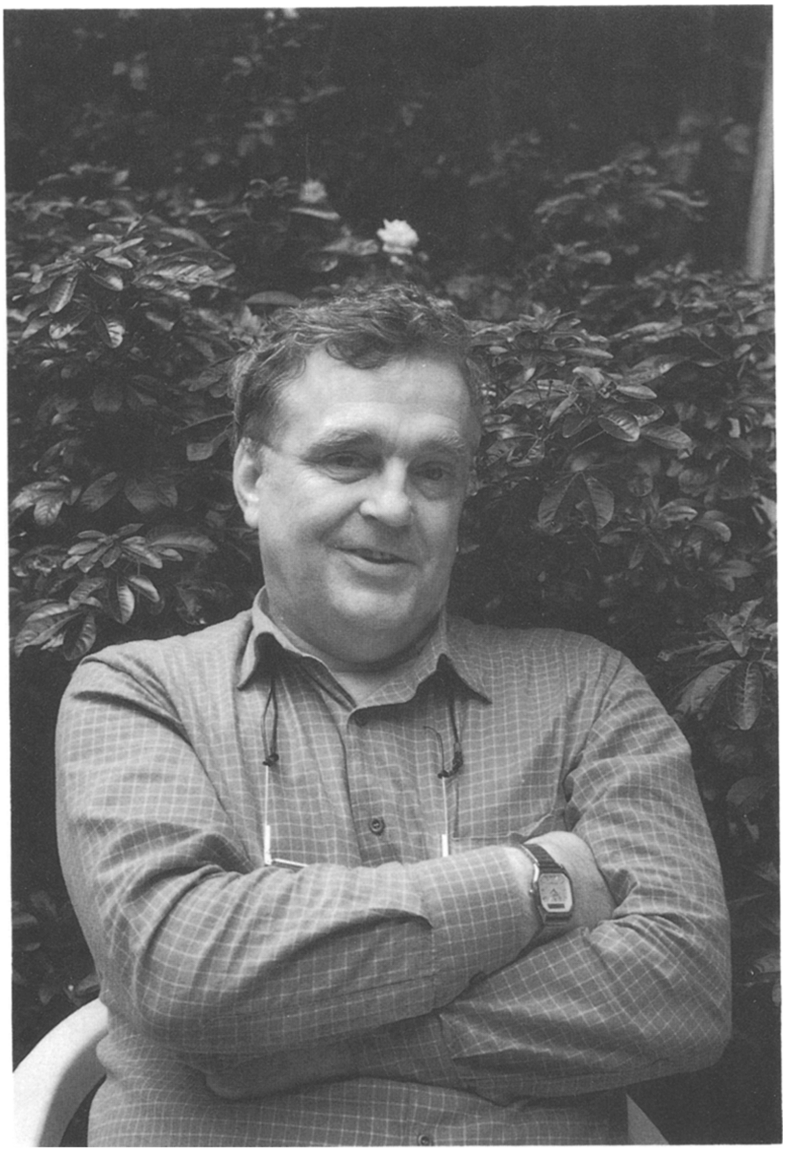

*David Michael Ritchie Park (1935–1990)*

Theoretical Computer Science 133 (1994) 187–200
Elsevier

# Obituary

# David Michael Ritchie Park (1935–1990) in memoriam

**Obituary – by Michael Paterson**

Friends and colleagues of David Park were shocked and saddened to learn of his death on 29th September, 1990, at the age of 55. His death was unexpected, though he had been seriously ill since the early summer.

David was a pioneer in several areas of computer science, and indeed he could reasonably be considered the senior theoretical computer scientist in Britain. After an undergraduate degree at Oxford he went to MIT and obtained his Ph.D. for work in model theory under Hartley Rogers. The late 1950s was the period when John McCarthy was developing his "theory of computation", a key feature of which was his novel list-processing language, LISP. David became involved with this project at an early stage and was one of the authors of the language LISP1. David became involved with this project at an early stage and was one of the authors of the language LISP1. His work at MIT and later, on program schemas and fixpoint theory, made important contributions to the early theory of computer science.

