Fixed Point Theory and Applications
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

New results on the Baire partial quasi-metric space, fixed point theory and asymptotic complexity analysis for recursive programs

Maryam A Alghamdi[1], Naseer Shahzad[2*] and Oscar Valero[3]

*Correspondence:
nshahzad@kau.edu.sa
[2]Department of Mathematics, King Abdulaziz University, P.O. Box 80203, Jeddah, 21859, Saudi Arabia
Full list of author information is available at the end of the article

Abstract

In Cerdà-Uguet *et al.* (Theory Comput. Syst. 50:387-399, 2012), a new mathematical fixed point technique, that uses the so-called Baire partial quasi-metric space, was introduced with the aim of providing the asymptotic complexity of a class of recursive algorithms. The aforementioned technique presents the advantage that requires less calculations than the quasi-metric original one given by Schellekens (Electron. Notes Theor. Comput. Sci. 1:211-232, 1995). In this paper we continue the study, started in Cerdà-Uguet *et al.* (Theory Comput. Syst. 50:387-399, 2012), on the use of partial quasi-metric spaces for asymptotic complexity analysis of algorithms. Concretely, our main purpose is to prove that the Baire partial quasi-metric space is an appropriate mathematical framework for discussing via fixed point arguments the asymptotic complexity of a general class of recursive algorithms to which all the algorithms analyzed in Cerdà-Uguet *et al.* (Theory Comput. Syst. 50:387-399, 2012) belong. The obtained results are illustrated by means of applying them to yield the complexity of two celebrated recursive algorithms which don not belong to the class discussed in Cerdà-Uguet *et al.* (Theory Comput. Syst. 50:387-399, 2012).

MSC: 47H10; 54E50; 68Q15; 68Q25; 68W40

Keywords: asymptotic complexity analysis; recurrence equation; quasi-metric; partial metric; Baire partial quasi-metric; running time of computing; fixed point

1 Introduction

In 1994, Matthews introduced the notion of partial metric space in order to obtain a new mathematical framework to model computational recursion processes in program verification by means of 'metric' fixed-point techniques [1]. Moreover, he gave an application of this new metric structure to parallel computing by means of a partial metric version of the celebrated Banach fixed-point theorem in [2]. Later on, in 1995, Schellekens introduced the (quasi-metric) complexity space as a new mathematical approach of the foundation for the asymptotic complexity analysis of algorithms [3]. The applicability of this theory to the asymptotic complexity analysis of Divide and Conquer algorithms was also illustrated by Schellekens, in the same reference, using fixed-point arguments based on the use of a quasi-metric version of the aforementioned Banach fixed-point theorem.

Motivated by the utility of quasi-metric spaces for the asymptotic complexity analysis of algorithms via fixed-point techniques and the usefulness of partial metric spaces to analyze program correctness, the possibility of analyzing the asymptotic complexity

of algorithms via Matthews' fixed-point theorem was discussed recently in [4]. However, in the aforesaid reference it was shown that, at first, such an analysis cannot be carried out through the aforementioned result. In consequence, in the preceding reference a new mathematical tool, which was called the Baire partial quasi-metric space, was introduced in order to provide a novel mathematical framework for the asymptotic complexity analysis of algorithms which unify, under the same framework, the quasi-metric fixed-point arguments originating with the Schellekens approach and the seminal ideas of Matthews on partial metric spaces. The mathematical framework based on the Baire partial quasi-metric space presents the advantage of providing the complexity of the algorithms under consideration through easier and fewer calculations than those given in the Schellekens methods. In [4], the applicability of the Baire partial-quasi-metric fixed-point techniques were illustrated discussing the complexity of a few particular and celebrated recursive algorithms. Nevertheless, in the class of recursive algorithms there are some whose complexity cannot be analyzed be means of the fixed-point techniques developed in [4]. Examples of this kind of algorithms are the well-know algorithms that solves the Towers of Hanoi puzzle and the algorithm that computes the value of the Fibonacci sequence, which we will call Hanoi and Fibonacci, respectively, in the following.

In this paper we continue and improve the study started in [4] of the use of partial quasi-metrics for asymptotic complexity analysis of algorithms. Concretely, our main purpose is to prove that the Baire partial quasi-metric space is a suitable mathematical framework for discussing by means of fixed-point arguments the asymptotic complexity of a general class of recursive algorithms to which the Hanoi, Fibonacci and all the algorithms analyzed in [4] belong.

The remainder of the paper is organized as follows: Section 2 is devoted, on the one hand, to recall the basics on asymptotic complexity analysis of algorithms and, on the other hand, to introduce the reader to the associated current metric fixed-point tools. Concretely, we recall briefly, with the aim of motivating our subsequent work, the Schellekens fixed-point method and how it can be applied to the complexity analysis of some Divide and Conquer algorithms. Moreover, the fundamentals on partial metric spaces and their utility in complexity analysis of algorithms are also discussed. Furthermore, in the same section, we recall, according to [4], how both frameworks, the quasi-metric and the partial metric one, are unified in order to construct a new mathematical approach, the so-called Baire partial quasi-metric space, which preserves the original Schellekens ideas and allows to apply the Matthews seminal ones to complexity analysis. In Section 3, our new results are presented. In particular, we introduce a new fixed-point technique based on the Baire partial quasi-metric space that extends the provided one in [4] and, in addition, allows to provide easily the complexity of some algorithms which cannot be analyzed by means of the aforementioned technique given in [4]. Finally, we validate the developed theory applying it to an analysis and retrieving the asymptotic complexity of the celebrated Hanoi and Fibonacci.

2 The asymptotic complexity analysis of algorithms and metric fixed-point tools

2.1 The fundamentals

From now on, $\mathbb{R}^+$ and $\mathbb{N}$ will denote the set of nonnegative real numbers and the set of positive integer numbers, respectively.

In complexity analysis of algorithms, the running time of computing, that is the time taking by an algorithm in order to solve the problem for which it has been designed, plays a crucial role (see, for instance, [5]). Usually there are several algorithms that are able to solve a fixed problem. Thus, in a natural way, the complexity analysis focus its interest on determining which of them has less complexity, that is which takes less running time of computing. In order to compare the complexity of all algorithms solving the same problem, the running time of computing of each algorithm is denoted by a function $T : \mathbb{N} \to (0, \infty]$ in such a way that $T(n)$ represents the time taken by the algorithm to solve the problem under consideration when the input of the algorithm is of size n. Hence, one can compare the running time of computing all the aforesaid algorithms by means of the comparison of their associated functions.

Nevertheless, given an algorithm, to asses which is the function that describes its running time of computing for every input size is, in many cases, a hard challenge. For this reason, in most situations it is enough to work, in order to compare the algorithm complexity, with an approximate running time of computing of each algorithm to be compared. To provide the running time of computing of algorithms in an approximate way is the main objective of the so-called asymptotic complexity analysis of algorithms.

In what follows we recall a few pertinent notions from asymptotic complexity analysis which will be a crucial role in our subsequent work. To this end, we will denote by $\mathcal{RT}$ the set of all functions from $\mathbb{N}$ into $(0, \infty]$.

Let $f \in \mathcal{RT}$ denote the running time of computing of a concrete algorithm. Moreover, given a certain function $g \in \mathcal{RT}$, consider that there exist $n_0 \in \mathbb{N}$ and $c > 0$ satisfying $f(n) \leq cg(n)$ for all $n \in \mathbb{N}$ with $n \geq n_0$ (of course $\leq$ stands for the usual order on $(0, \infty]$). Hence, it is clear that the function g provides an asymptotic upper bound of the running time f of the algorithm under consideration. Therefore, if we do not have exact information as regards the expression of the function f, then the function g yields us an 'approximate' information of the running time of computing f in such a way that the considered algorithm takes a time to solve the problem bounded asymptotically above by g. Following the standard notation, we will write $f \in \mathcal{O}(g)$ provided that $f, g \in \mathcal{RT}$ and g is an asymptotic upper bound of f.

