An extensional treatment of lazy data flow deadlock

S.G. Matthews*

Department of Computer Science, University of Warwick, Coventry, CV47AL, UK

Abstract

In an extensional treatment of dataflow deadlock Wadge (1981) introduced an elegant nonoperational test for proving that many of Kahn's data flow message passing networks (Kahn, 1974) must be free of deadlock; a test that "should extend to a much wider context" in the study of program correctness. Such a context has now been provided with the introduction of partial metric spaces (Matthews, 1992). These spaces can be used to describe semantic domains such as those used in lazy data flow languages (Wadge and Ashcroft, 1985). This paper develops Wadge's ideas on establishing an extensional theory of program correctness by using partial metric spaces to give a nonoperational treatment of lazy data flow deadlock.

1. Introduction

It has long been an accepted principal within Computer Science that any durable understanding of large complex software systems must eventually include a theory of program correctness. Such a theory would help us to either assess the correctness of existing software or, better still, tell us how to write it correctly in the first place. Hoare [3] suggested that the correctness of an Algol-like program could be understood through an input–output relation rigorously derived from the program text. In complete contrast others [7] have argued that correctness preserving formal methods can be applied to a formal specification to produce a correct system. Common to both approaches is an understanding of correctness through a rigorous mathematical/logical treatment of operational semantics. However, any programmer knows that the complexity of such semantics, as measured in the person hours needed to produce correct code, is highly disproportional to the number of lines in that code. Consequently the applicability of such operationally based approaches to correctness decreases as the software size increases.

A more pragmatic approach to correctness should be less operationally based than those above but, for reasons of efficiency, allow operational considerations when

* E-mail: sgm@dcs.warwick.ac.uk.
appropriate. In this paper we consider a correct program to be one having the correct relationships between its data objects, a familiar idea used in, for example, Hoare logics. Such an extensional theory may consist of temporal, spatial, and recursive relationships over a data domain. Wadge [10] offered an extensional treatment for the correctness property of deadlock in the context of a simple model of parallel computation. He conjectured that such a treatment of deadlock should generalise if an extensional theory of domains could first be established. In this paper we suggest such a theory using the author's partial metric [8], and then use it to give precisely the extensional treatment of deadlock conjectured to exist by Wadge.

2. Data flow networks

Kahn [5] considered an asynchronous message passing model of parallel computation consisting of deterministic sequential processes $P_0, \ldots, P_{n-1}$ communicating through uni-directional UNIX-style pipes. Each process has precisely one output pipe but may have zero to $n$ input pipes. Each computation step of a process, called a "firing", consists of the consumption of the next packet of information from an input pipe, producing a sequence of output packets on the output pipe, together with the updating of the process' internal state ready for the next firing. The behaviour of a process is precisely the sequence of its firings. The behaviour of netwok of such processes is a fair interleaving of all the process behaviours. Kahn conjectured correctly, later proved by Faustini [1], that an operational semantics could be formally defined for his networks and proved equivalent to the least fixed point of an appropriately chosen function. In order to study both Kahn's ideas and, later on, lazy data flow we first need a general definition for a data flow network.

Definition 2.1. A data flow network of $n \geq 1$ processes over a history domain $D$ (assumed to be a chain complete partial ordering with $\perp$) is tuple

$$\langle n : n \to n + 1, (\psi_i : D^n \to D | i \in n), (\varepsilon_i : \pi_i \to n | i \in n) \rangle$$

such that each $\psi_i$ is chain continuous. $\pi_i$ is the number of input pipes to process $P_i$. $\varepsilon_i$ specifies the process to which each input pipe of $P_i$ is connected, and $\psi_i$ is the function computed by $P_i$. Although the term pipe is used in this paper to refer to the uni-directional communication channels between processes it should be remembered that in some data flow networks pipes may be lazy. In a network each input pipe is some process' output pipe, however, many input pipes may receive identical copies of packets from the same output pipe. It is possible that an output pipe is not a process' input pipe, in which case the output packets from a process are not used by another process.

