Boffin Access

Open Access Journal | Volume: 1.2

Advances in Computer Sciences

ISSN 2517-5718

Scott's Qualitative Fixed Point Technique in Complexity Analysis of Algorithms

Maria Lopez-Ramirez[1],
Oscar Valero[2]*

[1]Juniper Innovating Travel Technology, Gremis Fusters, 33, Balearic Islands, Spain
[2]Department of Mathematical Sciences and Informatics, University of the Balearic Islands, Ctra, Baleares, Spain

Abstract

In 1972, D.S. Scott developed a qualitative mathematical technique for modeling the meaning of recursive specifications in Deno-tational Semantics. In this paper we show that the same original Scott's technique remains helpful for Asymptotic Complexity Analy-sis of algorithms requiring really a reduced number of hypotheses and elementary arguments. Thus, we will disclose that such a qualitative approach presents a uni ed mathematical method that is useful for Asymptotic Complexity Analysis and Denotational Semantics. More-over, we will emphasize the introduced technique applying the results to provide the asymptotic complexity (upper and lower bounds) of the running time of computing of a celebrated algorithm.

2010 AMS Classification: 47H10, 54F05, 68N30, 68Q55, 68Q25, 68W40

Keywords

Partial Order; Scott; Kleene; Fixed Point; Denotational Specification; Algorithmic Complexity; Recurrence Equation

Introduction

In 1972, D.S. Scott developed a mathematical technique for modeling the meaning of recursive specifications in Denotational Semantics. The programming languages used in order to implement such specifications allow, in general, the use of recursive definitions (denotational specifications) in such a way that the meaning of the aforesaid definitions is expressed in terms of its own meaning. Nowadays, the aforementioned mathematical tool is known as fixed point induction principle. Such a principle is based on fixed point theory, the Kleene's fixed point theorem, for monotone self-mappings defined in partially ordered sets (for a detailed treatment of the topic we refer the reader to [1-4]. Concretely, Scott's induction principle states that the meaning of a recursive specification is obtained as a fixed point of a non-recursive mapping, induced by the denotational specification, which is the supremum of the successive iterations of the aforesaid non-recursive mapping acting on a distinguished element of the model. The non-recursive mapping expresses the evolution of the program execution. Besides, the partial order encodes some computational information notion so that each successive iteration of the mapping matches up with an element of the mathematical model which is greater than (or equal to) those that are associated to the preceding steps of the program execution. Of course, it is assumed that each iteration of the program computation provides more information about the meaning of the algorithm than those executed before. Therefore, the mentioned fixed point encodes the total information about the meaning provided by the elements of the increasing sequence of successive iterations and, in addition, no more information can be extracted by the fixed point than that provided by each element of such a sequence.

The Scott's fixed point principle have been applied successfully to model computational processes that arise in a natural way in two fields of Computer Science different from Denotational Semantics. Concretely, it has been applied to Asymptotic Complexity Analysis [5] in and to Logic Programming in [6]. In the sequel we focus our attention on the application yielded to Asymptotic Complexity Analysis. In 1995, M.P. Schellekens showed in that the seminal Scott idea of modeling the meaning of a denotational specification as, at the same time, the fixed point of a non-recursive mapping and, in addition, the supremum of the successive iterations sequence can be adapted to Asymptotic Complexity Analysis of algorithms. Thus, Schellekens developed a fixed point technique to get asymptotic upper bounds of the complexity of those algorithms whose running time of computing satisfies a recurrence equation of Divide and Conquer type in such a way that the Scott spirit was preserved. The Shellekens technique has a fundamental difference from Scott's one. Concretely, the fixed point principle is now based on the Banach fixed point theorem instead of Kleene's fixed point theorem. It must be pointed out that the method of Schellekens introduces a \distance" function, in fact a quasi-metric, which yields a measure of the degree of ap-proximation of the elements that make up the model and, besides, encodes at

Article Information

DOI: 10.31021/acs.20181101
Article Type: Research Article
Journal Type: Open Access
Volume: 1 Issue: 2
Manuscript ID: ACS-1-101
Publisher: Boffin Access Limited

Received Date: 05 September 2017
Accepted Date: 27 December 2017
Published Date: 08 January 2018

***Corresponding author:**

Oscar Valero
Department of Mathematical Sciences and Informatics
University of the Balearic Islands
Ctra, Baleares, Spain
E-mail: o.valero@uib.es

Citation: Ramírez ML, Valero O. Scott's Qualitative Fixed Point Technique in Complexity Analysis of Algorithms. Adv Comput Sci. 2018 Jan; 1(2):101

the same time the information partial order. Moreover, in contrast to Scott's qualitative technique, the new method provides a unique fixed point of the self-mapping as the candidate running time of computing of the algorithm (the counterpart of the meaning of the denotational specification in the Scott case). The quantitative data approach of the Schellekens technique was seen as an advantage over the Scott approach and inspired many works on the mathematical foundations of Asymptotic Complexity Analysis (see, for instance, [7-16]).