Of course, an asymptotic upper bound itself does not provide much information on the complexity f of the algorithm under study. In order to give a tighter bound on the complexity f, in asymptotic complexity analysis of algorithms are used in addition asymptotic lower bounds of the running time of computing. Concretely, a function $h \in \mathcal{RT}$ provides an asymptotic lower bound of the unknown running time of computing f under consideration provided that there exist $n_0 \in \mathbb{N}$ and $c > 0$ such that $ch(n) \leq f(n)$ for all $n \in \mathbb{N}$ with $n \geq n_0$. When h is an asymptotic lower bound of f we will denote it, as usual, by $f \in \Omega(h)$.

Therefore when we have found two functions $h, g \in \mathcal{RT}$ such that $f \in \Omega(h) \cap \mathcal{O}(g)$, we obtain, through h and g, a complete asymptotic information about the time f taken by the algorithm under discussion. Moreover, in case of f obeying the condition $f \in \Omega(h) \cap \mathcal{O}(g)$, for any $h, g \in \mathcal{RT}$, we will say that $\Omega(h) \cap \mathcal{O}(g)$ is the complexity class of f. Furthermore, we will denoted the complexity class of f by $\Theta(l)$ provided the existence of a function $l \in \mathcal{RT}$ such that $f \in \Omega(l) \cap \mathcal{O}(l)$.

Accordingly, the main aim of the asymptotic complexity analysis is to describe the running time of algorithms by means of determining its asymptotic complexity class.

2.2 The Schellekens approach and the quasi-metric complexity space

According to [6], and following modern terminology, a quasi-metric space is a pair (X,d) such that X is a non-empty set and d is a quasi-metric on X, where by a quasi-metric we mean a function $d : X \times X \to \mathbb{R}^+$ such that for all $x,y,z \in X$:

(i) $d(x,y) = d(y,x) = 0 \Leftrightarrow x = y$;

(ii) $d(x,y) \leq d(x,z) + d(z,y)$.

Clearly, a metric space is a quasi-metric space (X,d) in such a way that d satisfies the next additional condition for all $x,y \in X$:

(iii) $d(x,y) = d(y,x)$.

It is well known that each quasi-metric d on X generates a T_0-topology $\mathcal{T}(d)$ on X which has as a base the family of open d-balls $\{B_d(x,\varepsilon) : x \in X, \varepsilon > 0\}$, where $B_d(x,\varepsilon) = \{y \in X : d(x,y) < \varepsilon\}$ for all $x \in X$ and $\varepsilon > 0$. Moreover, a quasi-metric space (X,d) is called bicomplete if the metric space (X,d^s) is complete, where the metric d^s is defined for all $x,y \in X$ by $d^s(x,y) = \max(d(x,y),d(y,x))$.

In 1995, as we have mentioned in Section 1, Schellekens introduced the so-called complexity space with the aim of developing a mathematical foundation for the asymptotic complexity analysis of algorithms [3]. Let us recall that the complexity space consists of the pair $(\mathcal{C},d_{\mathcal{C}})$, where

$$\mathcal{C} = \left\{ f \in \mathcal{RT} : \sum_{n=1}^{\infty} 2^{-n} \frac{1}{f(n)} < \infty \right\}$$

and $d_{\mathcal{C}}$ is the bicomplete quasi-metric on $\mathcal{C}$ defined by

$$d_{\mathcal{C}}(f,g) = \sum_{n=1}^{\infty} 2^{-n} \max\left(\frac{1}{g(n)} - \frac{1}{f(n)}, 0 \right). \tag{1}$$

Of course in equation (1) the convention that $\frac{1}{\infty} = 0$ is adopted.

Although in asymptotic complexity analysis of algorithms the running time of computing is represented by means of functions in $\mathcal{RT}$ and, thus, the condition '$\sum_{n=1}^{\infty} 2^{-n} \frac{1}{f(n)} < \infty$' in not required, according to [3], every reasonable algorithm, from a computability point of view, must obey the aforementioned condition. Hence each algorithm can be associated with a function belonging to $\mathcal{C}$ which represents, as a function of the input size, its running time of computing.

The utility of the complexity space in algorithm complexity is given, in part, by the computational interpretation of the numerical value $d_{\mathcal{C}}(f,g)$. Concretely, given two functions $f,g \in \mathcal{C}$, the complexity distance from f to g, that is $d_{\mathcal{C}}(f,g)$, provides information about the relative progress made in lowering the complexity by replacing any program A with complexity function f by any program B with complexity function g in such a way that if $f \neq g$, the fact that $d_{\mathcal{C}}(f,g) = 0$ can be interpreted as the program A is at least as efficient as the program B. In fact, the condition $d_{\mathcal{C}}(f,g) = 0$ implies that $f \in \mathcal{O}(g)$.

It must be stressed that the asymmetry of $d_{\mathcal{C}}$ is key in order to provide information about the increase of complexity whenever an algorithm is replaced by another one. Of course, a metric would give at most information on the increase but it would not be able to provide which algorithm of both, A or B, is more efficient.

In [3], Schellekens gave an application of the complexity space to asymptotic complexity analysis. In particular, he gave a new method, based on the use of the celebrated Banach

fixed-point theorem, to analyze the running time of computing of Divide and Conquer algorithms.

In what follows we recall briefly the aforesaid method, since this will allow the reader to gain a better understanding of the motivation for our subsequent work (exposed in Section 3). To this end, let us recall that the Banach fixed-point theorem can be stated in the quasi-metric framework as follows.

Theorem 1 *Let f be a mapping from a bicomplete quasi-metric space (X,d) into itself such that there exists $s \in [0,1)$ satisfying*

$$d\big(f(x),f(y)\big) \leq sd(x,y),$$

for all $x,y \in X$. Then f has a unique fixed point.

In many cases, the recursive structure of a Divide and Conquer algorithm yields the result that its running time of computing satisfies the recurrence equation

$$T(n) = \begin{cases} c & \text{if } n = 1, \\ aT(\frac{n}{b}) + h(n) & \text{if } n \in \mathbb{N}_b, \end{cases} \tag{2}$$

where $a,b \in \mathbb{N}$ with $a,b > 1$, $\mathbb{N}_b = \{b^k : k \in \mathbb{N}\}$, $c > 0$ and $h \in \mathcal{C}$ with $h(n) < \infty$ for all $n \in \mathbb{N}$.

Observe that for Divide and Conquer algorithms, it is sufficient to obtain the complexity on inputs of size n with n ranges over the set $\mathbb{N}_b$ (for a fuller treatment we refer the reader to [5]).

In order to provide the running time of computing of a Divide and Conquer algorithm satisfying the recurrence equation (2), we have to prove that the recurrence equation has a unique solution, which represents the running time, and, in addition, we have to compute the asymptotic complexity class of such a solution. As we show in the following, the announced Schellekens method is able to prove that equation (2) has a unique solution and to yield an asymptotic upper bound (asymptotic lower bounds of the running time of Divide and Conquer algorithms were obtained by Schellekens following standard arguments which are not based on the use of fixed-point techniques):

Denote by $\mathcal{C}_{b,c}$ the subset of $\mathcal{C}$ given by

$$\mathcal{C}_{b,c} = \big\{f \in \mathcal{C} : f(1) = c \text{ and } f(n) = \infty \text{ for all } n \in \mathbb{N}\backslash\mathbb{N}_b \text{ with } n > 1\big\}.$$

Then the quasi-metric space $(\mathcal{C}_{b,c}, d_{\mathcal{C}}|_{\mathcal{C}_{b,c}})$ is bicomplete, since the quasi-metric space $(\mathcal{C}, d_{\mathcal{C}})$ is bicomplete (see [7] for the bicompleteness of $\mathcal{C}$) and the set $\mathcal{C}_{b,c}$ is closed in $(\mathcal{C}, d_{\mathcal{C}}^s)$.