Definition 2.2. The computed value of a data flow network $(\pi, \psi, \varepsilon)$ of $n$ processes over a history domain $D$ is the least fixed point of the network function $f : D^n \to D^n$
where
\[ \forall X \in D^a, \forall i \in n, \ f(X)_i = \psi_i(X \circ e_i). \]

The history domain \( Ka(S) \) for a Kahn network is defined as follows. Let \( S^* \cup S^\omega \) denote the set of all finite and infinite sequences over \( S \), \( \omega \) denoting the set \( \{0, 1, \ldots \} \) of all nonnegative integers. \( \text{length}: S^* \cup S^\omega \to \omega \cup \{\omega\} \) is the function such that
\[ \text{length}(\langle x_0, x_1, \ldots \rangle) = \omega, \]
\[ \text{length}(\langle x_0, \ldots, x_{i-1} \rangle) = i. \]

**Definition 2.3.** The Kahn domain \( Ka(S) \) is the set \( S^* \cup S^\omega \) partially ordered by the initial segment relation \( \preceq \) defined by
\[ \forall x, y \in Ka(S) \quad x \preceq y \iff \text{length}(x) \leq \text{length}(y) \land \forall i < \text{length}(x), \ x_i = y_i. \]

**Definition 2.4.** A Kahn network is defined to be a data flow network over the Kahn domain, that is, a network of the form
\[ (\pi : n \to n + 1, (\psi_i : Ka(S)^* \to Ka(S) \mid i \in n), (e_i : \pi_i \to n \mid i \in n)). \]

Underlying the design of any high level model of computation are intentions as to what should constitute correct behaviour. With a Kahn network the intention is that for a computation to be correct each pipe must either have data packets eternally passing through it, or else have a finite number of packets properly terminated. Kahn's model can be formulated as one of infinite computation in which a finite terminated stream of packets can be represented as a stream having an infinite tail of end of data packets, as is done in Lucid [11]. Unfortunately though a network could live lock in which case some process fires an infinite number of times without sending anymore packets to its output pipe. Also, a network could deadlock in which case a process cannot fire because the next input needed for that firing to take place is dependent upon the output from that same firing. Clearly neither of the incorrect behaviours of live lock or deadlock can exist if there is an unending flow of packets along each and every pipe. Such a correct behaviour we term complete. Thus by proving a network to be complete (i.e. it has complete behaviour) we have proved the correctness properties of absence of live lock and absence of deadlock. This notion of a complete network can be captured precisely within Kahn's model as a network is complete if and only if its computed value is a tuple of infinite sequences.

To be able to reason whether or not a data object is complete we need first to be able to discuss the extent to which any data object is complete. We thus define the completeness of a data object to be the extent to which it is complete. In the case of the Kahn domain we use the function \( \text{length} \), where the greater the value of \( \text{length}(x) \) the greater is the completeness of \( x \). The completeness of a network can thus be studied without always having to refer to every detail of the operational semantics of interleaved firings; it should be a subject which can be explored using more abstract
tools when appropriate. Wadge [10] identified just such an instance of this phenomenon in his consideration of a class of networks (introduced in the next section) which, as he demonstrated, can be proved free of deadlock without reference to the operational semantics of fairly interleaved firings. This work suggested to Wadge that a formal theory of completeness should exist "which refers to data objects, and not computations". The contribution of the present paper is to develop a formal theory for studying the completeness relationships between data objects. In order to motivate the concepts which should be included in a formal theory of completeness we now analyse Wadge's treatment of Kahn data flow deadlock.

3. The cycle sum test

Wadge considered the deadlock properties of any Kahn network (of \( n \) processes with network function \( f \)) for which there exists \( M : n \times \omega \rightarrow \{ \ldots, -1, 0, 1, \ldots, \infty \} \) such that,

\[
\forall X \in Ka(S)^n, \quad \forall i \in n, \quad f(X)_i \geq \min_{j \in \omega} M(i, j) + \text{length}(X_j).
\]

He demonstrated that if the network passes the cycle sum test, which says that each cycle sum,

\[
M(I_0, I_1) + M(I_1, I_2) + M(I_2, I_3) + \cdots + M(I_{m-1}, I_0)
\]

(for any \( I : m \rightarrow n \)) must be greater than zero, then there can be no deadlock. To understand the test as a completeness result consider the case for \( n = 1 \), in which case \( M(0, 0) > 0 \) and,

\[
\forall x \in Ka(S), \quad \text{length}(f(x)) \geq \text{length}(x) + M(0, 0).
\]

Here the least fixed point of \( f \) can only satisfy this relation if it has length \( \infty \). Thus the cycle sum test has an instance of the following inductive principal in which we use \text{length} to measure completeness in the history domain \( Ka(S) \).