In 1964, he returned to the UK, first to the Mathematical Laboratory in Cambridge for a short time and then to join Christopher Strachey in the new Programming Research Group at Oxford. Moving to Warwick in 1968, David was one of the earliest members of the Computer Science Department and put much of his professional work into helping to build it up to its present position.

His inaugural lecture in 1983 focussed on the proper aims of computer science as a science and as an academic discipline. It contained a perceptive account of the main evolutionary threads of the subject and warned strongly against letting "ephemeral" knowledge, however attractive and challenging, displace the "essential" intellectual foundations of computer science.

Internationally he was probably best known for his work on program schemas, fairness and bisimulation. Program schemas made a clear separation between the control structure of a programming language and the operations performed on data. This formulation allowed the investigation of purely structural program transformations and provided a framework within which the descriptive power of different control structures could be compared.

David was one of the first to identify the importance of the notion of "fairness" in nondeterministic parallelism, and the need to give it a precise definition. How can we best describe and reason about systems in which some computation is performed "fairly", so that any process continually attempting to perform its computation is guaranteed to succeed eventually, but with no promise as to how long it may have to wait?

His elegant notion of "bisimulation" was just what was needed to clarify a thorny point in the semantics of concurrency. Two systems bisimulate each other if the effect of a step of one can be reproduced by a sequence of one or more steps of the other. The theory of bisimulation can be seen as an application of his earlier work on fixpoint induction.

In each of these areas, David Park's original ideas inspired new fields of research, and gained him an international reputation for his deep insights. However his clear and well-enunciated views on a variety of more practical and socio-cultural issues provided constant stimulation to his colleagues.

He recognised early the power and convenience of electronic mail and used it extensively in discussion and correspondence with colleagues and students. He believed intensely that this medium should not only transform teaching methods, but should also in many cases replace the committee (which inconveniently requires a lot of people to be in the same place at the same time) as a means of democratic discussion and decision-making. The students he tutored were delighted to engage in lengthy dialogues on a variety of subjects, including home-brewing, definitional problems in economics and the poetry of Gerard Manley Hopkins.

David was for a while chairman of the Computer Science Department at Warwick University, but he clearly did not enjoy administrative roles. Professionally, he was happiest when deeply engaged in his research or debating issues of scientific or educational importance.

In his nonacademic life he was keenly involved in politics, in music (both as a skilled player and as a computer specialist), and in a wealth of other interests. He will be sorely missed by his great number of friends and colleagues in many countries: for his unassuming manner and deep intellect, for his supportive friendship, for his intellectual honesty and for his warmth of spirit.

## Tribute from Jaco de Bakker

I first met David Park in September 1969 during a meeting of the IFIP Working Group 2.2 – then entitled Formal Language Description Languages – organized at Essex University by John Laski. 1969 was an important year for programming semantics and logic, and several ideas which turned out to be of lasting importance were reported at an early stage to the WG membership. In its April 1969 meeting in Vienna, Tony Hoare gave one of the first presentations on what was later to be called Hoare logic. The September meeting witnessed a remarkable confluence of ideas on fixed point theory as a means to understand recursion and program correctness. A precursor had been McCarthy's technique of recursion induction, the precise mathematical status of which remained unclear till the summer of 1969. As is so often observed in the history of scientific discoveries, the scene was set for a breakthrough. David, though present at the meeting, did not give a talk himself. However, he told me – and no doubt others as well – about his use of the Knaster–Tarski fixed point theorem in understanding recursion, more specifically in formulating the principle of fixed point induction in proving program properties. These ideas were soon to be published in his influential paper in Machine Intelligence 5. Independently, the late Hans Bekič had also realized the importance of the Knaster–Tarski result; he did give a talk to the Working Group (using these insights to better understand the work by Zohar Manna). Moreover, the result which was later baptized by some authors as Bekič' theorem was also announced there. Dana Scott and myself had worked as well that summer on the theory of recursion. My modest role was to state various open points in program semantics – such as an axiomatization of while statement equivalence – which were then quickly dealt with by Dana, often by an appeal to extant mathematical knowledge. In a number of letters with an abundance of ideas, seminal notions from denotational semantics made their first appearance, including the role of continuity, the induction role of the mu-calculus, as well as the relevance of the Knaster–Tarski theorem, and, independently, Bekič' theorem.