Next, given the recurrence equation (2), we introduce the functional $\Phi_T : \mathcal{C}_{b,c} \to \mathcal{C}_{b,c}$ as follows:

$$\Phi_T(f)(n) = \begin{cases} c & \text{if } n = 1, \\ \infty & \text{if } n \in \mathbb{N}\backslash\mathbb{N}_b \text{ and } n > 1, \\ af(\frac{n}{b}) + h(n) & \text{otherwise.} \end{cases} \tag{3}$$

It is clear that a function in $\mathcal{C}_{b,c}$ is a solution to the recurrence equation (2) if and only if it is a fixed point of the functional Φ_T. Then, it is not hard to check that

$$d_{\mathcal{C}}|_{\mathcal{C}_{b,c}}\big(\Phi_T(f),\Phi_T(g)\big) \leq \frac{1}{a} d_{\mathcal{C}}|_{\mathcal{C}_{b,c}}(f,g) \tag{4}$$

for all $f,g \in \mathcal{C}_{b,c}$. Consequently, Theorem 1 guarantees the existence of a unique fixed point f_T of Φ_T and, thus, the existence and uniqueness of the solution to the recurrence equation (2). Of course such a solution provides a unique possible representative of the running time.

In order to approximate, according to the Schellekens approach, the running time of computing of the algorithms under consideration it remains to give an asymptotic upper bound of f_T. But Schellekens proved that $f_T \in \mathcal{O}(g)$ provided that $g \in \mathcal{C}_{b,c}$ and that $\Phi_T(g) \leq g$.

The preceding facts mentioned allow us to state the next useful result [3].

Theorem 2 *A Divide and Conquer recurrence equation of the form* (2) *has a unique solution f_T in $\mathcal{C}_{b,c}$. Moreover if there exists $g \in \mathcal{C}_{b,c}$ such that the functional Φ_T associated with equation* (2) *obeys $\Phi_T(g) \leq g$, then $f_T \in \mathcal{O}(g)$.*

With the aim of illustrating the real utility of Theorem 2, Schellekens applied it to provide a description of an asymptotic upper bound of the running time of computing of Mergesort (for a detailed discussion of Mergesort see, for instance, [5]). Specifically, he gave a new proof of the well-known fact that the (average) running time of computing of Mergesort f_T^M is in $\mathcal{O}(n \log_2 n)$.

2.3 The partial metric space approach

In 1994, as we have announced in Section 1, Matthews introduced partial metric spaces with a twofold objective [1]. On the one hand, he claimed to present a new mathematical framework to model computational recursion processes, in the spirit of Scott (see [8] for details of Scott theory), in program verification by means of 'metric' fixed-point techniques. On the other hand, in order to achieve the later purpose, he wanted to obtain an extension of Banach's fixed-point theorem to the partial metric context.

Let us recall a few basic notions about partial metric spaces in order to introduce the Matthews fixed-point theorem.

Following [1], a partial metric space is a pair (X,p) such that X is a non-empty set X and p is a function $p : X \times X \to \mathbb{R}^+$ satisfying for all $x,y,z \in X$:

(i) $p(x,x) = p(x,y) = p(y,y) \Leftrightarrow x = y$;
(ii) $p(x,x) \leq p(x,y)$;
(iii) $p(x,y) = p(y,x)$;
(iv) $p(x,y) \leq p(x,z) + p(z,y) - p(z,z)$.

Of course, a metric on a non-empty set X is a partial metric p on X satisfying, in addition, the following condition for all $x \in X$:

(v) $p(x,x) = 0$.

It is well known that each partial metric p on X generates a T_0 topology $\mathcal{T}(p)$ on X which has as a base the family of open p-balls $\{B_p(x,\varepsilon) : x \in X, \varepsilon > 0\}$, where $B_p(x,\varepsilon) = \{y \in X : p(x,y) < p(x,x) + \varepsilon\}$ for all $x \in X$ and $\varepsilon > 0$. Moreover, as a consequence, a sequence

$(x_n)_{n\in\mathbb{N}}$ in a partial metric space (X,p) converges to a point $x \in X$ with respect to $\mathcal{T}(p) \Leftrightarrow p(x,x) = \lim_{n\to\infty} p(x,x_n)$.

With the aim of introducing a partial metric version of the Banach fixed-point theorem, Matthews defined the notion of completeness in partial metric spaces. In particular, and according to [1], a sequence $(x_n)_{n\in\mathbb{N}}$ in a partial metric space (X,p) is called a Cauchy sequence if $\lim_{n,m\to\infty} p(x_n,x_m)$ exists. A partial metric space (X,p) is said to be complete if every Cauchy sequence $(x_n)_{n\in\mathbb{N}}$ in X converges, with respect to $\mathcal{T}(p)$, to a point $x \in X$ such that $p(x,x) = \lim_{n,m\to\infty} p(x_n,x_m)$.

The announced new fixed-point theorem can be stated for partial metric spaces as follows.

Theorem 3 *Let f be a mapping from a complete partial metric space (X,p) into itself such that there is $s \in [0,1)$ satisfying*

$$p\big(f(x),f(y)\big) \le sp(x,y),$$

for all $x,y \in X$. Then f has a unique fixed point. Moreover if $x \in X$ is the fixed point of f, then $p(x,x) = 0$.

It is interesting to point out that Matthews used the preceding result to provide a suitable test for lazy data flow deadlock in Kahn's model of parallel computation (for a fuller treatment of the application we refer the reader to [2]).

In [4], motivated by the usefulness of partial metric spaces in some fields in Computer Science, it was wondered whether partial metric spaces are also useful, like quasi-metric spaces, in asymptotic complexity analysis in the spirit of Schellekens. Hence, in the same reference it was noted that although the set $\mathcal{C}$ can be endowed with the partial metric $p_{\mathcal{C}}$ defined in [9] for all $f,g \in \mathcal{C}$ by

$$p_{\mathcal{C}}(f,g) = \sum_{n=1}^{\infty} 2^{-n} \max\left(\frac{1}{f(n)}, \frac{1}{g(n)}\right),$$

Matthews fixed-point theorem cannot be directly applied to a discussion of the complexity class of the running time of algorithms through $(\mathcal{C},p_{\mathcal{C}})$. Concretely, in spite of the partial metric space $(\mathcal{C}_{b,c},p_{\mathcal{C}_{b,c}})$ being complete, it was proved that the condition

$$p_{\mathcal{C}}|_{\mathcal{C}_{b,c}}\big(\Phi_T(f),\Phi_T(g)\big) \le sp_{\mathcal{C}}|_{\mathcal{C}_{b,c}}(f,g)$$

does not hold for any $s \in [0,1)$, where Φ_T is the functional associated to the recurrence equation (2) and introduced in Section 2.2. Hence, the Matthews fixed-point theorem cannot be applied to an analysis of the running time of computing of the Divide and Conquer algorithms whose running time obeys equation (2).

At this point, it seems natural to wonder if another choice of the partial metric could allow to use Matthews fixed-point theorem for the purpose of discussing running time of computing in asymptotic complexity analysis of algorithms. However, it is interesting to note that the partial metric $p_{\mathcal{C}}$, as defined before, is the most natural partial metric that

can be defined on the set $\mathcal{C}$ because $d_{\mathcal{C}}$ is induced by $p_{\mathcal{C}}$ in the sense that, for all $f, g \in \mathcal{C}$,

$$d_{\mathcal{C}}(f,g) = p_{\mathcal{C}}(f,g) - p_{\mathcal{C}}(f,f)$$

and, hence, $\mathcal{T}(d_{\mathcal{C}}) = \mathcal{T}(p_{\mathcal{C}})$.

