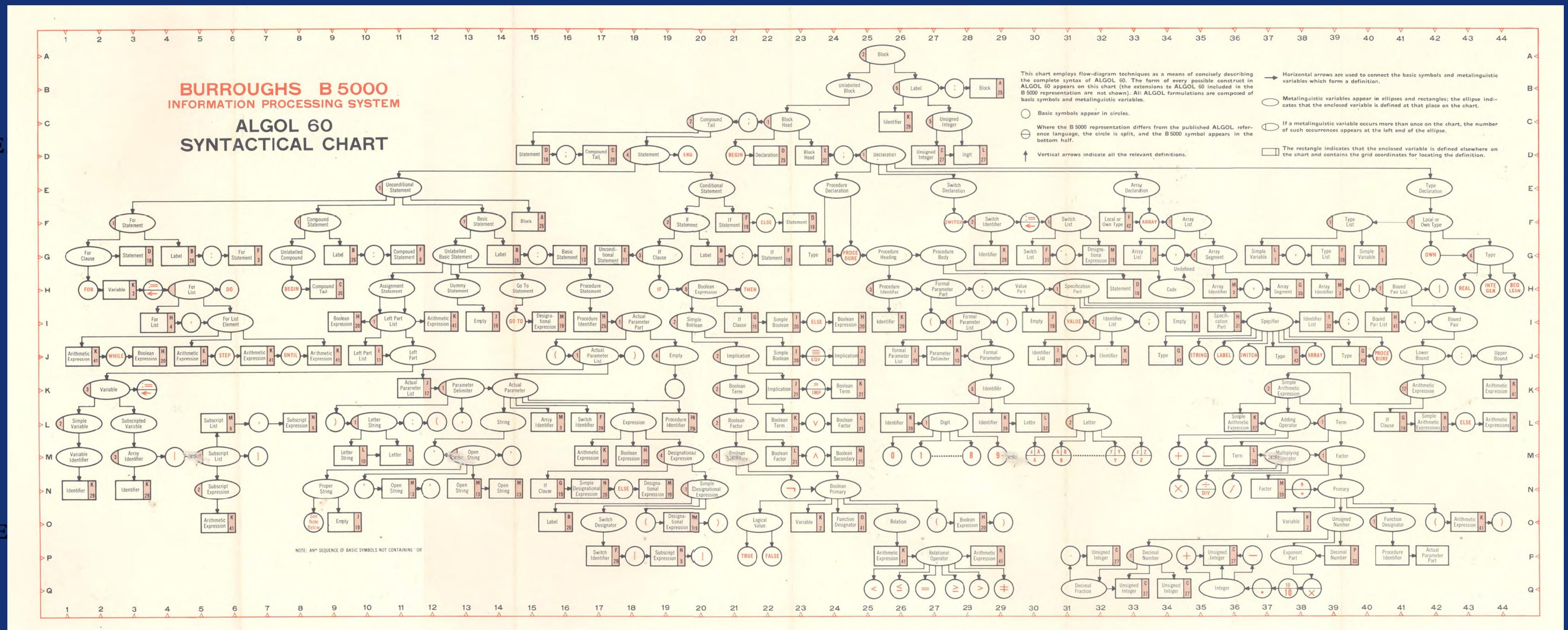
- 1.1 THE COMPUTER
- 1.2 PROBLEM SOLVING ON A COMPUTER
- 1.3 ALGORITHMS
- 1.4 LOGIC CHARTING
- 1.5 COMMUNICATING WITH THE COMPUTER
- Published in 1960, Algol was an object of stunning

beauty" but it was difficult to implement and was never widely adopted in practice.

Its mathematical definition

made it possible to reason about programs formally and Algol soon became the de facto language of academic computer science.

4. CONDITIONAL STATEMENTS	56
4.1 THE CONCEPT	56
4.2 BOOLEAN EXPRESSIONS	57
4.3 IF STATEMENT	60



Mathematical culture

The mathematical culture sees programs as mathematical entities. They can be formally analyzed and proven correct with respect to a precise specification from a customer.

Proponents of the mathematical culture believe that mathematical methods provide the right mechanism for tackling the increasing complexity of software systems.