Though not the sole originator of what may be called the early fixed point theory, David was certainly one of the main actors in this field. Soon Dana continued with the development of his lambda calculus models, and another fundamental result was then contributed by David, viz. the theorem stating the equality of Curry's Y-combinator and the least fixed point combinator. David impressed me greatly by his ability to very quickly assimilate the exciting new methods in semantics, adding major insights of his own. One further example must suffice here.

During the first ICALP meeting, organized by Maurice Nivat in 1972, David brought his student Peter Hitchcock, who talked on "Induction Rules and Termination Proofs". This paper introduced the intriguing notion of program derivative as a tool to analyze termination properties. Though I have struggled in more than one subsequent publication to obtain a full understanding of this, I am still not sure that I have fully grasped this somewhat elusive concept.

Since the early seventies, I have kept regular contact with David. He was the external examiner for my first Ph.D. student (Willem Paul de Roever). We continued to share interests in some arduous problems in the theory of semantics – fairness, data flow, and others. In 1982 David gave a well-received set of lectures on data flow in a course which Jan van Leeuwen and I organized in Amsterdam. After that, our contacts got less frequent, a chance meeting in Warwick in stormy January 1990 – during an ICALP PC meeting – being the final one. Somewhat to my surprise, he talked about his retirement days then, mentioning that he looked forward to spend more time in France.

We now know that only a very brief time was allotted to him. I shall remember David with admiration and affection, for his profound contributions to our field, and for his warm and humane personality.

## Tribute – from Robin Milner

I owe much to David Park, both for his friendship and for his exacting standard for what a theory should provide. In this context I want to focus on a technical matter which concerned us both; but I cannot resist first offering my lasting gratitude to him for something which is, after all, closely connected to these technical interactions. I came later than he did to the rigours of mathematical models. I got to know him in the late sixties when I was new to them, and fairly ignorant; I shall always remember how he mixed guidance and encouragement with our friendship.

In the early eighties David came to Edinburgh for his sabbatical leave from Warwick, and lived in the flat at the top of my house. He walked about the house a lot, often frowning at some article which he was getting to the bottom of. He came into the kitchen one day while I was washing dishes; I saw with apprehension that today's victim was my 1980 monograph on CCS. "Something's wrong ..." he began.

To keep the story short, he had found the ungainly way I had defined *observation equivalence*, approximating it from above by an omega-chain of decreasing equivalences. This was worth a frown or two. The omega-chain iterates a monotone function $F$ on relations, beginning with the universal relation. I knew the limit wasn't a fixed point of $F$, but I had thought I needed the inductive proof technique which comes along with omega-chains. He showed me what I should have known, if I had digested properly his earlier work on maximal fixed points: that by taking $F$'s maximal fixed point and using its characterisation as the union of a family of suitable relations, not only is the maths elegant but also the practical proof technique (for showing real systems equivalent) is a lot better.

The latter is the striking point. Within twenty-four hours we had seen the practical gain with great satisfaction, and (after hunting through thesauri) called these suitable relations *bisimulations*. Now they are widely used, often by people who know real systems but struggle with notions like fixed point, or even induction. This was a resounding success for David's conceptual insight; it is typical of him that the idea is hard to find in his writing, being half hidden in the paper called "Concurrency and Automata on Infinite Sequences".

## Tribute from Maurice Nivat

I believe I first met David Park in 1969 at a meeting of the IFIP Working Group 2.2, in Colchester. Indeed I met simultaneously there several people and also a fascinating field of research which was then in the making, "Formal semantics of programming languages". There were given presentations of the main ideas which were to give rise to what we call today denotational, operational and fixpoint semantics and a first contruction of D-infinity, the very surprising computation domain which is isomorphic to the space of continuous functions from itself into itself. Maybe never again did I happen to discover so many new and fruitful ideas in such a small amount of time. Of course, I was impressed, both by the ideas and by the people who were there and knew so many things I ignored. Trying to talk to several participants I found David very kind and open, easy to talk to, ready to spend time explaining to a newcomer both the technical details and the intuitive motivation leading to the beautiful mathematical contructions which made me so enthusiastic.

It is thanks to him that I understood the notion of fixed point and discovered that the context-free languages can also be seen as least fixed points of mappings attached to the grammars generating them (formally this was known but the meaning was not so clear, at least to me!).