**Completeness induction.** For each domain \( D \) equipped with a suitable notion of completeness, and for each function \( f : D \rightarrow D \) such that \( f(x) \) is always more complete (i.e. less partial) than \( x \), then \( f \) has a unique fixed point, and this point is complete.

Without a formal theory of completeness it is difficult to argue that the cycle sum test is a completeness result for \( n > 1 \). More of a problem is that without such a theory it is not possible to formulate the cycle sum test for data flow networks other than those of Kahn. The lazy data flow programming language Lucid [11] is just such a case in point. Let the domain \( Lu(S) \) of lazy histories be the partial ordering over
(S ∪ {⊥})^o such that

\[ \forall X, Y \in (S \cup \{\bot\})^o, \quad X \ll Y \iff \forall i \in n, \quad X_n = \bot \lor X_n = Y_n. \]

A data flow network over Lu(S) adds the notion of lazy evaluation [2] to a network over Ka(S), allowing us to demand and compute packets in any order. The Kahn domain is embedded into the Lucid domain by the function \( e: Ka(S) \to Lu(S) \) such that

\[
e(\langle x_0, x_1, \ldots, \rangle) = \langle x_0, x_1, \ldots, \rangle,
\]

\[
e(\langle x_0, \ldots, x_{i-1} \rangle) = \langle x_0, \ldots, x_{i-1}, \bot, \bot, \ldots \rangle.
\]

The following question naturally arises. Can we extend the cycle sum test to Lu(S), and so give an extensional treatment for lazy data flow deadlock? Unfortunately the cycle sum test pivots upon the notion of length for which there is no obvious counterpart in Lu(S). As every member of Lu(S) has (trivially) infinite length it is not possible to use completeness induction using length, and so there appears to be no cycle sum test for lazy data flow. The only way out of this dilemma is to establish a formal theory of completeness which generalises the notion of length.

4. Partial metric domains

Our aim is thus to construct a theory of domains which incorporates both a partial ordering for the purposes of least fixed point semantics and a notion of completeness for using completeness induction. This combination can be achieved using the following generalised metric.

**Definition 4.1.** A partial metric (or just pmetric, pronounced “p metric”) [8] over a set \( U \) is a function \( p: U^2 \to \mathbb{R} \) such that

- (P1) \( \forall x, y \in U, \quad x = y \Rightarrow p(x, x) = p(x, y) = p(y, y), \)
- (P2) \( \forall x, y \in U, \quad p(x, x) \leq p(x, y), \)
- (P3) \( \forall x, y \in U, \quad p(x, y) = p(y, x), \)
- (P4) \( \forall x, y, z \in U, \quad p(x, z) \leq p(x, y) + p(y, z) - p(y, y). \)

The pmetric axioms (P1)–(P4)\(^1\) are intended to be a minimal generalisation of the axioms for a metric [9] such that each object does not necessarily have to have zero distance from itself. Consequently a metric is precisely a pmetric such that for each \( x \in U, \quad p(x, x) = 0. \) \( p(x, x), \) referred to as the size or weight of \( x, \) is our measure of the completeness of \( x. \) The smaller \( p(x, x) \) the more complete \( x \) is, \( x \) being complete if

\(^1\) (P4) was first suggested to the author in *Matthews Metrics*, Steve Vickers, unpublished notes, Imperial College, 1987.
$p(x, x) = 0$. The pmetric thus gives us a formal framework for discussing completeness, however, we also need a partial ordering so that we can use least fixed point semantics as well. For each pmetric $p$ over $U$ the relation $\leq_p \subseteq U^2$ such that

$$\forall x, y \in U, \quad x \leq_p y \iff p(x, x) = p(x, y)$$

is a partial ordering [8]. The intuition behind $x \leq_p y$ is that $x$ and $y$ have as much information in common as $x$ has with itself. The simplest example of partial metric (which is not a metric because $p(\bot, \bot) = 1$) is the flat pmetric $p : (S \cup \{\bot\})^2 \to \{0, 1\}$ where

$$\forall x, y \in S \cup \{\bot\}, \quad p(x, y) = 0 \iff x = y \in S.$$  

Here $\leq_p$ is the usual ordering

$$\forall x, y \in S \cup \{\bot\}, \quad x \leq y \iff x = \bot \lor x = y \in S$$

for a flat domain. We can use the following standard construction to form countable products.