A BASIS FOR A MATHEMATICAL THEORY OF COMPUTATION¹

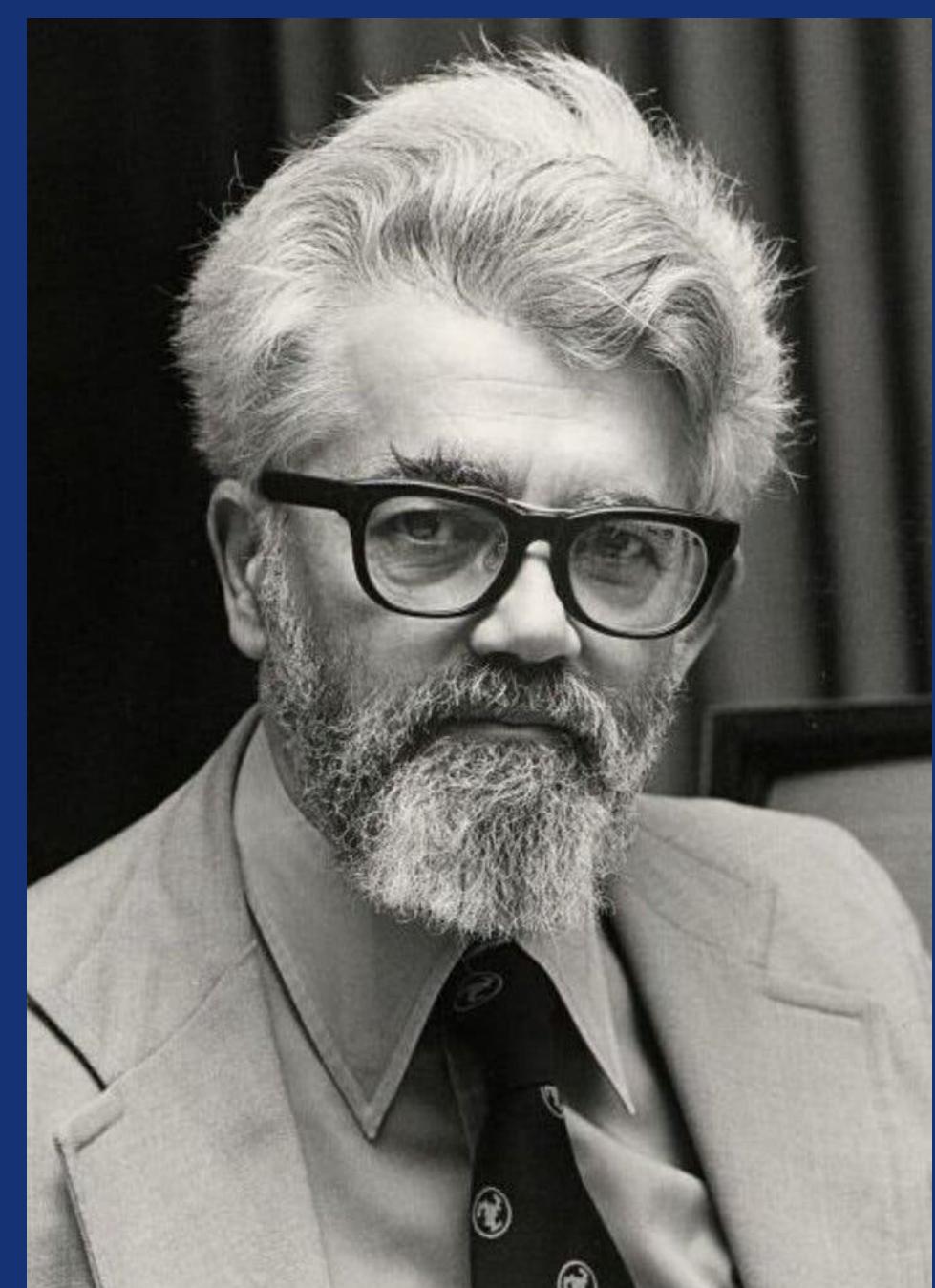
JOHN McCARTHY

Computation is to become one of the most important of the sciences. This is because it is the manner how machines can be made to carry out intellectual processes. We know that any intellectual process that can be carried out mechanically can be performed by a general purpose digital computer. The first direction of research is to make computers do so far clearly come far more from our weakness as programmers than from the intrinsic limitations of the machine. We hope that these solutions can be greatly refined by developing a mathematical science of computation.

The second relevant direction of research is the theory of computability as a science of computation. The first and oldest of these is numerical analysis. Unfortunately, its subject matter is too narrow to be of much help in solving general problems and it has only enthusiasm to be affected by the existence of automated computation.

In 1961, he outlined a mathematical theory of programs and defined questions we may want to ask—for example, are the given two programs equivalent?

The third direction of mathematical research is the theory of finite automata. Results which use the finiteness of the number of states tend not to be very useful in dealing with present computers which have so



An Axiomatic Basis for Computer Programming

C. A. R. HOARE
The Queen's University of Belfast,* Northern Ireland

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

KEY WORDS AND PHRASES: axiomatic method, theory of programming, proofs of programs, formal language definition, programming languages design, machine-independent programming, program documentation

CR CATEGORY: 4.0, 4.21, 4.22, 5.20, 5.21, 5.23, 5.24

PROOF OF ALGORITHMS BY GENERAL SNAPSHOT

PETER NAUR

Abstract.