Shortly after this Colchester meeting I was asked to form and head a team of research in the newly created "Institut de Recherche en Informatique and Automatique" (INRIA) and though I knew extremely little about the subject I proposed a team on "Verification and validation of programs" which was to last around ten years and was the place where a number of well-known french young scientists started doing research (some of them after getting a Ph.D. in the United States). We obviously badly needed help and fortunately found that help, mainly in the United Kingdom, and David Park played a crucial role: many visits of British researchers to INRIA and many stays of French researchers in Great Britain were made possible by an agreement between INRIA and the Universities of Warwick and Edinburgh (funded by INRIA and the Science Research Council) which was very friendly negotiated with David Park and Robin Milner. Thanks to this agreement David Park became a very popular figure frequently seen at INRIA and we all learned a lot from him: he was marvellous in this role saying little but always deep and precise things and letting us discover what was not said, always encouraging and only very fruitfully critical. I believe all those who shared that experience with me keep the same memory of a sort of holy period where we were really building something or at least had that feeling, thanks in particular to the generosity of British colleagues, among whom in the first place David who liked to let us believe that they knew less than they did and that we knew more than we did.

Another thing which I should mention is the very inspiring effect that David Park's work on "fairness" had on me. In this work were considered, maybe for the first time, "infinite computations", not as an abstract extension of finite ones but as one way of describing real properties of actual systems. All my own later work on infinite words, infinite trees and infinite computations stems from ideas which were first exposed in David's maybe most well-known paper "Concurrency and automata on infinite sequences" and which I had on many occasions to discuss with him.

Discussing with David was always a real pleasure, for he was when discussing on scientific matters exceptionally kind with his interlocutor and also because one could not discuss with him only on scientific matters. David's deep interest in all aspects of life, in philosophy and sociology, education, literature and art, even wine, made him a wonderful fellow to talk to: he had one of the dearest qualities, that after talking with David one would feel more secure, more confident and more intelligent.

## Tribute from Bill Wadge

I first met David Park in the Autumn of 1974, when I arrived at the University of Warwick to take up a position as a researcher on the SERC grant he held at the time with Mike Paterson. This was only my second academic job. My first had given me a rather jarring introduction to the North American academic rat race, and I decided to try my luck abroad. I ended up at Warwick almost by accident – it was the first opportunity that came along. In fact, I could not have made a better choice if I had searched for months.

Working in the Department with David in those days was like attending a nonterminating workshop in semantics and the theory of programming. Bob Constable was there for the year from Cornell, working on the (apparently) half-baked idea of extracting programs from constructive proofs. Mike Smyth arrived at the same time, to join David in his long term goal of finding denotational semantics for various kinds of nondeterminism. Adi Shamir and Danny Lehman also spent several months with us. And I recall bright, noctural graduate student named David May.

There was also a steady stream of visitors through the Department: Willem de Roever, Mike Machty, Robin Milner, Gordon Plotkin, Gilles Kahn, Joe Stoy, and many others. David Park was right at the centre of this group. They came to see him and each other, and in some way he acted as a clearing-house for all the different ideas about program semantics and verification. And there was a very good reason why he played so central a role: because all these different ideas were based on the notion of least fixed point, a concept I will always associate with David.

David and I argued amiably about everything, from politics to programming. He was initially very sceptical about the Lucid project, while for my part I accused denotational semantics of providing a fig-leaf of mathematical respectability for languages doomed to end up in the dustbin of history. My attitude to denotational semantics changed during a session with Ed Ashcroft, who pointed out that Lucid streams formed a domain over which all the Lucid operators were continuous; arbitrary combinations therefore have least fixed points. This simple observation opened up a whole range of possibilities – such as nonpointwise filters or branching time iteration – that made no sense in terms of the simple-minded operational models I'd been using up to that point. David himself was intrigued by these developments. He eventually used streams in his own work on nondeterminism, and even popularized "hiatons" (a word Ed and I coined).