Although the preceding quoted works argue that the Schellekens quantitative approach is very appropriate for Asymptotic Complexity Analysis and illustrate its strengths, the main target of this paper is to make a plea for the original Scott fixed point principle showing its utility for determining the upper and lower asymptotic bounds for those algorithms whose running time of computing fulfills a general recurrence equations. Thus, we will show that the Scott approach remains valid for both, Denotational Semantics and Asymptotic Complexity Analysis.

The remainder of the paper is organized as follows: In Section 2 we recall brie y the basic notions from asymptotic complexity of algorithms that we will play a central role in our subsequent discussion. Section 3 is devoted to showcase the utility of Kleene's fixed point theorem, and thus the Scott fixed point approach, for obtaining the upper and lower bounds of the complexity of those algorithms whose running time satisfies a general recurrence equation. All this will be made requiring really a reduced number of hypotheses and elementary arguments. Finally, we will illustrate the developed technique applying the results to provide the asymptotic complexity (upper and lower bounds) of the running time of Quick sort.

Preliminaries

In this section we recall, on the one hand, the celebrated Kleene fixed point theorem and the pertinent notions about partial ordered spaces that are necessary in order to state it. On the other hand, we remember the mathematical basics about Asymptotic Complexity Analysis of algorithms that will play a central role in Section 3, where we will apply the Scott methodology to determine asymptotic bounds for the running time of computing.

The Kleene fixed point theorem

Following [1] a partially ordered set is a pair $(X, \leq_X)$ such X is a nonempty set X and $\leq_X$ is a binary relation on X which fulfills for all *x*, y, z € X the following:

(i) $x \leq_X x$ (reflexivity);

(ii) $x \leq_X y$ and $y \leq_X x \Rightarrow x = y$ (anti symmetry);

(iii) $x \leq_X y$ and $y \leq_X$ (transitivity).

If $(X \leq x)$ is a partially ordered set and $Y \subseteq X$, then an upper bound for Y in $(X \leq x)$ is an element $x \in X$ such that $y \leq_X x$ for all $y \in Y$. An element $z \in Y$ is least in $(Y \leq_X)$ provided that $z \leq_X x$ for all $x \in Y$. Thus, the supremum for Y in $(X; \leq_X)$, if exists, is an element $z \in X$ which is an upper bound for Y and, in addition, it is least in the set $(UB(Y); \leq_X)$, where $UB(Y) = \{u \in X : u$ is an upper bound for Y$\}$. In addition, fixed $x \in X$, the sets $\{y \in X: x \leq_X y\}$ and $\{y \in X: y \leq_X x\}$ will be denoted by $\uparrow_{\leq_X} x$ and by $\downarrow_{\leq_X} x$, respectively.

On account of [17] a partially ordered set $(X; \leq_X)$ is said to be chain complete provided that every increasing sequence has supremum. A sequence $(x_n)_{n\in N}$ is said to be increasing whenever $x_n \leq Xx_{n+1}$ for all $n \in N$, where N denotes the positive integer numbers set. Moreover, a mapping from a partially ordered set $(X; \leq_X)$ into itself is monotone if $f(x) \leq x\, f(y)$ whenever $x \leq_X y$. Furthermore, a mapping from a partially ordered set $(X; \leq_X)$ into itself is said to be $\leq_X$-continuous if the supremum of the sequence $(f(x_n))_{n\in N}$ is $f(x)$ for every increasing sequence $(x_n)_{n\in N}$ whose supremum exists and is x. Observe that every $_X$-continuous mapping is always monotone with respect to $\leq_X$, i.e., $f(x) \leq_X f(y)$ whenever $x \leq_X y$. In the following, given a mapping from a partially ordered set $(X; \leq_X)$ into itself, we will denote the set $\{x \in X: f(x) = x\}$ by $Fix(f)$.

Taking into account the above concepts, Kleene's fixed point theorem can be stated as follows (see, for instance, [17]):

Theorem 1: Let $(X; \leq x)$ be a chain complete partially ordered set and let $f: X \to X$ be an $\leq x$-continuous mapping. Then the following assertions hold:

1. If there exists $x \in X$ such that $x \leq_X f(x)$, then there exists

$x^* \in Fix(f)$ such that $x^* \in \uparrow_{\leq_X} x$

2. If there exists $y \in X$ such that $f(y) \leq_X y$ and

$y \in \uparrow_{\leq_X} x$, then $x^* \leq_X y$

Fundamentals of Asymptotic Complexity Analysis

We follow [18] as main reference for Asymptotic Complexity Analysis of algorithms.

The complexity of an algorithm is determined to be the quantity of re-sources required by the algorithm to get a solution to the problem for which it has been designed. A typical resource is the running time of computing. Usually there are often a few algorithms that are able to get the solution to a fixed problem. So one of goals is to set which of them solve the problem taken lower time. With this aim, it is needed to compare their running time of computing. This is made by means of the asymptotic analysis in which the running time of an algorithm is denoted by a function $T: N \to]0, \infty]$ in such a way that T(n) matches up with the time taken by the algorithm to solve the problem, for which it has been designed, when the input data is of size n. Notice that we exclude 0 from the range of T because an algorithm always takes an amount of time in order to solve the problem for which it has been designed.

