Peter Landin Semantics Seminar 2020
Algol 60 @ 60 :
More on Semantics

Troy Kaighin Astarte
Newcastle University
troy.astarte@ncl.ac.uk
http://homepages.cs.ncl.ac.uk/troy.astarte/

LEVERHULME TRUST

I will mention briefly some general history of 1960s formal description; discuss 4.5 other A60 descriptions giving some brief history and some technical flavour; and consider briefly why ALGOL was considered interesting in the past, and why we should think so now
Mention PhD work—especially 'Four formal descriptions'

1960s: new thinking about programming

- IPL & LISP brought abstraction
- Strachey's 1963 summer school: *non-numerical* computation
- "This gives rise to a rather vague feeling of unease, and though we think we know what we mean about [certain language constructs] we are not altogether happy that we have really got to the bottom of the concepts involved."
- IAL (ALGOL 58) heralded excitement for descriptions
- *Formal Language Description Languages*: near Vienna, 1964

McCarthy, inspired by Newell, Simon, Shaw, trying to find a way to express concepts from AI, quite different from numerical applications; although note that S-expression form of Lisp was "included to impress logicians" and a more FORTRAN-like syntax was planned
Non-numeric: argument between Strachey & Fox
Unease from Strachey when working on CPL; McC wanted maths theory of comp
IAL (algebraic->algorithmic; see Durnova & Alberts) formal syntax and promised formal semantics
FLDL: meeting of implementers vs designers; theory vs. Practice

McCarthy's 'micro-ALGOL' (1964)

- McCarthy working on mathematical theory of computation
- Core: understanding programming languages and determining their correctness
- LISP-inspired functions for abstract syntax and semantics
- State vector + statements as functions to modify same

John McCarthy, 1960s

MTOC: like Kepler's laws of planetary motion derivable from Newton, what are basic principles of computation and what can we derive?
Influence of having been working on LISP shines through McC's semantics work
Photo: it's a 7090, probably puts the photo in the 60s

McCarthy: semantics

$$micro : \Pi \times \Sigma \times \mathbb{N} \rightarrow \Sigma$$

Σ : (state vector)

Program x state x statement number -> state
Abstract conception of machine (state) and interpretation function -> operational
Presented as a function w lambda terms -> deno
Small and neat definition, but didn’t cover much

VDL operational description (1968)

- IBM Vienna takes on PL/I language description in 1964
- Zemanek wants to demonstrate VDL (ULD-IIIvII) technique on smaller language
- ALGOL 60 description authored by logician Peter E. Lauer
- Definition by an “abstract machine” with large state

Core idea: a big abstract machine, with states corresponding to program states, and statements alter this machine. Gave operational a bad name! (Spot the Landin influence!)
[error/undefined: a keyword that if encountered indicates some kind of error, usually textually explained; no method given for recovery]

Staff of IBM Vienna Lab, 1964

The VAB team, around 1964.
From left to right: (standing) Peter Lucas, George Leser, Viktor Kudielka, Kurt Walk; (seated) Ernst Rothauser, Kurt Bandat, Heinz Zemanek, Norbert Teufelhart. Missing Bekic

VDL: semantics

$$\textit{int-program} : \textit{abstract-program} \times \Xi \rightarrow \Xi\text{-set}$$

$$\Xi : (\underline{DN}, \underline{E}, \underline{D}, \underline{UN}, \underline{C}, \underline{CI})$$

DN: denotation directory; E: environment; D: dump; (spot PJL influence!) UN: unique name counter
Parallelism: C and CI are trees of potential executions
Big method, powerful, but awkward to use

Exit operational description (1972)

- Cliff Jones in Vienna, working on using formal definition in language design with Lucas
- Alternative jump handling: *exit* mechanism (Jones & Henhapl, 1970)
- Small state components passed between interpretation functions + copy rule
- ALGOL 60 definition authored by Dave Allen, Dave Chapman, & Cliff Jones

Cliff Jones, 1986

on assignment in 1968; returned 1970
Difficult lemma, proving which parts of state remain unchanged: state too big!
Smaller state, and “jumps shouldn’t take the machine by surprise”
Passing about [Abn], almost always null, unless in a jump, in which case the correct statement is found (a little clunky to check in every interpretation…)
[error cases for undefined—not handled]

Hursley functional: semantics

$$int\text{-}program : abstract\text{-}program \times \Sigma \rightarrow \Sigma$$

$$\Sigma : (vl, dn, [Abn])$$

Vl = value list (like a state vector), dn = denotation directory, Abn: contains labels in case of a jump, empty otherwise
Printed with large gaps so you can line it up with the ALGOL report

Oxford denotational description (1974)