We did collaborate on one rewarding venture: the WAFL language. WAFL originated in a visit by David Turner, who gave a seminar on early work with SASL. At the time SASL had infinite lists, but only circular ones, implemented with cyclic pointer chains. After the talk, over tea, David and I suggested that one could implement arbitrary infinite lists using Landin's closures. Turner thought for a moment and agreed – though I suspect he was already working on it. At any rate David and I decided not to wait and enlisted a grad student to write an infinitary-lisp interpreter in BCPL. This took less than a month, and I still recall the thrill of seeing it compute factorials without recursion, using the textbook definition of the Y-combinator. David took charge of the code, roled up his sleeves, and spent weeks hacking improvements, such as replacing tail recursions by jumps. We both used WAFL in functional programming and semantics courses, and David produced a series of WAFL notes which presented denotational semantics (especially, the role of $\perp$) from a programmer's point of view.

My time at Warwick was a wonderful and truly educational experience – I learned so much from David and his whole group. I vividly recall one incident which demonstrates just how much I owe them. I was driving down to Oxford with David and Willem de Roever, listening to them discuss possible approaches to the relational semantics of fair merge (a perennial topic). I interrupted to ask about a fine point of terminology: what was the difference between operational and denotational semantics? (For those not in theory, this is like asking a mathematician about the difference between integration and differentiation!)

Those times were wonderful in another way – in the relaxed and collegial atmosphere. For example, my silly question in the car could have been greeted with knowing looks and gales of derisive laughter. But it wasn't. Instead, they patiently explained the crucial distinction to me, and gave not the slightest hint that they resented the distraction.

I think David deserves a lot of credit for the healthy atmosphere of those days. As far as I could tell, he was motivated almost exclusively by a desire to improve programming languages and our understanding of them. I don't think I ever heard him discuss research in careerist terms, or boast about the size of his grants and the length of his CV. He was a researcher because he simply loved it.

I'll always be grateful to David Park for teaching me (and, later, my students) so much about the mathematical theory of programming languages, and about research in general. And I will always remember his basically selfless dedication to the advancement of science.

## Tribute from Peter Welch

From 1970 to 1972, I was one of David's small number of research students. Although this seems a ridiculously short span of time some twenty years later, those years were so rich and exciting that they remain the major force behind my present life.

David was brilliant to talk to – whether sitting on the floor of his office in a crowded undergraduate session arguing about some kind of induction proof, or in a more relaxed way (i.e. I could grab a chair) hearing about the magical ultrapowers of ultraproducts of ultrafilters, or just sitting over a pint worrying about broken relationships with girl-friends. The ability David possessed was to simplify whatever problem was causing you trouble to its essentials. If that still left you staring at a brick wall, the really clever thing he did was to come up with an apparently different problem for you to solve – that made you adopt a whole new line of reasoning – and that unexpectedly (to you!) solved your original problem as a simple corollary.

David liked to think on his feet – literally. When he was really deep into explaining something, he would regularly alarm his listeners with his carefree disregard for chairs, desks and even walls as he paced up and down with ever-increasing energy. Lecture rooms were the most dangerous, as their width enabled David to build up frightening velocities.

I remember one walk especially, in a park in Leamington Spa. It was towards the end of my formal registration as a research student and I still had not proved the main theorem for my thesis. I had constructed a D-infinity lattice of approximate normal-forms of the lambda-calculus and needed to show that it modelled beta-reductions. I had been worrying at this for over a year so it was clearly time for serious walk!

We took David's enormously athletic Doberman (I think) called Rudi (I think), unleashed him when we got through the park gates and, then, I tried to keep up with both of them. Now, mapping an arbitrary lambda-expression on to the normal-form lattice produced a set of coordinates that reflected a strange reduction sequence which was very different from classical normal-order. To prove my model, it was sufficient to show that this reduction sequence went "sufficiently far" – but, so far, I had failed. When I caught up with David, he had abstracted the particular nature of the reduction sequence coming out from the lattice into a more general notion that we christened "inside-out" reductions. He had also deduced that I only had to show that one of those went sufficiently far and the lattice-sequence would be bound to follow. The rest of the walk was spent excitedly failing to find any counter-examples to the new conjecture. Somehow the traffic missed us and we returned to David's home safely.

About a month later, David's generalisation had given me sufficient elbow room to wield the logical sledgehammer I had been trying unsuccessfully to use before and the result was proved. I was not particularly happy with the proof – and David less so – but it got me my Ph.D.!