Since the running time does not only depend on the input data size n, but it depends also on the particular input and the distribution of the input data. Hence, it is usually distinguished three possible behaviors when the running time of an algorithm is discussed. These are the so-called best case, the worst case and the average case. The best case and the worst case for an input of size n are defined by the minimum and the maximum running time of computing over all inputs of size n, respectively. The average case for an input of size n is defined by the expected value or average running time of computing over all inputs of size n.

Generally, fixed an algorithm, to state the exact expression of the function which provides its running time of computing is a hard task. For this reason, in most situations the analysis is focused on bounding the running time of computing and, thus, to yield an approximation of it. Of course, to approximate the running time of computing we can consider an upper and lower bound which are given by the so-called O and Ω asymptotic complexity classes. Next we recall such notions. To this end, from now on, $\leq$ will stand for the usual partial order on $]0, \infty]$. In the sequel, we will denote by T the set $\{f: N \to]0, \infty]$: f is monotone with respect to and unbounded. Observe that if a function f models the running time of computing of an algorithm, then f is expected to be monotone and unbounded.

Consider two functions $f, g \in T$. Then $f \in O(g)$ $(f \in \Omega(g))$ if and only if there exist $n_0 \in N$ and $c \in]0, \infty[$ satisfying $f(n) \leq cg(n)$ $(cg(n) \leq f(n))$ for all $n \in N$ with $n \geq n_0$. Moreover, the case in which $f \in O(g) \bigcap n\, \Omega(g)$ is denoted by $f \in (g)$.

Clearly if $f \in T$ describes the running time of computing of an algorithm under discussion, then the fact that $f \in O(g)$ gives that the function g represents an asymptotic upper bound of the running time. Therefore if we do not know the exact expression of the function f, then the function g provides an approximate information of the running time of computing for each input size n, f(n), in the sense that the algorithm takes a time to process the input data of size n bounded above by the value g(n). Clearly, a similar interpretation can be given when we consider $f \in (g)$.

We end the section noting that the set $\mathcal{T}$ becomes a partially ordered set when we endow it with the partial order $\leq_{\mathcal{T}}$ given by $f \leq_{\mathcal{T}} g \Leftrightarrow f(n) \leq g(n)$ for all $n \in N$.

The Scott fixed point technique applied to Asymptotic Complexity Analysis

In this section we show the utility of Kleene's fixed point theorem, and thus of the Scott fixed point technique, as a mathematical tool for the analysis of the asymptotic complexity of algorithms.

Usually the analysis of the running time of computing of algorithms leads up recurrence equations on N of the following type:

$$T(n) = \begin{cases} c_n & \text{if } n \le n_o \\ \sum_{i=1}^{k} a_i T(n-i) + d(n) & \text{if } n > n_o \end{cases} \quad (1)$$

where d ϵ T such that d(n) < ∞ for all n ϵ N, n_0, k ϵ N and c_i, a_j ϵ]0, ∞ [for all i = 1,………,n_0 and j = 1,………,k.

However, a class of recurrence equation that differs from those of type (1) and that arise in a natural way in the analysis of the running time of computing of many Divide and Conquer algorithms are the so-called multi term master recurrences (see, for instance, [19]). The aforementioned recurrence equation is given as follows:

$$T(n) = \begin{cases} c_n & \text{if } n \le n_o \\ \sum_{i=1}^{k} a_i T(\lceil n/b_i \rceil + d(n) & \text{if } n > n_o \end{cases} \quad (2)$$

where d ϵ T such that d(n) < ∞ for all n ϵ N, n_0, k ϵ N, and c_i, a_j ϵ]0, ∞ [for all i = 1,………,n_0 and j = 1,…….,k.

According to [20] the functions d in the preceding expressions are called forcing or input functions.

Examples of algorithms whose running time fulfills a recurrence equation of the above introduced types (either (1) or (2)) are the celebrated Quick sort (worst case), Merge sort (average case), Hanoi Towers Puzzle, Large two (average case) and Fibonacci (see, for instance, [5,18,20,21]). Notice that among the aforementioned algorithms there are recursive and non-recursive.

In the classical literature many techniques have been developed for obtaining the asymptotic bounds of those algorithms whose running time of computing satisfies the preceding recurrence equations. In general, such techniques are specific for each case under study and involve tedious and hard arguments either from mathematical induction or from calculus involving integrals or limits. A general view of the classical treatment of the topic can be found in [18,22].

In what follows we develop a method based on the Kleene fixed point theorem which allows to determine asymptotic bounds in those cases in which the running time of computing satisfies a general recurrence equation which retrieves as a particular case the recurrence equations of type (1) and (2). The new method does not intend to compete with the standard techniques to analyze the complexity of algorithms based on the classical arguments.

The authentic purpose of the novel method is to introduce a formal treatment of asymptotic complexity by means of really basic and elementary arguments which provide, in some sense, a fixed point theoretical counterpart of the classical techniques in the same way that Scott's fixed point theorem made in Denotational Semantics (see [2-4]). Besides the new method presents the advantage, on the one hand, of avoiding to assume a "convergence condition" in the spirit of Schellekens for all functions involved, that is $\sum_{n=1}^{\infty} 2^{-n} \frac{1}{f(n)} < \infty$ (we refer the reader for a detailed treatment of the topic to [5,15]), and, on the other hand, of preserving the Scott spirit and being able to compute, in a natural way and simultaneously, both the meaning and the running time of computing of recursive algorithms.