- ‘Mathematical semantics’ from Strachey’s ideas, with underpinning from Scott
- Smaller state, greater abstraction than Landin & Allen, Chapman, Jones.
- ALGOL 60 definition authored by Peter D. Mosses during PhD with formal metalanguage

Christopher Strachey, 1964

Rough history: Strachey interested in PLs while running a consultancy in early 60s with PJL, then working on CPL; wanted to use functions as a base for modelling computation. Untyped LC and *Y* combinator (from PJL) before meeting Scott in Vienna and the logician providing a basis
“Shorter and less algorithmic”
Mosses’ thesis (1975) on a Semantics Implementation System: feed it a definition and it gives you a compiler…
Photo: at FLDL; apologies to PDM for no contemporary photograph!
[undefined, etc: each domain has an “error found element” ‘?’ Which is incomparable except with top and bottom]

Mosses: semantics

$$compiler : Prog \to U \to C \to C$$

$Prog$: deduction tree

$$U : I \to Den$$

$$C : S \to S$$

Deduction tree like abstract syntax; U for environments; C for continuations; S for states, complicated by locations for blocks/procs

VDM denotational description (1978)

- New IBM "Future Systems" in early 70s: Vienna to write a PL/I compiler
- Definition in 1974, denotational approach with exit mechanism (*tixe* combinator)
- FS killed, but Jones & Bjørner salvaged 'VDM'
- ALGOL 60 definition authored by Cliff Jones & Wolfgang Henhapl (republished 1982)
- Aim: equal abstraction to Mosses, but more readable

Dines Bjørner, early 1980s

Jones back in 1973, joined by Bjørner
Jones heard S lecture; Bekic had been with Landin at QMU late 60s
[context conditions help with type checking; reserved "error" word results from dynamic mismatches)

VDM: semantics

$$M : D \to Env \to \Sigma \to (\Sigma \times [Abn])$$

$$D : \text{abstract program part}$$

$$Env : Id \mapsto Den$$

$$\Sigma : Scalarloc \mapsto [ScalarVal]$$

Combinator ; defined as composition except when Abn present, in which case skip second part until a tixe in block can find it

Fascination with ALGOL 60

- “a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all its successors.” — Tony Hoare
- Became seen as “European”: mathematical, precise, elegant… inefficient!
- A benchmark for machines, research groups, definers
- Influential: Jovial, Alcor, NELIAC, ALGOL-W (Pascal), CPL, Simula
- CACM’s algorithms section used ALGOL 60 (only one PL/I!)

European source: David Nofre (although it really was equally American)

If your formalism works with A60, it probably works with anything! As BTD pointed out, many many compilers (even much later than language’s shelf-life)

Why ALGOL?

- Many features: nested phrases; jumps; recursion; ‘own’ variables; by name…
- Deliberately general
- Machine independent: the document became the definition
- Reification of programming *languages*
- Formal specs legitimised language study
- “ALGOL-like” as a watchword

Fraser Duncan, 1964

Features links back to benchmark; some (‘statement, declaration, type, block’) gained their popularity thanks to ALGOL effort
General: see article by Alberts, Daylight: Amsterdam in particular argued for lack of arbitrary restrictions
No machine to fall back on: document better be right! Enables formalism.
PLs became an object of study: Priestley calls it paradigmatic (note: IAL described as a ‘language’—Priestley, Nofre, Alberts wrote about language metaphor)
EWD: formalisation provided an academic impetus to study programming languages: not just means to end!
ALGOL-like: i.e. regular grammar; or has blocks, procs, and recursion;
Fraser Duncan there from FLDL: in his after dinner speech he mentioned that the phrase ALGOL-like had come to mean so much during the FLDL conf, the only thing everyone could agree on was that ALGOL was not an ALGOL-like language!

Away from ALGOL

- Initial industrial support (Bull, Elliott Bros., IBM) waned
  - Inertia? Not a product? Too much research!
- ALGOL 68 fiasco
- ALGOL 60 as a turning point away from machines and towards programs, “software engineering”
- ALGOL is much studied (see *Annals* special issue [36, 2014]) but there’s plenty more!