On a visit to IRIA (now INRIA) around this same time, David talked about the inside-out conjecture (as it then was). Jean-Jacques Levy discovered that its proof would establish a vital, but so far missing, property ("substitutivity") for a model of the lambda-calculus he had been developing. It turned out that his model and the normal-form lattice were essentially the same, even though they had very different constructions. Jean-Jacques' creation gave a model of beta-equivalence straight from its definition, but substitutivity was not obvious; the normal-form lattice was immediately substitutive, but beta-modelling was hard to prove; the "completeness" of inside-out reductions unified them.

Jean-Jacques, not knowing about my sledgehammer, proceeded to devise "labels" to keep track of lambda-expressions during reduction and very quickly came up with a whole series of short and very elegant proofs not only for the inside-out theorem, but for the classical ones as well (Church-Rosser and normal-order). David was a little worried by this and apologised to me for provoking such elegance when my own work was only just finished, unpublished and somewhat ponderous. David was very concerned about good behaviour between academics and in protecting his research students. But, of course, David had done the right thing and, in any case, I was over-the-moon with Jean-Jacques' work and very glad to drop that sledgehammer!

I'm now working on the theory and application of parallel computing for high-performance and safety-critical systems. A major tool is the CSP/CCS-derived language-and-logic called occam, named after the XIVth century philosopher famous for nagging his contemporaries to get rid of needlessly complicated ideas (Occam's Razor). William of Occam seems a very familar figure! David's approach to life and its problems remain with me – I greatly value his friendship and that of Joanna and the rest of his family and everything he taught me.

## Bibliography – by Steve Matthews

The bibliography below contains a list with commentary of David Parks's publications, unpublished reports, and other significant papers, all recovered from his Warwick office. In order to assist future researchers this material is to be deposited with the British National Archive for Computing in Manchester.

David's academic career can best be described in his own words [1976a]. "My research has been dominated by two preoccupations, originally rather separate: (1) the design and implementation of programming languages, together with the notions, practical and abstract, on which designs are based; (2) abstract mathematics, particularly mathematical logic and associated algebraic theories." The "suspicion that interesting associations could be made between these two areas" fuelled a history of research from program schemas through formalisms for proofs, and on into semantics; a history we have all undoubtedly profited from.

David [1986] records his career beginning with compulsory national service from 1953 to 1955, serving as a ground radar fitter in Britain's Royal Air Force. In 1957 at the age of 22 he worked as a programmer for Ferranti Ltd., programming the Pegasus computer. This very practical experience appears to have been in tandem with his studies for a first degree in Mathematics from Oxford University. Even at this early stage we see developing twin interests in practical computing and mathematics which would characterise his whole career. From 1958 to 1964 David worked as a teaching assistant in both the Mathematics Department and the Research Laboratory for Electronics of MIT, a period in which he would both work on LISP and obtain a doctorate in model theory. In 1960 he appears as contributor to the first LISP programmer's manual [1960a]. LISP1 was a programming system for computing symbolic expressions upon the IBM 704. The overall design was John McCarthy's, with the manual written largely by P. Fox. According to this manual the first LISP compiler was written by D. Edwards with David's "assistance". This hard-nosed programming experience was complemented by a Ph.D. thesis [1964a] in model theory, also from MIT. According to Wilfred Hodges (Queen Mary & Westfield College, London) David appears to have been the first person to have proved a version of the following Neumann–Neumann Lemma. Let $p$, $m$ be nonnegative integers and $\{A_i\colon i\in I\}$ a set of finite sets, each of which has cardinality at most $p$. Then there are at most $pm$ sets $B$ such that (1) $B$ has cardinality at most $m$, (2) $B$ meets each $A_i$ and (3) no proper subset of $B$ meets each $A_i$. Peter Neumann's Lemma can be derived from David's, a fact not appreciated until after the latter's death as his thesis was never published.