Inspired by the fact that Matthews fixed-point theorem does not constitute a suitable tool for the aforementioned purpose, a thorough study of the possibility of applying the partial metric $p_{\mathcal{C}}$ to asymptotic complexity analysis of algorithms has been done recently in [10]. In particular, new fixed-point theorems which differ from the Matthews one have been proved in such a way that they provide a new mathematical basis to carry out an asymptotic complexity analysis of algorithms via partial metrics (concretely via $p_{\mathcal{C}}$).

2.4 The Baire partial quasi-metric space approach

Since Theorem 3 cannot be used to analyze the complexity of those algorithms whose running time of computing is the solution to a recurrence equation (2), the Baire partial quasi-metric space was introduced in [4]. This new mathematical structure allows us to carry out the asymptotic complexity analysis of algorithms in the spirit of Schellekens but now involving the original fixed-point arguments of Matthews.

In order to introduce the aforementioned Baire partial quasi-metric space and the associated fixed-point technique for complexity analysis developed in [4], let us first recall some basics on partial quasi-metrics.

Following [11], a partial quasi-metric space is a pair (X, q) where X is a non-empty set and q is a partial quasi-metric on X. By a partial quasi-metric we mean a function $q : X \times X \to \mathbb{R}^+$ such that for all $x, y, z \in X$:

(i) $q(x,x) \leq q(x,y)$;
(ii) $q(x,x) \leq q(y,x)$;
(iii) $q(x,y) \leq q(x,z) + q(z,y) - q(z,z)$;
(iv) $q(x,x) = q(x,y)$ and $q(y,y) = q(y,x) \Leftrightarrow x = y$.

Of course, a partial metric on a set X is a partial quasi-metric satisfying in addition the condition:

(v) $q(x,y) = q(y,x)$ for all $x, y \in X$.

Similarly to the case of partial metric spaces a partial quasi-metric q generates a T_0-topology $\mathcal{T}(q)$ on X which has as a base the family of open q-balls $\{B_q(x,\varepsilon) : x \in X, \varepsilon > 0\}$, where $B_q(x,\varepsilon) = \{y \in X : q(x,y) < q(x,x) + \varepsilon\}$ for all $x \in X$ and $\varepsilon > 0$.

On account of [11], and similarly to the partial metric case, each partial quasi-metric q on X induces a quasi-metric $d_q : X \times X \to \mathbb{R}^+$ in the following way:

$$d_q(x,y) = q(x,y) - q(x,x)$$

for all $x, y \in X$. Moreover, a partial quasi-metric space (X, q) is said to be complete provided that the associated quasi-metric space (X, d_q) is bicomplete.

The Matthews fixed-point theorem, Theorem 3 in Section 2.3, was extended to the context of partial quasi-metric spaces as follows [11].

Theorem 4 *Let f be a mapping from a complete partial quasi-metric space (X,q) into itself such that there is $s \in [0,1)$, satisfying*

$$q\big(f(x),f(y)\big) \leq sq(x,y), \tag{5}$$

for all $x,y \in X$. Then f has a unique fixed point. Moreover if $x \in X$ is the fixed point of f, then $q(x,x) = 0$.

It is worth to point out that generalizations of the preceding result have been obtained for a few type of contractions in partial quasi-metrics spaces in [12] and [13], recently.

In the light of the exposed notions, and according to [4], the Baire partial quasi-metric space can be introduced as follows.

Let Σ be a non-empty alphabet endowed with a partial order $\preceq$. Denote by Σ^∞ the set of all finite and infinite sequences (words) over Σ. Moreover, if $x \in \Sigma^\infty$ denote by $l(x)$ the length of x ($l(x) \in [1,\infty]$). Furthermore, given $x,y \in \Sigma^\infty$, we will say that x is a subprefix of y, denoted by $x \sqsubseteq_{sp} y$, provided that $l(x) \leq l(y)$ and $x_k \preceq y_k$ for all $k \leq l(x)$.

Set $l_\preceq(x,y) = \sup\{n \in \mathbb{N} : x_k \preceq y_k \text{ for all } k \leq n\}$ whenever there exists $n_0 \in \mathbb{N}$ such that $n_0 \leq l(x)$, $x_k \preceq y_k$ for all $k \leq n_0$, and $l_\preceq(x,y) = 0$ otherwise. Then the Baire partial quasi-metric space is the complete partial quasi-metric space (Σ^∞, q_B), where $q_B : \Sigma^\infty \times \Sigma^\infty \to \mathbb{R}^+$ is defined by

$$q_B(x,y) = 2^{-l_\preceq(x,y)}$$

for all $x,y \in \Sigma^\infty$.

As announced before, the Baire partial quasi-metric was introduced in order to apply, in some sense, partial metric fixed-point arguments to asymptotic complexity analysis of algorithms. Concretely, the new partial quasi-metric structure was applied successfully to discussion of the asymptotic complexity of algorithms whose running time of computing is typically given by the recurrence equation

$$T(n) = \begin{cases} c & \text{if } n = 1, \\ T(n-1) + h(n) & \text{if } n \geq 2, \end{cases} \tag{6}$$

with $c > 0$ and $h \in \mathcal{RT}$ such that $h(n) < \infty$ for all $n \in \mathbb{N}$.

More specifically, the utility in asymptotic complexity analysis of the fixed-point technique developed in [4] lies in the following result.

Theorem 5 *Let $\Sigma = (0,\infty]$. Fix $c \in \Sigma$ and $z \in \Sigma^\infty$ with $l(z) = \infty$ and $z_k \neq \infty$ for all $k \in \mathbb{N}$ with $k \geq 2$. Let Σ_c^∞ be the subset of Σ^∞ given by $\Sigma_c^\infty := \{y \in \Sigma^\infty : 1 \leq l(y) \text{ and } y_1 = c\}$ and let $\Phi_z : \Sigma_c^\infty \to \Sigma_c^\infty$ be the mapping defined by*

$$\big(\Phi_z(x)\big)_m := \begin{cases} c & \text{if } k = 1, \\ x_{m-1} + z_m & \text{if } 2 \leq m \leq l(x)+1. \end{cases}$$

Then Φ_z has a unique fixed point $w \in \Sigma_c^\infty$. Moreover, $l(w) = \infty$. Furthermore, if $u \in \Sigma_c^\infty$ such that $\Phi_z(u) \sqsubseteq_{sp} u$ then $w \sqsubseteq_{sp} u$.

The new mathematical technique that follows from Theorem 5 is provided by the result below.

Corollary 6 *Let* $\mathcal{RT}_c = \{f \in \mathcal{RT} : f(1) = c\}$ *and let* $\Gamma_T : \mathcal{C}_c \to \mathcal{C}_c$ *be the functional associated to recurrence equation* (6) *and given by*

$$\Gamma_T(f)(n) = \begin{cases} c & \text{if } n = 1, \\ f(n-1) + h(n) & \text{if } n \geq 2, \end{cases} \tag{7}$$

for all $f \in \mathcal{RT}$. *Then the following assertions hold*:

(1) *A recurrence equation of the form* (6) *has a unique solution* $f_T \in \mathcal{RT}_c$.

(2) *If there exists* $g \in \mathcal{RT}_c$ *such that* $\Gamma_T(g) \leq g$, *then* $f_T \in \mathcal{O}(g)$.