The technique

Let k ϵ N and let g_i : N → N be an unbounded monotone function with respect to the partial order ≤ such that $g_i(n) < n$ for all i = 1,…..,k.

Fix n_0 ϵ N and denote by N_{n_0} the set {n ϵ N : n > n_0}. Assume that ϕ: N_{n0}]0, ∞]^k →]0, ∞] is an unbounded monotone function with respect to the partial order ≤, where ≤ is defined point-wise, i.e., (x_1……… x_{k+1})≤ (y_1,…………, y_{k+1}) ⟺ x_i ≤ y_i for all i = 1,…….,k + 1.

Next fit x c_1,……….,c_k ϵ]0, ∞ [. Consider the general recurrence equations on N given for all n ϵ N by

$$T(n) = \begin{cases} c_n & \text{if } n \le n_o \\ \Phi(n, T(g_1(n)), \ldots, T(g_k(n))) & \text{if } n > n_o \end{cases} \quad (3)$$

Clearly the recurrence equations of type (1) and (2) are obtained as a particular case of recurrence equations (3) when

$$\Phi(n, x_1, \ldots, x_k) = \sum_{i=1}^{k} a_i x_i + d(n), \quad (4)$$

for all n ϵ N_{n0} and for all x; x_1,……………, x_k ϵ]0, ∞ [and, in addition, the implicated functions g_i are chosen respectively as follows:

1. $g_i(n)$ = n-i for all n ϵ N_{n0} and for all i = 1; : : : ; k,

2. $g_i(n)$ = $T(n) = \begin{cases} c_n \\ \sum_{i=1}^{k} a_i(n) x_i + d(n) \end{cases}$ for all n ϵ N_{n0} and for all i = 1; : : : ; k,

Another family of recurrence equations that arise in a natural way in the asymptotic analysis of algorithms is the given as follows:

$$T(n) = \begin{cases} c_n & if\ n \le n_o \\ \sum_{i=1}^{k} a_i(n) x_i + d(n) & if\ n \ge n_o \end{cases} \quad (5)$$

where the constants a_1,……….,a_k have been replaced by functions a_i : N_{n0}→]0, ∞ [for all i = 1,………, k. Clearly, the preceding recurrences equations can be retrieved from (3) taking the function as follows:

$$\Phi(n, x_1, \ldots, x_k) = \sum_{i=1}^{k} a_i(n) x_i + d(n) \quad (6)$$

for all n ϵ N_{n0} and for all x; x_1,…………,x_k ϵ]0, ∞ [. Of course the one-dimensional case appears often in the literature (see [23]), i.e., when k = 1 and, hence,

$$T(n) = \begin{cases} c_n & \text{if } n \le n_o \\ a(n) T(g(n)) + d(n) & \text{if } n > n_o \end{cases} \quad (7)$$

With a: N_{n0}→] 0, ∞ [and d ϵ τ

An illustrative example of those algorithms whose running time of computing satisfies a recurrence of type (7) is the Quick sort (average behavior). Indeed, its running time is the solution to the recurrence equation given below (see, for instance, [20,22]):

$$T(n) = \begin{cases} c & if\ n = 1 \\ \frac{n+1}{n} T(n-1) + 2 & if\ n > 1 \end{cases} \quad (8)$$

With the aim of getting a technique able to provide asymptotic bounds of the running time of computing of algorithms whose analysis yields the pre-ceding recurrence equations, we first stress that every recurrence equation of type (3) has trivially always a unique solution provided the initial conditions c_1,…………,c_n and taking into account that Φ and $g_1,\dots\dots,g_k$ are functions (and thus they give a unique image for a given input). So we only need to focus our attention on how we can get a bound for such a solution without knowing its specific expression. To this end, denote by $\tau_{n_o c}$ the subset of $\tau_{n_o c}$ given by

$\tau_{n_o c}$ = {f ϵ τ : f(n) = c_n for all n ≤n_0}.

Now, de ne the functional F_{Φ}: $\tau_{n_o c} \to \tau_{n_o c}$ by

$$F_{\Phi}(f)(n) = \begin{cases} c_n & \text{if } n \le n_o \\ \Phi(n, f(g_{1(n)),\dots,f(g_k}(n))) & \text{if } n > n_o \end{cases} \qquad (9)$$

for all f ϵ $\tau_{n_o c}$. It is clear that a function belonging to $\tau_{n_o c}$ is a solution to the recurrence equation (3) if and only if it is a fixed point of the functional F_{Φ}.

Taking into account the preceding notation we show that the fundamental assumptions that are required by the Kleene fixed point theorem, chain completeness and order-continuity, are satisfied in our approach.

Proposition 2: ($\tau_{n_o c}$ $\le_{\tau_{n_o c}}$) is chain complete.