Bull is one counter-example: the company supported it for a time in the 1960s (see Mounier-Kuhn). Nofre makes point about research
60s and 70s pre-unbundling: so if no ALGOL compiler came with your computer, you had to write it yourself, or get it from someone who had
ALGOL 68: not going to get into it! But it scared WGs away from committee-work for a long time and led to demise of IFIP products
ALGOL part of trend away from machine specifics towards greater utility (at first maths) and ultimately towards individual programs, projects

- Troy K. Astarte. Formalising Meaning: a History of Programming Language Semantics. PhD thesis, Newcastle University, June 2019.
- Troy K. Astarte and Cliff B. Jones. Formal semantics of ALGOL 60: Four descriptions in their historical context. In Liesbeth De Mol and Giuseppe Primiero, editors, *Reflections on Programming Systems - Historical and Philosophical Aspects*, pages 71–141. Springer Philosophical Studies Series, 2018.
- Cliff B. Jones and Troy K. Astarte. An exegesis of four formal descriptions of ALGOL 60. Technical Report CS-TR-1498, Newcastle University School of Computer Science, September 2016.
- Troy K. Astarte. The history of programming language semantics: an overview. Technical Report CS-TR-1533, Newcastle University School of Computer Science, June 2020.

References to my work on this subject.

- John McCarthy. A formal description of a subset of ALGOL. In T. B. Steel, editor, *Formal Language Description Languages for Computer Programming*. North-Holland, 1966.
- Peter E. Lauer. Formal definition of ALGOL 60. Technical Report 25.088, IBM Laboratory Vienna, December 1968.
- C. D. Allen, D. N. Chapman, and C. B. Jones. A formal definition of ALGOL 60. Technical Report 12.105, IBM Laboratory Hursley, August 1972.
- Peter Mosses. The mathematical semantics of ALGOL 60. Technical report, Programming Research Group, January 1974.
- Wolfgang Henhapl and Cliff B. Jones. A formal definition of ALGOL 60 as described in the 1975 modified report. In D. Bjørner and Cliff B. Jones, editors, *The Vienna Development Method: The Meta-Language*, LNCS 61, pages 305–336. Springer-Verlag, 1978.
- Wolfgang Henhapl and Cliff B. Jones. ALGOL 60. In Dines Bjørner and Cliff B. Jones, editors, *Formal Specification and Software Development*, chapter 6, pages 141–174. Prentice Hall International, 1982.

- All available at http://homepages.cs.ncl.ac.uk/cliff.jones/semantics-library/

References for the ALGOL descriptions discussed herein.

- Helena Durnova and Gerard Alberts. Was Algol 60 the first algorithmic language? *IEEE Annals of the History of Computing*, 36(4):104, 2014.
- John McCarthy. Towards a mathematical science of computation. In *IFIP Congress*, volume 62, pages 21–28, 1963.
- C. Strachey. Towards a formal semantics. In T. N. Steel, editor, *Formal Language Description Languages for Computer Programming*. North-Holland, 1966.
- Leslie Fox, editor. *Advances in Programming and Non-Numerical Computation*. Pergamon, 1966.
- IFIP. Working Conference Vienna 1964 Formal Language Description Languages. Program. Christopher Strachey Collection, Bodleian Library, Oxford. Box 287, E.39, February 1964.
- Isaac L. Auerbach. IFIP—the early years: 1960–1971. In Heinz Zemanek, editor, *A Quarter Century of IFIP*, pages 71–94, 1986.
- Peter David Mosses. Mathematical semantics and compiler generation. PhD thesis, University of Oxford, April 1975.
- Christopher P. Wadsworth. Letter to Christopher Strachey. Christopher Strachey Collection, Bodleian Library, Oxford. Box 302, J.44. 26 March 1974.

General references, part 1.

- C. Strachey and M. V. Wilkes. Some proposals for improving the efficiency of ALGOL 60. *CACM*, 4(11):488–491, November 1961.
- C.A.R. Hoare. Hints on programming language design. Technical Report STAN-CS-73-403, Stanford University, Stanford, CA, USA, 1973.
- Mark Priestley. *A Science of Operations: Machines, Logic and the Invention of Programming*. Springer Science & Business Media, 2011.
- David Nofre, Mark Priestley, and Gerard Alberts. When technology became language: The origins of the linguistic conception of computer programming, 1950–1960. *Technology and Culture*, 55(1):40–75, 1 2014.
- David Nofre. Unraveling ALGOL: US, Europe, and the creation of a programming language. *IEEE Annals of the History of Computing*, 32(2):58–68, 2010.
- Gerard Alberts and Edgar G. Daylight. Universality versus locality: The Amsterdam style of Algol implementation. *IEEE Annals of the History of Computing*, 36(4):52–63, 2014.
- Pierre Mounier-Kuhn. ALGOL in France: from universal project to embedded culture. *IEEE Annals of the History of Computing*, 36(4):6–25, 2014.
- David Nofre. The Politics of Early Programming Languages: IBM and the Algol Project. *Historical Studies in the Natural Sciences* 51(3), 2021 (forthcoming).
- Gerard Alberts, editor. ALGOL Culture and Programming Styles, volume 36 of *Annals of the History of Computing*. IEEE, 2014.

General references, part 2.