In the light of the preceding results, it should be pointed out that they allow one to provide, via fixed-point techniques, an asymptotic upper bound of the running time of computing of those algorithms under consideration but not its complexity class (according to the Schellekens approach exposed in Section 2.2). Moreover, they allow to achieve this without assuming the condition '$\sum_{n=1}^{\infty} 2^{-n} \frac{1}{f(n)} < \infty$ for all $f \in \mathcal{RT}$', which is an advantage with respect to the Schellekens approach.

3 The new results

It is clear that the running time of computing of all recursive algorithms does not satisfy necessarily the recurrence equation (6). In fact, there are recursive algorithms whose running time of computing obeys the recurrence equation

$$T(n) = \begin{cases} c_n & \text{if } 1 \leq n \leq k, \\ \sum_{i=1}^{k} a_i T(n-i) + h(n) & \text{if } n > k, \end{cases} \tag{8}$$

where $h \in \mathcal{RT}$ such that $h(n) < \infty$ for all $n \in \mathbb{N}$, $k \in \mathbb{N}$, $c_i > 0$, and $a_i \geq 1$ for all $1 \leq i \leq k$.

Two well-known examples of recursive algorithms whose running time satisfies the preceding recurrence equation are the algorithm that solves the Towers of Hanoi puzzle, which we will call Hanoi, and the algorithm that computes the value of the Fibonacci sequence at any given index n with $n \in \mathbb{N}$, which we will call Fibonacci. Concretely, the running time of computing of Hanoi satisfies the next recurrence equation

$$T(n) = \begin{cases} c & \text{if } n = 1, \\ 2T(n-1) + d & \text{if } n \geq 2, \end{cases} \tag{9}$$

where $c, d > 0$. Moreover, the running time of computing of Fibonacci satisfies the recurrence equation

$$T(n) = \begin{cases} 2c & \text{if } n = 1, \\ 3c & \text{if } n = 2, \\ T(n-1) + T(n-2) + 4c & \text{if } n > 2, \end{cases} \tag{10}$$

where $c > 0$.

Obviously, the recurrence equations (9) and (10) can be retrieved as a particular case of the recurrence equation (8). Observe, also, that the recurrence equation (6) can be retrieved from the recurrence equation (8). For a fuller treatment of Hanoi and Fibonacci we refer the reader to [14].

In [15] and [16], the Schellekens approach, exposed in Section 2.2, was extended and a new fixed-point technique based on the use of the (quasi-metric) complexity space was introduced in order to provide the complexity class of those recursive algorithms with running time satisfying the recurrence equation (8). The aforesaid technique was applied successfully to yield lower and upper asymptotic complexity bounds of Hanoi and Fibonacci in the same references. However, such a quasi-metric fixed-point technique involves arduous computations (see, for instance, Theorem 5 in [15] and Theorem 5 in [16]) when compared with the technique based on the Baire partial quasi-metric and given by Theorem 5.

Motivated, on the one hand, by the fact that Theorem 5 is not able to provide the complexity class of algorithms like Hanoi and Fibonacci and, on the other hand, by the fact that the Baire partial quasi-metric allows one to develop fixed-point techniques that require fewer and easier calculations than those involved by the Schellekens approach and exposed in [15] and [16], in this section we introduce a new mathematical technique based on the Baire partial quasi-metric, given by Theorem 9, which extends the technique provided by Theorem 5 and, in addition, allows to provide easily, by means of Theorem 11, the asymptotic class of those recursive algorithms whose running time satisfies the recurrence equation (8).

3.1 The new fixed-point technique

In order to present the announced technique we need to introduce the following lemmata.

Lemma 7 *Let* $\Sigma = (0, \infty]$, $k \in \mathbb{N}$ *and* $c_1, c_2, \ldots, c_k \in \Sigma$. *Let* $\Sigma^{\infty}_{c,k}$ *be the subset of* Σ^{∞} *given by* $\Sigma^{\infty}_{c,k} := \{y \in \Sigma^{\infty} : k \leq l(y)$ *and* $y_m = c_m$ *with* $1 \leq m \leq k\}$. *Then* $\Sigma^{\infty}_{c,k}$ *is closed in* $(\Sigma^{\infty}, d^s_{q_B})$.

Proof Suppose that $(u^n)_{n\in\mathbb{N}}$ is a sequence in $\Sigma^{\infty}_{c,k}$ which converges to w in $(\Sigma^{\infty}, d^s_{q_B})$. First of all we prove that $k \leq l(w)$. Indeed, assume that $l(w) < k$. Then taking $\varepsilon = 2^{-l(w)} - 2^{-k}$ we find that there exists $n_0 \in \mathbb{N}$ such that

$$2^{-l_{\preceq}(u^n,w)} - 2^{-l(u^n)} < \varepsilon \tag{11}$$

for all $n \geq n_0$. Since $k \leq l(u^n)$ and $l_{\preceq}(u^n, w) \leq l(w)$ for all $n \geq n_0$ we obtain from inequality (11)

$$2^{-l(w)} - 2^{-k} \leq 2^{-l_{\preceq}(u^n,w)} - 2^{-l(u^n)} < \varepsilon,$$

which is a contradiction. So $k \leq l(w)$.

Next we show that $w_m = c_m$ for all $1 \leq m \leq k$. To this end, assume that there exists $k_0 \in \mathbb{N}$ with $k_0 < k$ such that $w_m = c_m$ for all $m \leq k_0$ and $w_{k_0+1} \neq c_{k_0+1}$. We distinguish two possible cases:

Case 1. $w_{k_0+1} < c_{k_0+1}$. It is clear that $2^{-l_{\preceq}(w,u^n)} = 2^{-k_0}$ for all n. Taking $\varepsilon = 2^{-k_0} - 2^{-l(w)}$ we have guaranteed the existence of $n_0 \in \mathbb{N}$ such that

$$2^{-l_{\preceq}(w,u^n)} - 2^{-l(w)} < \varepsilon \tag{12}$$

for all $n \geq n_0$, since $(u^n)_{n\in\mathbb{N}}$ converges to w in $(\Sigma^\infty, d^s_{q_B})$. Hence

$$2^{-k_0} - 2^{-l(w)} = 2^{-l_\preceq(w,u^n)} - 2^{-l(w)} < \varepsilon$$

which is impossible.

Case 2. $w_{k_0+1} > c_{k_0+1}$. Then, given $\varepsilon = 2^{-k}$, there exists $n_0 \in \mathbb{N}$ such that

$$2^{-l_\preceq(u^n,w)} - 2^{-l(u^n)} < \varepsilon \tag{13}$$

for all $n \geq n_0$. Whence

$$2^{-k_0} - 2^{-l(u^n)} < 2^{-k}$$

for all $n \geq n_0$. Hence

$$2^{-k_0} < 2^{-k} + 2^{-l(u^n)} \leq 2^{-k+1}.$$

It follows that $k + 1 \leq k_0 < k$, which is impossible.

Consequently $w_m = c_m$ for all $1 \leq m \leq k$ and, thus, $w \in \Sigma^\infty_{c,k}$. Therefore $\Sigma^\infty_{c,k}$ is closed in $(\Sigma^\infty, d^s_{q_B})$. □

Lemma 8 *Let f be a mapping from a complete partial quasi-metric space (X, q) into itself such that there is $s \in [0, 1[$ satisfying*

$$q\big(f(x), f(y)\big) \leq sq(x, y), \tag{14}$$

for all $x, y \in X$. Then f has a unique fixed point x. Moreover, the following assertions hold:

(1) *If there exists $y \in X$ such that $q(f(y), y) = 0$ then $q(x, y) = 0$.*

(2) *If there exists $y \in X$ such that $q(y, f(y)) = 0$ then $q(y, x) = 0$.*

Proof First of all we note that the existence and uniqueness of the fixed point x of f is guaranteed by Theorem 4.