While still in USA David developed his interest in program schemas and like others began to address the foundational problems of representing programs in logical systems. Doing part time research at Bolt, Beranek & Newman Inc. of Cambridge (MA), David co-authored "The Undecidability Problem for Program Schemata" [1964b] with David Luckham. Interest in finding a positive solution to the decidability problem for proving the equivalence of "programs written in an elementary formal language" was motivated by the need to find transformation methods for simplifying programs of the 1960s. Earlier work by Yu.I. Ianov in 1960 had generated optimism, consequently Luckham and Park "surprised" themselves by proving that the problem was actually recursively undecidable after all: "the only essential difference between our program schemas and Ianov's lies in the representation of a finite number of storage locations (possible in our language but not Ianov's)".

By the time of David's return to England in 1964 to take up a research assistantship at the Cambridge Mathematical Laboratory he had established himself as an authority on unsolvability. This can be seen from his regular contributions to the American Mathematical Society Notices [1966–1967], and by his contribution of a definition for the term "Unsolvable Problems" to the Encylopedia of Linguistics, Information, and Control [1968b]. Although during 1964–1966 David worked upon the design and implementation of CPL for the Titan, his publications suggested otherwise. While at Cambridge he supervised the Ph.D. thesis of Mike Paterson, a later collaborator along with David Luckham [1970] where they argue that "some comparatively straightforward properties of programs ... cannot easily be represented as validity or satisfiability problems in a first-order language". Two years later in 1966 David moved to Oxford University to become a Senior Research Officer continuing his work on program schemas while also introducing himself to what would become the Scott-Strachey school. In 1968 David moved to the new Warwick University and its blossoming Mathematics Institute. Eventually David would, with others, establish Warwick's School (later to become a Department) of Computer Science.

David's interest in unsolvable problems in seen to mature into solutions in [1971] where, like others, he is "concerned with general strategies for proving the second-order sentences obtained by carrying out the representation in terms of convergence formulas". It is "a general form of argument by mathematical induction ... which is applicable to properties of minimal fixpoints, and applicable therefore to convergence formulas". As David claimed, fixpoint induction was "important" because it subsumes both McCarthy's recursion induction and proof methods used by Floyd. In [1976b] the ideas of fixpoint induction are seen to mature into work on the mu-calculus. While accepting first-order logic to be inadequate for reasoning about programs, David adds the minimal fixpoint operator mu. "This paper will make rigorous the intuition that the mu-calculus is indeed strictly intermediate in expressive power between first- and second-order logics".

Into the 1970s David admits, like many, being "much stimulated" by the work of Dana Scott. A nice example of his efforts in this area is the unpublished "The Y-Combinator in Scott's Lambda-Calculus Models" [1967c] in which he discusses the relationship between the paradoxical operator

$$\mathrm{Y} = \lambda x . (\lambda y . x(yy))(\lambda y . x(yy))$$

of the lambda calculus and the weaker minimal fixpoint operator

$$\mathrm{Y}^* = \lambda x . \bigsqcup_{n=0}^{\omega} x \bot,$$

a distinction as much overlooked as is his contribution to its discussion.

Such interests are clearly related to David's equally unrecognised contributions to functional programming, in particular on the emerging topic of lazy evalution. This work is now only documented in lecture notes on WAFL, the Warwick Functional Language [1981c].

If David's name had to be associated with any two words of the Computer Science vocabulary they are perhaps "fairness' and "bisimulation". His critics of the time would have unjustly labelled his interest in fairness [1983a,b] an obsession; in fact his interest was for substantial reasons, as he wrote in [1983a]: fairness "accounts for the most important respect in which the formal description of programming languages involving parallelism can be expected to be inadequate". The problem was elegantly considered in terms of nondeterministic extensions for Gilles Kahn's computing networks of 1974.

The paper on automata on infinite sequences [1981a] also treats the fairness question, since two automata may accept exactly the same finite sequences, but differ in that only one of them – the "unfair" one – will accept a certain infinite sequences, say $a^\omega$ the infinite sequence of $a$'s. But the paper is likely to be remembered mostly for introducing the notion of bisimulation, which has had a profound influence on the theory of concurrent computation – and even has applications in the classical domain theory of Scott.