Proof: Let $(f_m)_m$ ϵ_N be an increasing sequence in ($\tau_{n_o c}$ ≤τ) Then the function f ϵ $\tau_{n_o c}$ given by

f(n) = sup{f_m(n) : m ϵ N}

for all n ϵ N is the supremum of $(f_m)_m$ ϵ_N in ($\tau_{n_o c}$ ≤τ). Clearly if f_m ϵ $\tau_{n_o c}$

for all m ϵ N, then f ϵ $\tau_{n_o c}$

Theorem 3:

Let n_0 ϵ N and let Φ be the unbounded monotone function associated to (3). If there exists K ϵ]0, ∞ [such that

$$\Phi(n, x_1 + \varepsilon, \dots, x_k + \varepsilon) < \Phi(n, x_1, \dots, x_k) + K_{\varepsilon} \qquad (10)$$

for all n ϵ N_{n0} and for all $x_1,\dots\dots\dots,x_k$; " ϵ]0, ∞[, then F_{Φ} is ≤τ -continuous.

Proof: Suppose that $(f_m)_m$ ϵ_N is an increasing sequence in ($\tau_{n_o c}$ ≤ $\tau_{n_o c}$). Then Proposition 2 guarantees that the function f ϵ $\tau_{n_o c}$ given by

$$f(n) = \sup_{m \in \mathbb{N}} f_m(n)$$

for all n ϵ N is the supremum of $(f_m)_m$ ϵ_N. Next de ne the function f ϵ $\tau_{n_o c}$ by

$$f_{\Phi}(n) = \begin{cases} c_n & \text{if } n \le n_o \\ \sup_{m\in\mathbb{N}} \Phi(n, f_m(g_1(n)),\dots, f_m(g_k(n))) & \text{if } n > n_o \end{cases}$$

Since Φ is monotone with respect to ≤ and fm ≤τ f for all m ϵ N we obtain that

$$\Phi(n, f_m(g_1(n)),\dots, f_m(g_k(n))) \le \Phi(n, f(g_1(n)),\dots, f(g_k(n)))$$

for all n ϵ N_{n0}. It follows that

$$f_{\Phi}(n) \le \Phi(n, f(g_1(n)),\dots, f(g_k(n))) \qquad (11)$$

for all n ϵ N_{n0}. Hence $f_{\Phi} \le \tau F_{\Phi}(f)$.

It remains to prove that F_{Φ} (f) $\le$ f_{Φ}. To this end, we can assume without loss of generality that $f_{\Phi}(n) < \infty$ for all n ϵ N_{n0}, since otherwise, by inequality (11), we deduce that $F_{\Phi}(f)(n) = \Phi(n, f(g_1(n)),\dots, f(g_k(n))) = \infty$. Fix n ϵ N_{n0}. Then, given m_{ε} ϵ]0, ∞ [, there exists m_{ε} ϵ N such that

$$f(g_i(n)) < f_m(g_i(n)) + \varepsilon$$

for all i = 1,………,k.

Hence the monotony of Φ and (10) give

$$\Phi(n, f(g_1(n)),\dots, f(g_k(n))) \le$$

$$\Phi(n, f_{m\in}(g_1(n)) + \varepsilon,\dots, f_{m\in}(g_k(n)) + \varepsilon)$$

$$\Phi(n, f_{m\in}(g_1(n)),\dots, f_{m\in}(g_k(n))) + \varepsilon K) \le$$

$$f_{\Phi}(n) + \varepsilon K$$

Whence we deduce that F_{Φ}(f)(n) ≤f_{Φ}(n) for all n ϵ N_{n0}. So F_{Φ}(f) τ f_{Φ}.

Therefore we conclude that F_{Φ}(f) = f_{Φ} and, thus, that F_{Φ} is ≤τ continuous.

Notice that the functions $_{\Phi}$ given by (4) and (6) are unbounded monotone and, in addition, they satisfy the regularity condition (10).

The next example proves that only the monotony of does not provide the ≤ τ -continuity of the function F_{Φ}.

Example 4: Fix n_0 ϵ N and c_1,……c ϵ]0, ∞ [. Let g_1, g_2: N→N be two unbounded monotone functions with respect to ≤ such that g_i(n) < n for all i = 1; 2. Consider the function ф: N_{n0} X]0, ∞] →]0, ∞] given by

$$\Phi(n, x) = \begin{cases} \infty & if x \ge 1 \\ x & if x < 1 \end{cases}$$

It is obvious that ф is monotone with respect to ≤. Moreover, ф does not satisfy the regularity condition (10), since there is not K ϵ]0, ∞ [such that

$$\Phi(n, x + \varepsilon) \le \Phi(n, x) + K\varepsilon$$

for all n ϵ N_{n0} and x, ϵ]0, τ [. Indeed, put τ = 1 and take x =1/2. Then

$$\frac{1}{2} + K = \Phi(n, \frac{1}{2}) + K \le \Phi(n, \frac{3}{2}) = \infty.$$

A straightforward computation shows that F_{Φ} is not ≤τ -continuous. Clearly the sequence $(f_m)_m$ ϵ N in $\tau_{n_o,c}$, given by

$$f_m(n) = \begin{cases} c_n & \text{if } n \le n_o \\ 1 - \dfrac{1}{m+1} & \text{if } n > n_o \end{cases}$$

for all m ϵ N, is increasing in ($\tau_{n_o,c}$ $\le_{\tau_{n_o,c}}$) and its supremum is the function f ϵ $\tau_{n_o,c}$ given by

$$f_m(n) = \begin{cases} c_n & \text{if } n \le n_o \\ 1 & \text{if } n > n_o \end{cases}$$

Besides, the sequence (F_{Φ} $(f_m))_m$ ϵ_N is increasing in (T $\tau_{,c}$ ≤ τ) and its supremum is again the function f. Nevertheless, F_{Φ} (f) ≠ f

The next example shows that only the regularity condition (10) for is not enough in order to assure the $\le_{n_o}$ -continuity of the function F.