(1) Assume for the purpose of contradiction that $q(x, y) > 0$. Then the inequality (14) yields

$$\begin{aligned} q(x, y) &\leq q\big(x, f(y)\big) + q\big(f(y), y\big) - q\big(f(y), f(y)\big) \\ &\leq q\big(x, f(y)\big) \\ &= q\big(f(x), f(y)\big) \\ &\leq sq(x, y). \end{aligned}$$

It follows that $1 \leq s < 1$, which is a contradiction.

(2) Similar arguments to those given in the proof of (1) apply to the proof of assertion (2). □

The next result provides the mathematical foundations for the fixed-point technique useful for analyzing the complexity class of recursive algorithms whose running time of computing satisfies the recurrence equation (8).

Theorem 9 *Let* $\Sigma = (0,\infty]$ *and* $k \in \mathbb{N}$. *Fix* $a_1, c_1, a_2, c_2, \ldots, a_k, c_k \in \Sigma$ *and* $z \in \Sigma^\infty$ *with* $l(z) = \infty$ *and* $z_m \neq \infty$ *for all* $m \in \mathbb{N}$ *and* $m \geq k+1$. *Let* $\Phi_z : \Sigma^\infty_{c,k} \to \Sigma^\infty_{c,k}$ *be the mapping defined by*

$$\big(\Phi_z(x)\big)_m := \begin{cases} c_m & \text{if } 1 \leq m \leq k, \\ \sum_{i=1}^{k} a_i x_{m-i} + z_m & \text{if } k+1 \leq m \leq l(x)+1. \end{cases}$$

Then the following assertions hold:

(1) Φ_z *has a unique fixed point* $w \in \Sigma^\infty_{c,k}$.
(2) $l(w) = \infty$.
(3) *If* $u \in \Sigma^\infty_{c,k}$ *such that* $\Phi_z(u) \sqsubseteq_{sp} u$, *then* $w \sqsubseteq_{sp} u$.
(4) *If* $v \in \Sigma^\infty_{c,k}$, *then* $v \sqsubseteq_{sp} w$ *provided that* $l(v) = \infty$ *and* $v \sqsubseteq_{sp} \Phi_z(v)$.

Proof The completeness of the Baire partial quasi-metric space (Σ^∞, q_B) shows that the quasi-metric space (Σ^∞, d_{q_B}) is bicomplete. Moreover, by Lemma 7, we see that the set $\Sigma^\infty_{c,k}$ is closed in $(\Sigma^\infty, d^s_{q_B})$. Thus the quasi-metric space $(\Sigma^\infty_{c,k}, d_{q_B})$ is bicomplete. Whence we deduce that the partial quasi-metric space $(\Sigma^\infty_{c,k}, q_B)$ is complete. Furthermore, it is a straightforward computation to check that the mapping $\Phi_z : \Sigma^\infty_{c,k} \to \Sigma^\infty_{c,k}$ obeys the following inequality:

$$q_B\big(\Phi_z(u), \Phi_z(v)\big) \leq \frac{1}{2^{k+1}} \leq \frac{1}{2} q_B(u,v)$$

for all $u, v \in \Sigma^\infty_{c,k}$. It follows, by Theorem 4, that Φ_z has a unique fixed point $w \in \Sigma^\infty_{c,k}$ with $q_B(w,w) = 0$. Hence $l(w) = \infty$. So we have proved statements (1) and (2).

Now assume the existence of $u \in \Sigma^\infty_{c,k}$ such that $\Phi_z(u) \sqsubseteq_{sp} u$. It follows, by construction of Φ_z, that $l(u) = l(\Phi_z(u)) = \infty$. Since $\Phi_z(u) \sqsubseteq_{sp} u$ we obtain $q_B(\Phi_z(u), u) = 0$. Hence, by assertion (1) in statement of Lemma 8, we find that $q_B(w,u) = 0$. Whence we deduce that $l_{\preceq}(w,u) = \infty$ and, hence, that $w \sqsubseteq_{sp} u$. This proves statement (3).

Finally, assume the existence of $v \in \Sigma^\infty_{c,k}$ such that $l(v) = \infty$. It follows that $l(\Phi_z(v)) = \infty$. Since $v \sqsubseteq_{sp} \Phi_z(v)$ we deduce that $q_B(v, \Phi_z(v)) = 0$. Hence, by assertion (2) in statement of Lemma 8, we find that $q_B(v,w) = 0$. Whence we deduce that $v \sqsubseteq_{sp} w$. This proves statement (4). □

We must emphasize that taking $k = 1$ and $a_1 = 1$ in statement of Theorem 9 we retrieve, as a particular case, Theorem 5. Moreover, setting $k = 1$ and $a_1 = a$ for any $a \in \mathbb{N}$ with $a > 1$ in statement of Theorem 9 we obtain the following corollary.

Corollary 10 *Let* $\Sigma = (0,\infty]$. *Fix* $a, c \in \Sigma$ *and* $z \in \Sigma^\infty$ *with* $a > 1$, $l(z) = \infty$ *and* $z_k \neq \infty$ *for all* $k \in \mathbb{N}$ *and* $k \geq 2$. *Let* Σ^∞_c *be the subset of* Σ^∞ *given by* $\Sigma^\infty_c := \{y \in \Sigma^\infty : 1 \leq l(y)$ *and* $y_1 = c\}$ *and let* $\Phi_z : \Sigma^\infty_c \to \Sigma^\infty_c$ *be the mapping defined by*

$$\big(\Phi_z(x)\big)_m := \begin{cases} c & \text{if } m = 1, \\ a x_{m-1} + z_m & \text{if } 2 \leq m \leq l(x)+1. \end{cases}$$

Then the following assertions hold:

(1) Φ_z *has a unique fixed point* $w \in \Sigma^\infty_c$.
(2) $l(w) = \infty$.

(3) *If* $u \in \Sigma_c^\infty$ *such that* $\Phi_z(u) \sqsubseteq_{sp} u$, *then* $w \sqsubseteq_{sp} u$.
(4) *If* $v \in \Sigma_c^\infty$, *then* $v \sqsubseteq_{sp} w$ *provided that* $l(v) = \infty$ *and* $v \sqsubseteq_{sp} \Phi_z(v)$.

Next we want to emphasize that the Divide and Conquer recurrence equation (2) can be transformed into the recurrence equation

$$S(m) = \begin{cases} c & \text{if } m = 1, \\ aS(m-1) + r(m) & \text{if } m > 1, \end{cases} \tag{15}$$

where $S(m) = T(b^{m-1})$ and $r(m) = h(b^{m-1})$ for all $m \in \mathbb{N}$ (recall that $\mathbb{N}_b = \{b^k : k \in \mathbb{N}\}$ with $b \in \mathbb{N}$ and $b > 1$).

Therefore Corollary 10, and with it Theorem 9, retrieve as a particular case Theorem 6 given in [4], which was obtained with the aim of analyzing the asymptotic complexity of those algorithms with running time of computing satisfying the recurrence equation (2) by means of fixed-point arguments based on the Baire partial quasi-metric space. In this direction, Theorem 9 allows us to unify Theorem 5 and Theorem 6 and, hence, to obtain a unique fixed-point technique, via the Baire partial quasi-metric space, for the mathematical foundation of the asymptotic complexity analysis of those algorithms whose running time leads in a natural way to recurrence equations of the type of equations (2), (6), and (16), where

$$T(n) = \begin{cases} c & \text{if } n = 1, \\ aT(n-1) + h(n) & \text{if } n \geq 2, \end{cases} \tag{16}$$

with $a > 1$, $c > 0$, and $h \in \mathcal{RT}$ such that $h(n) < \infty$ for all $n \in \mathbb{N}$.