Influenced personally by John McCarthy in the 1950s at MIT, David's career was quite simply one long journey in search of the essential knowledge in Computer Science. Nowhere can this be seen more clearly than in his public Inaugural Lecture [1983c] to Warwick University following accession to a personal chair in 1982. He used the occasion to campaign vigorously for the education of "Computer Specialists" to consist only of essential knowledge (e.g. structured programming and complexity theory), while the ephemeral knowledge (e.g. language syntax and system calls) should be picked up later "on the job". An innovative contribution here was to include technology-based asynchronous learning methods – such as E-mail – as part of this essential knowledge, reviving an idea which dates back to McCarthy and Strachey.

[1960a] J. McCarthy et al., *LISP 1 Programmer's Manual*, Artificial Intelligence Group, Computation Center and Research Laboratory of Electronics, MIT, Cambridge, MA.

[1963] T. Marill et al., Cyclops 1: a second generation pattern recognition system, *Proc. 1963 Fall Joint Computer Conference.*

[1964a] D.M.R. Park, Set theoretic constructions in model theory, Ph.D. Thesis MIT.

[1964b] D. Luckham and D. Park, The undecidability of the equivalence problem for program schemata, Scientific Report 1, USAF Contract AF (19-628)-3998, Bolt Beranek & Newman Inc., Cambridge, MA.

[1966] J.N. Buxton, J.C. Gray, and D. Park, *CPL Elementary Programming Manual*, University Mathematical Laboratory, Cambridge.

[1966–7] D. Park, Abstracts 66T-459, 66T-512, 67T-3, 67T-122, 67T-258, American Mathematical Society Notices.

[1968a] D. Park, Some semantics for data-structures, in: D. Michie, ed., *Machine Intelligence 5* (Edinburgh Univ. Press).

[1968b] Meacher, ed., Unsolvable problems, in: *Encylopedia of Linguistics, Information and Control* (Pergamon Press, London).

[1970] D. Park, on formalised computer programs, *J. Comput. System Sci.* **4**, 220–249.

[1971] D. Park, Fixpoint induction and proofs of program properties, in: D. Michie and B. Meltzer, eds., *Machine Intelligence 5*, (Edinburgh University Press).

[1973] P. Hitchcock and D. Park, Induction rules & termination proofs, in: ed. M. Nivat, *Automata Languages & Programming* (North-Holland, Amsterdam).

[1975] A.J. Kfoury and D. Park,.On the termination of program schemas, *Inform. and Control* 29, pp. 243–251.

[1976a] D. Park, Notes on research interests, Memorandum.

[1976b] D. Park, Finiteness is Mu-ineffable, *Theorct. Comput. Sci.* **3**, pp. 173–181.

[1976c] D. Park, The Y-combinator in Scott's lambda-calculus models (revised version), Theory of Computation Report No. 13, Dept. of Computer Science, University of Warwick.

[1979] D. Park, When are two effectively given domains identical?, in: *Proc. 4th GI Conf. on Theoretical Computer Science*, Aachen, Lecture Notes in Computer Science, Vol. 67 (Springer, Berlin).

[1980] D. Park, On the semantics of fair parallelism, in: Bjorner, ed., *Abstract Software Specifications*, Lecture Notes in Computer Science, Vol. 86, (Springer, Berlin).

[1981a] D. Park, Concurrency and automata on infinite sequences, *Proc. 5th GI Conf. on Theoretical Computer Science*, Karlsruhe, Lecture Notes in Computer Science, Vol. 104 (Springer, Berlin).

[1981b] D. Park, A predicate transformer for weak fair iteration in: *Proc. 6th IBM Symp. on Mathematical Foundations of Computer Science*, Hakone, IBM Japan.

[1981c] D.M.R. Park, Notes on functional programming, *Lecture Notes*, Dept. of Computer Science, University of Warkwick, 1981 (estimated).

[1983a] D. Park, The fairness problem in nondeterministic computing networks, in: de Bakker and van Leeuwen eds., *Foundations of Computer Science*, Mathematisch Centrum, Amsterdam.

[1983b] D. Park, Reasoning with fairness constraints, in: Karpinski, ed., *Proc. Foundations of Computation Theory*, Borgholm, Lecture Notes in Computer Science, Vol. 158 (Springer, Berlin).

[1983c] D. Park, Essential and ephemeral knowledge: the culture and education of computer specialists, Text of an Inaugural Lecture, University of Warwick, 28th November.

[1986] D.M.R. Park, Curriculum Vitae.