A constructive approach to the question of proofs of algorithms is to consider proofs that an object resulting from the execution of an algorithm possesses certain static characteristics. It is shown by an elementary example how this possibility may be used to prove the correctness of an algorithm written in ALGOL 60. The stepping stone of the approach is what is called General Snapshots, i.e., executing the algorithm and then saving the state of the computer for further show to be used for proving.

Peter Naur
In 1961, he outlined a mathematical theory of programs and defined questions we may want to ask—for example, are the given two programs equivalent?

The third direction of mathematical research is the theory of finite automata. Results which use the finiteness of the number of states tend not to be very useful in dealing with present computers which have so

many more states than there will ever be in any program. This will go in parallel with the introduction of these principles in the elementary school curricula. One subject which will then come up for attention is that of proving the correctness of algorithms. The purpose of the present article is to show in an elementary way that this subject not only exists,

but is ripe to be used in practice. The illustrations are phrased in ALGOL 60, but the technique may be used with any programming language.

Copyright © 1966 by Peter Naur.

* Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

576 Communications of the ACM

Volume 9 / Number 3 / March 1966

5. The Semantics of Microalgol

The semantics of Microalgol is given by a recursively defined function $\text{micro}(\pi, \xi)$ that gives the state in which a Microalgol program π will terminate if it is entered in state ξ . First, however, we have

Academic computer science

At a time when some saw university departments for computer science as analogous to departments for railroad, automobile or television

For definiteness we shall assume that the department of computer science is real would be changed, however, if we restricted the science to mathematics helped

to establish it as a real, respectable intellectual activity. Then we have

$\text{value}(\tau, \xi) = \text{if isvar}(\tau)$

$\text{else if issum}(\tau) \text{ then value}(addend(\tau), \xi) + \text{value}(summand(\tau), \xi)$

$\text{else if isprod}(\tau) \text{ then value}(multiplier(\tau), \xi) \cdot \text{value}(product(\tau), \xi)$

$\text{else if iscond}(\tau) \text{ then value(proposition}(\tau), \xi) \text{ else value(antecedent}(\tau), \xi) \text{ else value(then}(\tau), \xi)$

$\text{else if isleft}(\tau) \text{ then value(leftl}(\tau), \xi) + \text{value(leftr}(\tau), \xi)$

$\text{else if isright}(\tau) \text{ then value(rightl}(\tau), \xi) + \text{value(rightr}(\tau), \xi)$

$\text{else if isseq}(\tau) \text{ then value(left}(\tau), \xi) + \text{value(right}(\tau), \xi)$

$\text{else if islet}(\tau) \text{ then value(left}(\tau), \xi) + \text{value(right}(\tau), \xi)$

$\text{else if goto}(\tau) \text{ then micro}(\pi, \text{a(sn, n+1, a(left(s, value(right(s, s), s))))})$

$\text{else if if value(proposition(s), \xi) then numb(destination(s), \pi) \text{ else n+1, \xi})) (statement(n, \pi)) (c(sn, \xi))$

This completes the description of abstract Microalgol. In order to describe a concrete Micro-algol it is only necessary to represent the abstract syntactic predicates and functions by predicates and functions on strings.

7. What about ALGOL

The semantics of Microalgol was described entirely by the two formulas of section 5. Algol is considerably more complicated, but it will be relatively easy to write down the functions once we have decided what goes into the state. The following complications arise.

Peter Landin

1. We have to be able to describe the situation in which a term is partially evaluated in order to describe the execution of a type procedure.

Pioneer of programming language theory and formal semantics.

of axioms it is possible to deduce such simple theorems as:

$$x = x + y \times 0$$

$$y < r \Rightarrow r + y \cdot q = (r - y) + y \times (1 + q)$$

The proof of the second theorem is as follows:

$$A5 \quad (r - y) + y \times (1 + q) = (r - y) + (y + y \times q)$$

$$A6 \quad r + y \times q \quad \text{provided } y < r$$

The axioms A1 to A9 define a partially ordered set of integers, an infinite set of integers in multiple dimensions. However, they are also true of the finite sets of "integers" which are manipulated by computers provided that they are confined to nonnegative numbers. Their truth is independent of the size of the set; furthermore, it is largely independent of the choice of technique applied in the event of "overflow"; for example:

(1) Strict interpretation: the result of an overflowing operation does not exist; when overflow occurs, the offending program never completes its operation. Note that in this case, the equalities of A1 to A9 are strict, in the sense that both sides exist or fail to exist together.

(2) Firm boundary: the result of an overflowing operation is taken as the maximum value represented.

(3) Modulo arithmetic: the result of an overflowing operation is computed modulo the size of the set of integers.

These three techniques—partial, firm, and modulo arithmetic—illustrated in Table II, have been implemented and multiple model in which 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648,