**Lemma 4.1.** For each bounded pmetric $p$ over $U$ the function $p^\omega : (U^\omega)^2 \to \mathbb{R}$ where

$$\forall X, Y \in U^\omega, \quad p^\omega(X, Y) = \sum_{i \in \omega} p(X_i, Y_i) \times 2^{-i}$$

is a pmetric such that

$$\forall X \leq_p X', Y \leq_p Y' \in U^\omega, \quad p^\omega(X, Y) = p^\omega(X', Y') \iff \forall i \in \omega, p(X_i, Y_i) = p(X_i', Y_i')$$

**Proof.** Follows using (P1)–(P4). \qed

Thus,

$$\forall X, Y \in U^\omega, \quad X \leq_p Y \iff \forall i \in \omega, X_i \leq_p Y_i,$$

which is precisely the ordering used over $U = Lu(S)$ in the previous section. Similarly we can define the finite product $p^n : (U^n)^2 \to \mathbb{R}$.

The Kahn domain can be defined using a pmetric over $U = Ka(S)$. Let $p(x, y)$ be one divided by two to the power of the length of the longest sequence which is an initial segment of both $x$ and $y$. Then $\leq_p$ is the initial segment ordering, and for each $x$, the size $p(x, x)$ of $x$ is $2^{-\text{length}(x)}$. As $p|S^\omega$ is precisely the Baire metric of classical descriptive set theory [6] we term $p$ the Baire partial metric.

The partial metric framework for modelling completeness is chosen because we can use the following fixed point result, actually a generalisation of Banach's contraction mapping theorem [9], to give us a completeness induction theorem. With this result we can go forward in the next section to formulate a test for lazy data flow deadlock, and in fact, for any pmetric.
Theorem 4.1 (The partial metric contraction mapping theorem, Matthews [8]). For each complete pmetric\(^2\) \(p: U^2 \rightarrow \mathbb{R}\), and for each function \(f: U \rightarrow U\) such that
\[
30 \leq c < 1, \quad \forall x, y \in U, \quad p(f(x), f(y)) \leq c \times p(x, y)
\]
called a contraction, firstly there exists a unique \(a \in U\) such that \(a = f(a)\), and secondly \(p(a, a) = 0\).

5. The cycle contraction mapping theorem

In this section we prove the main technical result of this paper. The cycle contraction mapping theorem generalises, to the world of partial metric spaces, both the cycle sum test over the Kahn domain and Banach's contraction mapping theorem over complete metric spaces [9]. Banach introduced the notion of contraction as a tool for repeatedly reducing the distance between two points in such a way that we may converge upon a unique fixed point. Now we consider not points but \(n\)-tuples of points, and so we need a generalized notion of contraction \(f: U^n \rightarrow U^n\). For this we are inspired by the cycle sum test to consider functions of the form \(c: n^2 \rightarrow \mathbb{R}\).

Definition 5.1. For all \(i < j\) in \(\omega\) a path from \(i\) to \(j\) is a function \(\rho: \{i, i + 1, \ldots, j\} \rightarrow n\). For each path \(\rho\) from \(i\) to \(j\) let \(\# \rho\) denote \(j - i\). A path \(\rho\) from \(i\) to \(j\) is a cycle if \(\rho_i = \rho_j\). A subpath is a restriction of a path which is itself a path. A path is cycle free if it has no subpath which is a cycle. Paths \(\rho\) from \(i\) to \(j\) and \(\rho'\) from \(i'\) to \(j'\) are disjoint if \(j \leq i'\) or \(j' \leq i\). The product \(*c\) of a path \(\rho\) from \(i\) to \(j\) in a function \(c: n^2 \rightarrow \mathbb{R}\) is
\[
c(\rho_0, \rho_1, \rho_2, \ldots, \rho_{j-1}) = c(\rho_0, \rho_1) \times c(\rho_2, \rho_3) \times \cdots \times c(\rho_{j-1}, \rho_j).
\]
A cycle contraction constant is a function \(c: n^2 \rightarrow \mathbb{R}\) such that the product of every cycle in \(c\) is less than 1.

The cycle sum test can be formulated using a cycle contraction constant over the Baire partial metric space. Suppose \(M: n \rightarrow \{\cdots, -1, 0, 1, \ldots, \infty\}\) is one of Wadge's functions for a Kahn network as described earlier. Let \(c: n^2 \rightarrow \mathbb{R}\) be such that
\[
\forall i, j \in \omega, \quad c(i, j) = 2^{-M(i, j)}.
\]
Then the network passes the cycle sum test if and only if \(c\) is a cycle contraction constant.