Notice that the recurrence equation (9) associated to Hanoi is a concrete case of equation (16). Besides, celebrated examples of algorithms whose running time of computing lead to the recurrence equations (2) and (16) are, for instance, Mergesort and Quicksort (for more details see [5] and [14]).

3.2 The application to asymptotic complexity analysis

Notice that Theorem 9 yields a new fixed-point technique to analyze the complexity class of all those algorithms whose running time of computing satisfies the recurrence equation (8). Indeed, we associate to the recurrence equation (8) the functional $\Gamma_T : \mathcal{RT}_{c,k} \to \mathcal{RT}_{c,k}$ defined by

$$\Gamma_T(f)(n) = \begin{cases} c_n & \text{if } 1 \leq n \leq k, \\ \sum_{i=1}^{k} a_i f(n-i) + h(n) & \text{if } n > k. \end{cases} \tag{17}$$

Then Theorem 9 provides, in a very easy way without involving arduous computations, the promised asymptotic complexity technique, Theorem 11, to determine the asymptotic complexity (upper and lower) bounds, and thus the complexity class, of the running time of those algorithms under study.

Theorem 11 *A recurrence equation of the form* (8) *has a unique solution* $f_T \in \mathcal{RT}_{c,k}$. *Moreover, if there exist* $g, h \in \mathcal{RT}_{c,k}$ *such that* $\Gamma_T(g) \leq g$ *and* $h \leq \Gamma_T(h)$, *then* $f_T \in \Omega(h) \cap \mathcal{O}(g)$. *Furthermore,* $f_T \in \Theta(l)$ *provided that there exist* $c, d > 0$ *and* $l \in \mathcal{RT}_{c,k}$ *in such a way that the preceding functions* g, h *satisfy* $h(n) = cl(n)$ *and* $g(n) = dl(n)$ *for all* $n \in \mathbb{N}$.

Proof Let $w \in \Sigma_{c,k}^{\infty}$ be the fixed point of the mapping Φ_z ensured by Theorem 9. Now, define $f_w \in \mathcal{RT}_{c,k}$ by $f_w(n) = w_n$ for all $n \in \mathbb{N}$. Then we immediately see that f_w is the unique solution to the recurrence equation (8). So f_w can be identified with the running time of computing of a recursive algorithm satisfying the recurrence equation (8), say f_T. Of course, $f_T(n) = f_w(n) = w_n$ for all $n \in \mathbb{N}$.

Next assume that there exists $g \in \mathcal{RT}_{c,k}$ such that $\Gamma_T(g) \preceq g$. Then we identify such a function with a word $u^g \in \Sigma_{c,k}^{\infty}$ which is defined by $u_n^g = g(n)$ for all $n \in \mathbb{N}$. It is obvious that $\Phi_z(u^g) \sqsubseteq_{sp} u^g$. It follows, by Theorem 9, that $w \sqsubseteq_{sp} u^g$. Hence we have showed that $f_w \in \mathcal{O}(g)$ or, equivalently, that $f_T \in \mathcal{O}(g)$.

Now suppose that there exists $h \in \mathcal{RT}_{c,k}$ such that $h \preceq \Gamma_T(h)$. Then we identify such a function with a word $v^h \in \Sigma_{c,k}^{\infty}$ which is defined by $v_n^h = h(n)$ for all $n \in \mathbb{N}$. Clearly $l(v^h) = \infty$ and $v^h \sqsubseteq_{sp} \Phi_z(v^h)$. By Theorem 9 we obtain $v^h \sqsubseteq_{sp} w$. Whence we deduce that $h \preceq f_w$ and, hence, that $f_w \in \Omega(h)$ or, equivalently, that $f_T \in \Omega(h)$.

Finally, assume that there exist $c, d > 0$ and $l \in \mathcal{RT}_{c,k}$ such that the function g, h satisfy $g(n) = cl(n)$ and $h(n) = dl(n)$ for all $n \in \mathbb{N}$. Since $\Gamma_T(g) \preceq g$ and $h \preceq \Gamma_T(h)$ we have $f_w \in \Omega(h) \cap \mathcal{O}(g)$. It follows that $f_w \in \Omega(l) \cap \mathcal{O}(l)$. Consequently $f_w \in \Theta(l)$. Thus $f_T \in \Theta(l)$. □

Observe that, although Theorem 5 can be obtained from Theorem 9, the latter gives a few more information about the running time of computing under consideration than the former. This is due to the fact that the assertion (4) in statement of Theorem 9 allows to determine the lower asymptotic complexity bound for algorithms whose running time obeys equation (6) while that Theorem 5 does not allow one to do this. Consequently, the version of Theorem 5 retrieved from Theorem 9 improves Theorem 5. So the new fixed-point result is clearly a non-trivial extension of the old one.

Furthermore, it must be stressed that a first attempt to apply fixed-point techniques based on the use of words to complexity analysis of algorithms was made in [17] by means of the so-called Baire balanced quasi-metric. However, the aforementioned quasi-metric fixed-point technique was only able to provide the existence and uniqueness of solution to recurrence equations and not the complexity class. So, in some sense, our new technique improves those given in [17], since Theorem 9 allows to obtain the asymptotic complexity class of the solution to a recurrence equation associated to an algorithm and, thus, the complexity class of its running time of computing.

In order to help the reader to glimpse the utility of Theorem 11, and with it also of Theorem 9, we end the paper showing that, like the Schellekens approach, the developed theory is helpful in providing the asymptotic complexity class of the recursive algorithms whose running time satisfies the recurrence equation (8). In particular, we apply Theorem 11 to the analysis of Hanoi and Fibonacci, retrieving their well-known asymptotic complexity class [14].

Hanoi As mentioned before, the running time of computing of Hanoi matches up with the solution, under the uniform cost criterion assumption (see [14]), to the recurrence equation (9). Of course, such a recurrence equation can be retrieved from equation (8) as a particular case when we put $k = 1$, $c_1 = c$, $a_1 = 2$ and $h(n) = d$ for all $n \in \mathbb{N}$. Consider the functional associated with equation (9) and given by equation (17) as follows:

$$\Gamma_T^H(f)(n) = \begin{cases} c & \text{if } n = 1, \\ 2f(n-1) + d & \text{if } n \geq 2, \end{cases} \tag{18}$$

for all $f \in \mathcal{RT}_{c,1}$. Then Theorem 11 yields the existence and uniqueness of the solution f_T^H (in $\mathcal{RT}_{c,1}$), which obviously represents the running time of computing of Hanoi, to the above recurrence equation.

It is a straightforward computation to check that $\Gamma_T^H(g) \preceq g$ for any $g \in \mathcal{RT}_{c,1}$ if and only if $g(1) = c$ and $g(n) \geq 2^{n-1}(d+c) - d$ for all $n \in \mathbb{N}$ with $n \geq 2$. Thus, by Theorem 11, we deduce that $f_T^H \in \mathcal{O}(g)$, where

$$g(n) = \begin{cases} c & \text{if } n = 1, \\ 2^{n-1}(d+c) - d & \text{if } n \geq 2. \end{cases} \tag{19}$$

Moreover, it is not hard to check that $h \preceq \Gamma_T^H(h)$ for any $h \in \mathcal{RT}_{c,1}$ if and only if $h(1) = c$ and $h(n) \leq 2^{n-1}(d+c) - d$ for all $n \in \mathbb{N}$ with $n \geq 2$. Thus, by Theorem 11, we deduce that $f_T^H \in \Omega(h)$, where

$$h(n) = \begin{cases} c & \text{if } n = 1, \\ 2^{n-1}(d+c) - d & \text{if } n \geq 2. \end{cases} \tag{20}$$

Accordingly, by Theorem 11, we obtain $f_T^H \in \Omega(h) \cap \mathcal{O}(g)$ with g and h given by equations (19) and (20), respectively. Since $g = h$ we conclude, by Theorem 11, that $f_T^H \in \Theta(g)$. Therefore we have proved the following result.