Example 5: Fix n_0 ϵ N and c_1,……c_{n_0} ϵ]0, ∞ [. Let g_1, g_2: N→N be two unbounded monotone functions with respect to ≤ such that g_i(n) < n for all i = 1; 2. Consider the function ф: N_{n0} X]0, ∞ $]^2$ →]0, ∞] given by

$$\Phi(n, x_1, x_2) = \begin{cases} \infty & If\ x_1 = \infty \text{ or } x_2 = \infty \\ x_1.x_2 & \text{otherwise} \end{cases}$$

It is evident that satisfies the regularity condition (10), since

$$\frac{1}{x_1+x_2+2\varepsilon}<\frac{1}{x_1+x_2}+\varepsilon$$

for all n ϵ N_n and for all x_1, x_2, ϵ]0, ∞]. Moreover, it is clear ϕ is not monotone with respect to ≤. Indeed,

$$\Phi(n,\frac{x_1}{2},\frac{x_2}{2})=\frac{2}{x_1+x_2}\geq\frac{1}{x_1+x_2}=\Phi(n,x_1,x_2)$$

for all x_1, x_2, ϵ]0, ∞]. It follows that F_ϕ is not monotone with respect to ≤ and, thus, it is not $\leq_\tau$ continuous.

The next example yields that the $\leq_{n_o}$ continuity of F_ϕ does not guarantee that the function fulfills the condition (10).

Example 6: Fix n_0 ϵ N and $c_1,\ldots\ldots,c_{n_o}$ ϵ]0, ∞ [. Let g_1, g_2: N→N be two unbounded monotone functions with respect to ≤ such that $g_i(n)$ < n for all i = 1; 2. Consider the function ϕ: N_{n0}X]0, ∞ $]^2$ →]0, ∞] given by

$$\Phi(n,x_1,x_2)=\begin{cases}\infty & \mathit{If}\ x_1=\infty \text{ or } x_2=\infty\\ x_1.x_2 & \text{otherwise}\end{cases}$$

It is evident that the function F_ϕ is $\leq_\infty$ -continuous. However, the function ϕ does not satisfy the regularity condition (10). Indeed, assume with the purpose of contradiction that the aforesaid condition is hold. Then, there exists K ϵ]0, ∞]such that

$$\Phi(n,x_1+\varepsilon,x_2+\varepsilon)<\Phi(n,x_1,x_2)+\varepsilon K$$

for all n ϵ N_{n0} and for all x_1, x_2, ϵ]0, ...]It follows that

$$x_1+x_2+\varepsilon<K$$

for all x_1, x_2, ϵ]0, ∞]. Set $x_1 = x_2 = K/2$. Then, from the preceding inequality, we deduce that

$$K+\varepsilon<K$$

This is impossible. So, we conclude that does not satisfy the regularity condition (10).

Once we have guaranteed that in our framework the main components required to apply the Kleene fixed point theorem are hold, we introduce the new methodology to provide asymptotic complexity bounds for those algorithms whose running time of computing fulfills a recurrence equation of type (3).

Theorem 7: Let n_0 ϵ N and let $c_1,\ldots\ldots,c_{n0}$ 2]0, ∞[. Assume that Φ: N_{n0} X]0, ∞ $]^k$ →]0; ∞] is an unbounded monotone function with respect to ≤ which satisfies the regularity condition (10) and that f is the solution to the recurrence equation of type (3). If there exists g ϵ τ_{n_oc} such that g ≤ τ_{n_oc} F_ϕ(g), then f ϵ Ω(g). Moreover, if there exists h ϵ $\tau_{n_o,c}$ such that h ϵ↑ $\tau_{n_o,c}$ g and F_ϕ (h) $\sup_{m\in\mathbb{N}}$ h, then f ϵ O(h).

Proof: First of all, note that f is the unique solution to (3) and, thus, the unique fixed point of F_ϕ in $\tau_{n_o,c}$. Suppose that there exists g ϵ $\tau_{n_o,c}$ such that g ≤ τ_{n_oc} F_ϕ(g). Since ($\tau_{n_o,c}$ ≤ τ) is chain complete and ϕ is $\leq_{\tau_{n_o,c}}$ continuous we have, by assertion 1) in the statement of Kleene's fixed point theorem (Theorem 1), that there exists a fixed point f* ϵ $\tau_{n_o,c}$ of F_ϕ such that f* ϵ↑≤ $\tau_{n_o,c}$ g and, hence, that f* ϵ Ω (g). The fact that F_ϕ admits only a unique fixed point gives that f = f* and that f ϵΩ(g).