Lemma 5.1. For each cycle free path \(\rho\) from \(i\) to \(j\) in \(c: n^2 \rightarrow \mathbb{R}\), \(\# \rho < n\).

\(^2\)A complete pmetric is one in which every Cachy sequence converges. All the pmetrics considered in this paper are complete. For more details on complete pmetrics see [8].
Proof. The cardinality of \( \{\rho_i, \ldots, \rho_j\} \) is \( \#\rho + 1 \) as \( \rho \) is cycle free. But \( \{\rho_i, \ldots, \rho_j\} \subseteq n \), then \( \#\rho + 1 \leq n \). □

Thus for each cycle contraction constant \( c \) we can define \( \uparrow c \) to be the maximum product of a cycle free path.

Lemma 5.2. Each path \( \rho \) from \( i \) to \( j \) in \( c : n^2 \to \mathbb{R} \) has \( \lfloor \#\rho/n \rfloor \) disjoint cycles.

Proof. \( \rho \) has the disjoint subpaths \( \rho \{i, \ldots, i+n\}, \rho \{i+n, \ldots, i+2n\}, \ldots, \rho \{i+(\lfloor \#\rho/n \rfloor - 1) \times n, \ldots, i+\lfloor \#\rho/n \rfloor \times n\} \).

Thus by Lemma 5.1 each of these subpaths has at least one cycle. □

Lemma 5.3. The supremum \( \downarrow c \) of the set of all cycle products of a cycle contraction constant \( c \) is less than 1.

Proof. For each cycle \( \rho \) in \( c \) we can, by Lemma 5.1, repeatedly remove subcycles to get a subcycle \( \rho' \) such that \( \#\rho' < n \) and (as \( c \) is a cycle contraction constant) \( *\rho \leq *\rho' < 1 \).

There are however only a finite number of cycles \( \rho' \) in \( c \) such that \( \#\rho' < n \), and so \( \downarrow c < 1 \). □

Lemma 5.4. For each cycle contraction constant \( c : n^2 \to \mathbb{R} \) we can choose \( \bullet c \in \omega \) such that for each path \( \rho \), \( \#\rho = \bullet c \Rightarrow *\rho \leq 1/2n \).

Proof. For each path \( \rho \) we can (by Lemma 5.1 and 5.2) remove \( k \geq \lfloor \#\rho/n \rfloor \) disjoint subcycles to leave a cycle free sub path. Thus, \( *\rho \leq \uparrow c \times \downarrow c^k \). Thus as by Lemma 5.3 \( \downarrow c < 1 \), by choosing large enough \( \#\rho \) we can make \( *\rho \) as small as we like. □

We can define our notion of contraction suitable for functions in \( U^n \to U^n \) over a pmetric \( p : U^2 \to \mathbb{R} \).

Definition 5.2. \( f : U^n \to U^n \) is a cycle contraction if there exists a cycle contraction constant \( c : n^2 \to \mathbb{R} \) such that

\[
\forall X, Y \in U^n, \forall i, j \in n, \quad p(f(X)_i, f(Y)_j) \leq c(i,j) \times p(X_j, Y_j).
\]

Theorem 5.1 (The cycle contraction mapping theorem). Each cycle contraction \( f : U^n \to U^n \) over a complete partial metric \( p : U^2 \to \mathbb{R} \) has a unique fixed point, and this point is complete.

Proof. It can be easily shown that if a function composed with itself many times has a unique fixed point then that point must also be the unique fixed point of that function. We thus look at \( f^{**} \), \( f \) composed with itself \( \bullet c \) times.
By Definition 5.2, for each $\rho$ from $i$ to $i$ in $c$ such that $\# \rho = \bullet c$ times.

$$p(f^*(x)_{i}, f^*(y)_{i}) \leq \rho \times p(x_i, y_i) \leq p(x_i, y_i)/2n.$$  

Thus,

$$p^n(f^*(x), f^*(y)) \leq p^n(x, y)/2.$$  

Thus by Theorem 4.1 $f^*$ has a unique fixed point, and this point is complete.  

6. Lazy data flow deadlock

Armed now with the cycle contraction mapping theorem we can return to the subject of this paper, lazy data flow deadlock. First we need to understand what it means for an arbitrary data flow network to either deadlock or livelock.