Corollary 12 *The complexity class of the running time of computing of Hanoi is* $\Theta(2^{n-1})$.

Fibonacci As we have exposed above, the running time of computing of Fibonacci is the solution to the recurrence equation (10). Clearly, such a recurrence equation can be retrieved from equation (8) as a particular case when we put $k = 2$, $a_1 = a_2 = 1$, $c_1 = 2c$, $c_2 = 3c$ and $h(n) = 4c$ for all $n \in \mathbb{N}$. Consider the functional associated with equation (10) and given by equation (17) as follows:

$$\Gamma_T^F(f)(n) = \begin{cases} 2c & \text{if } n = 1, \\ 3c & \text{if } n = 2, \\ f(n-1) + f(n-2) + 4c & \text{if } n > 2, \end{cases}$$

for all $f \in \mathcal{RT}_{c,2}$. Then Theorem 11 yields the existence and uniqueness of the solution f_T^F (in $\mathcal{RT}_{c,2}$), which obviously represents the running time of computing of Fibonacci, to the above recurrence equation.

In order to guess the asymptotic bounds we fix $r > 0$ and $\alpha > 0$ and define the function $g_{\alpha,r} \in \mathcal{RT}_{c,2}$ by

$$g_{\alpha,r}(n) = \begin{cases} 2c & \text{if } n = 1, \\ 3c & \text{if } n = 2, \\ r\alpha^n & \text{if } n > 2. \end{cases}$$

In what follows we denote the value $\frac{1+\sqrt{5}}{2}$ by ϕ. It is a straightforward computation to check that $\Gamma_T^F(g_{\alpha,r}) \preceq g_{\alpha,r}$ if and only if $\alpha > \phi$ and $r \geq \frac{4c}{\phi^3 - \phi^2 - \phi}$. Then, by Theorem 11, $f_T^H \in$

$\mathcal{O}(g_{\alpha,r})$ for all $\alpha > \phi$ and $r \geq \frac{4c}{\phi^3-\phi^2-\phi}$. Hence, taking $\alpha = 1.619$, we immediately have $f_T^F \in \mathcal{O}(g_{1.619,\frac{4c}{\phi^3-\phi^2-\phi}})$.

Moreover, it is not hard to check that $g_{\alpha,r} \leq \Gamma_T^F(g_{\alpha,r})$ if and only if $\alpha \leq \phi$ for all $r > 0$. It follows, by Theorem 11, that $f_T^F \in \Omega(g_{\alpha,r})$ for all $\alpha \leq \phi$ and all $r > 0$. Hence, taking $\alpha = \phi$, we obtain $f_T^F \in \Omega(g_{\phi,r})$.

Consequently, by Theorem 11, we obtain $f_T^F \in \Omega(g_{\phi,r}) \cap \mathcal{O}(g_{1.619,\frac{4c}{\phi^3-\phi^2-\phi}})$ for all $r > 0$. Therefore we have proved the following result.

Corollary 13 *The running time of computing of Fibonacci belongs to the complexity class* $\Omega((1.618)^n) \cap \mathcal{O}((1.619)^n)$.

Observe that the fact that the running time of computing is in $\Omega((1.618)^n) \cap \mathcal{O}((1.619)^n)$ guarantees that any other function, different from $g_{\alpha,r}$, chosen to determine the complexity class of f_T^F would not be able to provide the suitable lower and upper asymptotic bounds at the same time.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
All authors contributed equally and took part in every discussion. All authors read and approved the final manuscript.

Author details
[1]Department of Mathematics, Sciences Faculty for Girls, King Abdulaziz University, P.O. Box 4087, Jeddah, 21491, Saudi Arabia. [2]Department of Mathematics, King Abdulaziz University, P.O. Box 80203, Jeddah, 21859, Saudi Arabia. [3]Departamento de Ciencias Matemáticas e Informática, Universidad de las Islas Baleares, Ctra. de Valldemossa km. 7.5, Palma de Mallorca, 07122, Spain.

Acknowledgements
This work was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. 363-002-D1434. The authors, therefore acknowledge with thanks DSR for technical and financial support.

Received: 10 September 2013 Accepted: 12 December 2013 Published: 20 January 2014

References

1. Matthews, SG: Partial metric topology. Ann. N.Y. Acad. Sci. **728**, 183-197 (1994)
2. Matthews, SG: An extensional treatment of lazy data flow deadlock. Theor. Comput. Sci. **151**, 195-205 (1995)
3. Schellekens, MP: The Smyth completion: a common foundation for denotational semantics and complexity analysis. Electron. Notes Theor. Comput. Sci. **1**, 211-232 (1995)
4. Cerdà-Uguet, MA, Schellekens, MP, Valero, O: The Baire partial quasimetric space: a mathematical tool for the asymptotic complexity analysis in computer science. Theory Comput. Syst. **50**, 387-399 (2012)
5. Brassard, G, Bratley, P: Algorithms: Theory and Practice. Prentice Hall, New Jersey (1988)
6. Künzi, HPA: Nonsymmetric distances and their associated topologies: about the origins of basic ideas in the area of asymmetric topology. In: Aull, CE, Lowen, R (eds.) Handbook of the History of General Topology, vol. 3, pp. 853-968. Kluwer Academic, Dordrecht (2001)
7. Romaguera, S, Schellekens, MP: Quasi-metric properties of complexity spaces. Topol. Appl. **98**, 311-322 (1999)
8. Scott, DS: Outline of a mathematical theory of computation. In: Proc. 4th Annual Princeton Conference on Information Sciences and Systems, pp. 169-176 (1970)
9. Oltra, S, Romaguera, S, Sánchez-Pérez, EA: Bicompleting weightable quasi-metric spaces and partial metric spaces. Rend. Circ. Mat. Palermo **51**, 151-162 (2002)
10. Alghamdi, MA, Shahzad, N, Valero, O: Fixed point theorems in generalized metric spaces with applications to computer science. Fixed Point Theory Appl. **2013**, Article ID 118 (2013)
11. Künzi, HPA, Pajooshesh, H, Schellekens, MP: Partial quasi-metrics. Theor. Comput. Sci. **365**, 237-246 (2006)
12. Karapinar, E, Erhan, IM, Öztürk, A: Fixed point theorems on quasi-partial metric spaces. Math. Comput. Model. **57**, 2442-2448 (2013)
13. Shoaib, A, Arshad, M, Ahmad, J: Fixed point results of locally contractive mappings in ordered quasi-partial metric spaces. Sci. World J. **2013**, Article ID 194897 (2013)
14. Cull, P, Flahive, M, Robson, R: Difference Equations: From Rabbits to Chaos. Springer, New York (2005)
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. **89**, 1728-1741 (2012)
16. Romaguera, S, Valero, O: A common mathematical framework for asymptotic complexity analysis and denotational semantics for recursive programs based on complexity spaces. In: Afzal, MT (ed.) Semantics-Advances in Theories and Mathematical Models, vol. 1, pp. 99-120. InTech, Rijeka (2012)

17. Rodríguez-López, J, Romaguera, S, Valero, O: Denotational semantics for programming languages, balanced quasi-metrics and fixed points. Int. J. Comput. Math. **85**, 623-630 (2008)

doi:10.1186/1687-1812-2014-14
Cite this article as: Alghamdi et al.: New results on the Baire partial quasi-metric space, fixed point theory and asymptotic complexity analysis for recursive programs. *Fixed Point Theory and Applications* 2014 **2014**:14.