Suppose that there exists h ϵ τ_{n_oc} such that h* ϵ↑≤ τ_{n_oc} g and F (h) ≤ τ_{n_oc} h. Assertion 2) in the statement of Theorem 1 provides that f≤ $\tau_{n_o,c}$ h. Therefore, f ϵ O(h).

In the light of the preceding result we draw in the inference that in order to get the asymptotic bounds of an algorithm whose running time of computing satisfies a recurrence equation of type (3), it is enough to search the bounds among those functions in $\tau_{n_o,c}$ that are "post- fixed "point of F_ϕ (g $\tau_{n_o,c}$ F (g)) and those that are a "pre- fixed" point of F_ϕ(F_ϕ(h)≤ $\tau_{n_o,c}$ h). Moreover, the condition in the statement of Theorem 7 that states a relationship between the post fixed point g and the pre-fixed point h, that is h ϵ ↑ g, points out that really the upper bound class O(h) must be included in the lower bound class Ω (g) which is reasonable from a complexity theory viewpoint.

An illustrative example

Let us consider Quicksort. According to its running time of computing f_Q, for the average behavior, is the solution to the recurrence equation (12), i.e., to the following recurrence [22]:

where c ϵ]0, ∞[.

$$T(n)=\begin{cases}c & \mathit{if}\ n=1\\ \frac{n+1}{n}T(n-1)+2 & \mathit{if}\ n>1\end{cases} \tag{12}$$

As indicated in Subsection 3.1, the preceding recurrence equation is retrieved from (3) when we consider:

$$k=1,\ c_1=c \text{ and } n_0=2.$$

$$g_1(n)=n-1,\ a_1(n)=\frac{n+1}{n} \text{ and } d(n)=2 \text{ for all } n\in\ \mathbb{N}_2$$

$$\Phi(n,x)=\frac{n+1}{n}x+2 \text{ for all } n\in\ \mathbb{N}_2 \text{ and for all } x\in]0,\ \infty[.$$

$$\Phi(n,\infty)=\infty \text{ for all } n\in\ \mathbb{N}_2.$$

$$f_\Phi(f)(n)=\Phi(n,f(n-1))=\frac{n+1}{n}f(n-1)+2 \text{ for all } f\in\tau_{1,c}=\{f\in\tau: f(1)=c\} \text{ and for all } n\in\ \mathbb{N}_2.$$

Notice that ϕ (n, x + ϵ) < ϕ(n, x)+ K ϵ for all K ϵ]2, ∞[, n ϵ N_2 and x ϵ]0, ∞[.

In the light of the preceding facts we have that all assumptions required in the statement of Theorem 7 are hold. With the aim of making clear the use of the technique yielded by the aforesaid theorem we verify that the asymptotic bounds known in the literature are retrieved from our approach. To this end, let f_Q be the solution to the recurrence equation (12).

On the one hand, it is not hard to check that, given g ϵ $\tau_{1,c}$, g ≤ $\tau_{1,c}$ F_Φ(g) ⟺g fulfills the following:

$$g(n)\leq\begin{cases} & \mathit{if}\ n=1\\ 2+-C & \mathit{if}\ n=2\\ \frac{}{2}(n+1)+2(n+1)\sum\ \frac{}{2} & \mathit{if}\ n\geq 3\end{cases}$$

On the other hand, it is not hard to check that, given h ϵ $\tau_{1,c}$, F_Φ(h)≤ $\tau_{1,c}$ h⟺h fulfills the following:

$$h(n)\geq\begin{cases}c & \mathit{if}\ n=1\\ 2+\frac{3}{2}C & \mathit{if}\ n=2\\ \frac{c}{2}(n+1)+2(n+1)\sum_{i=1}^{n-2}\frac{1}{i+2} & \mathit{if}\ n\geq 3\end{cases}$$

Next take f ϵ $\tau_{1,c}$ defined by

$$f(n)=\begin{cases}c & \mathit{if}\ n=1\\ 2+\frac{3}{2}C & \mathit{if}\ n=2\\ \frac{c}{2}(n+1)+2(n+1)\sum_{i=1}^{n-2}\frac{1}{i+2} & \mathit{if}\ n\geq 3\end{cases}$$

By Theorem 7 we deduce that f_QϵΘ(f) which immediately gives the well-known asymptotic bound f_QϵΘ(f_{log}) (see, for instance, [24]), where $f_{\log} \in \mathcal{T}_{1,c}$ with

$$f_{\log}(n) = \begin{cases} c & \text{if } n = 1 \\ 2 + \frac{3}{2}C & \text{if } n = 2 \\ n\log(n) & \text{if } n \geq 3 \end{cases} \quad (13)$$

We end this subsection pointing out that Scott's technique presents a unified mathematical approach useful at the same time for Asymptotic Complexity Analysis and Denotational Semantics. Thus, for instance, such a method allows to model simultaneously both, the meaning and the running time of computing of recursive algorithms using recursive denotational specifications. An easy, but representative, example to which the technique could be applied is given by an algorithm computing the factorial function by means of a recursive specification whose meaning satisfies the following de-notational specification (14) and its running time of computing fulfills the following recurrence equation (15):

$$f\ act(n) = \begin{cases} 1 & \text{if } n = 1 \\ n\,f\ act(n-1) & \text{if } n > 1 \end{cases} \quad (14)$$

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ T(n-1) + d & \text{if } n > 1 \end{cases} \quad (15)$$

where c, d > 0.