Definition 6.1. A data flow network over a $\rho$-metric history domain (i.e. a history domain defined using a $\rho$-metric) locks if its computed value is not complete.  

The extensional concept of locking in a network captures precisely the operational idea that for some reason a network may stop in its computation before reaching a successful conclusion. The cause may be deadlock, livelock, or something else. At the high level of extensionality we can only discuss the extent to which a network's behaviour has stopped. The causes must be examined using the network's lower level operational semantics. Consequently extensionality cannot distinguish between deadlock and livelock whose effect upon the computation's completion is identical. If the intention of extensionality were to rigorously discuss operational semantics then we would certainly have failed as two very different behaviours cannot be distinguished using our methods. However, extensionality is intended to work alongside operational semantics in a spirit of pragmatism by discussing abstractions of its ideas. Unlike others [3, 7] we do not seek to replace operational semantics with an equally complicated formalism.

Wadge suggested that with a theory of completeness we would “... allow a fixed-point semantics for a large class of ‘obviously terminating’ recursive programs which would be mathematically ‘conventional’ in that it could completely avoid reference to partial objects and approximation”. We can respond positively to this suggestion by refining Theorem 5.1 to use only complete objects. To do this we must first make some sensible assumptions. Firstly, each domain should have enough complete objects, and distances should not be any larger than is necessary. It should be stressed that while the following sensible definitions appear appropriate for the Kahn and Lucid domains we have no guarantee that they are the right ones for other $\rho$-metrics.
Definition 6.2. A pmetric $p$ over $U$ is sensible if
\[ \forall x, y \in U, \exists x', y' \in U, x \preceq_p x' \land y \preceq_p y' \land p(x', x') = p(y', y') = 0 \land p(x, y) = p(x', y'). \]

Both the flat and Baire partial pmetrics are sensible, as are the finite and countable products of a sensible pmetric. Our second sensible assumption is a generalisation of monotonicity.

Definition 6.3. For each sensible pmetric $p$ over $U$, $f: U \to U$ is sensible if
\[ \forall x \preceq_p x', y \preceq_p y', p(x, y) = p(x', y') \iff p(f(x), f(y)) = p(f(x'), f(y')). \]

Theorem 6.1 (The complete cycle contraction mapping theorem). For each sensible complete pmetric $p: U^2 \to \mathbb{R}$ and sensible $f: U^n \to U^n$ such that $f|\{X \in U^n | p^n(X, X) = 0\}$ has a cycle contraction constant $c$, $f$ has a unique fixed point, and this point is complete.

Proof. By Theorem 5.1 it is sufficient to show that $c$ is a cycle contraction constant for $f$. Suppose $X, Y \in U^n$. Then as $p^n$ is sensible we can find $X', Y'$ as in Definition 6.2. Thus by Lemma 4.1 and as sensible functions are monotonic, for each $i \in \mathbb{N}$, $p(f(X)_i, f(Y)_i) = p(f(X')_i, f(Y')_i)$.

Thus,
\[ \forall i, j \in \mathbb{N}, p(f(X)_i, f(Y)_i) \leq c(i, j) \times p(X'_j, Y'_j) = c(i, j) \times p(X_j, Y_j). \]

The implication of this result for Lucid is that Wadge's cycle sum test has now been generalised into a computable test for proving some lazy programs deadlock free.

7. Towards a general notion of completeness

We have in this paper presented a formal theory of completeness using partial metrics. However, a general theory of completeness will require much more than the author has currently provided. In particular it is unclear how to measure the completeness of a function. This question must be answered if reflexive domain equations are to be solved and a pmetric model for the lambda calculus established. Similarly it is unclear how in general to define pmetrics such that as many as possible of the maximal objects are complete. This matter must be resolved if useful pmetrics are to be constructed for applications in program verification.

It is far from certain to the author that completeness is in general a theory of quantities as may have been suggested by our use of the pmetric distance function.
More likely completeness is in general a notion in what we may term partial mathematics. Conventional mathematics is founded upon the equivalence relation and Hausdorff separability, ideas which are far too strong for computer science. As Hovsepian convincingly argues [4] we have yet to fully appreciate this fact. In the face of parallelism the Scott–Strachey world of $T_0$ spaces appears to be running out of steam, and so point set topology itself may have to give way to a weaker framework based upon partial objects instead of separable points. Only then is completeness likely to find a general notion.

References