Conclusion

In 1972, D.S. Scott developed a qualitative mathematical technique for modeling the meaning of recursive algorithms in Denotational Semantics. We have shown that the same original Scott's technique remains valid for Asymptotic Complexity Analysis of algorithms. So we have seen that such a qualitative approach presents a unified mathematical method that is useful for Asymptotic Complexity Analysis and Denotational Semantics. This fact presents an advantage over the quantitative technique, based on Banach's fixed point theorem, introduced in 1995 by M.P. Schellekens because such a technique has been designed mainly for Asymptotic Complexity Analysis. Moreover, the use of the qualitative approach agrees with the qualitative character of the complexity analysis of algorithms in Computer Science, where to provide the asymptotic behavior of the running time is more important than to get the exact expression of the running time itself. Further-more, using the qualitative framework we avoid to require the convergence condition" assumed by the quantitative Schellekens approach whose unique purpose is to guarantee the soundness of a distance, which provides the quantitative information, but it has not a priori a computational motivation.

Acknowledgement

This research was funded by the Spanish Ministry of Economy and Competitiveness under Grants TIN2014-56381-REDT (LODISCO) and TIN2016-81731-REDT (LODISCO II) and AEI/FEDER, UE funds.

References

1. Davey BA, Priestley HA. Introduction to Lattices and Order. Cam-bridge University Press, Cambridge; 1990.
2. Scott DS. Outline of a mathematical theory of computation, in: Proc. 4th Annual Princeton Conference on Information Sciences and Systems. 1970; 169-176.
3. CA Gunter, DS Scott. Semantic domains, in: Handbook of Theoretical Computer Science, ed. by J van Leewen. 1990; Vol. B; MIT Press, Cambridge: 663-674.
4. Stoy JE. Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory, MIT Press, Cambridge; 1977.
5. Schellekens MP. The Smyth completion: a common foundation for the denotational semantics and complexity analysis. Electron Notes Theor Comput Sci. 1995; 1: 535-556.
6. Hitzler P, Seda AK. Mathematical Aspects of Logic Programming Semantics. CRC Press, Boca Raton; 2011.
7. Alghamdi MA, Shahzad N, Valero O. Fixed point theorems in generalized metric spaces with applications to Computer Science. Fixed Point Theory A. 2013; 2013: 118.
8. Alghamdi MA, Shahzad N, Valero O. On fixed point theory in topological posets, extended quasi-metrics and an application to asymptotic complexity analysis of algorithms. Fixed Point Theory A. 2015; 2015: 179.
9. Uguet MAC, Schellekens MP, Valero O. The Baire partial quasi-metric space: a mathematical tool for the asymptotic complexity analysis in Computer Science. Theor Comput Syst. 2012; 50: 387-399.
10. Garcia RLM, Romaguera S, Schellekens MP. Applications of the complexity space to the general probabilistic Divide and Conquer algorithms. J Math Anal Appl. 2008; 348: 346-355.
11. Z Mohammadi, O Valero. A new contribution to fixed point theory in partial quasi-metric spaces and its applications to asymptotic complexity analysis of algorithms. Topol Appl. 2016; 203: 42-56.
12. Lopez JR, Schellekens MP, Valero O. An extension of the dual complexity space and an application to Computer Science. Topol Appl. 2009; 156: 3052-3061.
13. Romaguera S, Schellekens MP. Quasi-metric properties of complexity spaces. Topol Appl. 1999; 98: 311-322.
14. Romaguera S, Schellekens MP, Valero O. The complexity space of partial functions: A connection between Complexity Analysis and De-notational Semantics. Int J Comput Math. 2011; 88: 1819-1829.
15. Romaguera S, Tirado P, Valero O. New results on mathematical foundations of asymptotic complexity analysis of algorithms via complexity spaces. Int J Comput Math. 2012; 89: 1728-1741.
16. Romaguera S, Valero O. A common mathematical framework for asymptotic complexity analysis and denotational semantics for recur-sive programs based on complexity spaces. Advan Theories Math Models. 2012; 1: 99-120.
17. Baranga A. The contraction principle as a particular case of Kleene's fixed point theorem. Discrete Math. 1991; 98: 75-79.
18. G Brassard, P Bratley. Algorithms: Theory and Practice, Prentice Hall, New Jersey; 1988.
19. M Kao. Multiple-size divide-and-conquer recurrences. ACM SIGACT News. 1997; 28: 67-69.
20. Cull P, Flahive R, Robson R. Difference Equations: from Rabbits to Chaos. Springer, New York; 1985.
21. Blum M, Floyd R, Pratt V, Rivest R, Tarjan R. Time bounds for selection. J Comput Syst Sci. 1973; 7: 448-461.
22. Cormen TH, Leiserson CE, Rivest RL. Introduction to Algorithms, MIT Press. New York; 1990.
23. Wang X, Fu Q. A frame for general divide-and-conquer recurrences. Inform Process Lett. 1996; 59: 45-51.
24. R Miller, L Boxed. Algorithms Sequential and Parallel: A Unified Approach (Charles River Media) Hingham; 2005.