Vahedi, B., Kuhn, W., & Ballatore, A. (2016). Question-Based Spatial Computing - A Case Study. In T. Sarjakoski, M. Y. Santos, & L. T. Sarjakoski (Eds.), Lecture Notes in Geoinformation and Cartography (AGILE 2016) (pp. 37–50). Berlin: Springer.

Question-Based Spatial Computing - A Case Study

Behzad Vahedi, Werner Kuhn, and Andrea Ballatore

Abstract Geographic Information Systems (GIS) support spatial problem solving by large repositories of procedures, which are mainly operating on map layers. These procedures and their parameters are often not easy to understand and use, especially not for domain experts without extensive GIS training. This hinders a wider adoption of mapping and spatial analysis across disciplines. Building on the idea of core concepts of spatial information, and further developing the language for spatial computing based on them, we introduce an alternative approach to spatial analysis, based on the idea that users should be able to ask questions about the environment, rather than finding and executing procedures on map layers. We define such questions in terms of the core concepts of spatial information, and use data abstraction instead of procedural abstraction to structure command spaces for application programmers (and ultimately for end users). We sketch an implementation in Python that enables application programmers to dispatch computations to existing GIS capabilities. The gains in usability and conceptual clarity are illustrated through a case study from economics, comparing a traditional procedural solution with our declarative approach. The case study shows a reduction of computational steps by around 45%, as well as smaller and better organized command spaces.

Key words: Spatial computing, core concepts, question-based analysis, abstract data types

Behzad Vahedi (✉)
Center for Spatial Studies, Department of Geography, University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA,
e-mail: behzad@geog.ucsb.edu

Werner Kuhn
Center for Spatial Studies, Department of Geography, University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA, e-mail: werner@ucsb.edu

Andrea Ballatore
Center for Spatial Studies, University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA, e-mail: aballatore@spatial.ucsb.edu

1 Introduction

Geographic Information Systems (GIS), whether proprietary or open source, have evolved into large and complex toolboxes that require expert technical knowledge to be applied to real-world spatial problems. This evolution toward greater complexity happened primarily because of the growing demand for spatial computing in business and government operations. The demand for more functionality has mostly been addressed through "creeping featurism", i.e., incremental extensions to existing tools rather than a comprehensive approach to designing the next generation of spatial computing systems (Tomlinson, 2007).

As a result, scientists and decision makers can often only benefit from GIS after prolonged training. Moreover, domain competence tends to get separated from technological prowess, producing disjointed and potentially sub-optimal solutions to problems. It is then often impossible or too time consuming to iterate solution procedures or explore alternatives.

GIS users and application programmers who wish to ask *questions* using spatial data are forced to write *procedures* to generate and manipulate maps that, hopefully, will answer the questions. Domain questions are often lost in translation during the formulation and execution of these procedures. We believe that an approach based on the idea of asking spatial questions will create new and greater opportunities for programmers to develop applications and eventually allow users to interact with spatial computing systems in a way that is more interrogative and, thus, more natural and productive. Clearly, the challenge to address is to define the vocabulary in terms of which such questions can be asked and answered.

This paper proposes a novel approach to address this challenge and, more generally, the fragmented landscape of spatial computing. To do so, it utilizes the core concepts of spatial information presented earlier (Kuhn, 2012; Kuhn and Ballatore, 2015) and Abstract Data Types (ADTs) built on them to propose a new style of computational solution to spatial problems, in terms of spatial *questions* to be asked rather than *procedures* to be executed. The approach is illustrated by a case study from economics, in which a domain expert evaluates economic activity in China by assessing nocturnal luminosity around roads. All source code is available online.[1]

The remainder of this paper is organized as follows. Section 2 explains the idea of question-based computing and the differences to procedural approaches. Section 3 presents a summary of related work and is followed by Section 4 on the notion of core concepts of spatial information and the pertinent computations. The case study is presented and its Python-based implementation is shown in Section 5, before concluding with a discussion of results and ongoing research (Section 6).

[1] `https://github.com/spatial-ucsb/ConceptsOfSpatialInformation`

2 Question-based spatial computing

We posit that users want to find *answers to spatial questions* when they use GIS. These questions might stem from different domains, ranging from economics to ecology and public health, but they usually reflect some form of spatial thinking and reasoning used by humans (Gao and Goodchild, 2013). Yet, GIS are not designed in a way that would allow users to find and understand answers without going through procedures that are often complicated and difficult to assemble. GIS users do not really want to assemble chains of procedures, but have to learn to do so, often with difficulty, because there is no alternative. Application developers, in turn, find it hard to develop question-answering tools, because the computations available to them are not organized around contents and questions, but around layers, data formats, and historically grown commands and parameters.

A system organized around questions rather than procedures could be beneficial for end users since it would be closer to their needs and thus easier to understand. Users of such a system would not need to worry about the procedures and their sometimes numerous input parameters, often resulting from implementation details. The ideal situation could be compared to the realm of relational databases where Edgar F. Codd in the 1970s defined an algebra with a well-founded semantics for asking questions about the data stored in relational databases (Codd, 1970). He defined tables as the "core concept" of data in databases and then defined five "core computations" on tables, namely selection, projection, Cartesian product, set union, and set difference (Codd, 1970). These operations form the Relational Algebra that is the foundation for the Structured Query Language (SQL). SQL allows users to access data by SELECTing attributes FROM tables WHERE some conditions hold. This is a purely declarative approach, focusing on questions and answers rather than (at the user level) on procedures to operate on data. We are seeking a similar approach for GIS, enabling users to ask questions without concerning themselves about particular spatial data structures and their procedures.

To achieve this longer term goal, we propose an abstraction on GIS information contents, borrowing the concept of Abstract Data Types (ADT) from computer science (Liskov and Zilles, 1974) and applying it to the user level. An ADT has been defined as a "class of objects whose logical behavior is defined by a set of values and a set of operations" (Dale and Walker, 1996). ADT specifications capture a wide range of implementations, but users of the ADT only need to understand the specification, rather than the implementation, to know how instances of the type behave (Guttag and Horning, 1978). Relational algebra can be seen as a user-level ADT for databases. Applying this idea to GIS, we use the core concepts of spatial information as classes of objects and define core computations as operations capturing their behavior. These classes and operations can then be used to ask and answer spatial questions; each concept has a set of specific computations, answering questions and defining its behavior.

3 Related work

To the best of our knowledge, the realization of a question-based approach to spatial computing with user-level abstract data types has no direct predecessors or competitors, although there are a number of related initiatives and technologies that deserve discussion.

The communities of the Semantic Web and of Applied Ontology[2] have produced innovative ways to express and query spatial information, mainly based on the paradigm of Linked Data (Kuhn et al., 2014). Analytical capabilities, however, are not normally available with these approaches. Geospatial ontologies specify key application domains and data, e.g., land cover or hydrology (Ahlqvist, 2005; Ames et al.), but without the level of generalization sought here. The deeper semantic questions, concerning what contents users want to talk about and manipulate at a GIS user interface, have hardly been addressed so far in the form of implemented approaches. More generally, there is still no commonly accepted classification of the types of spatial (or even just geographic) information–contrasting with the vast literature on data types and structures to store that information.

Standardization has mainly focused on service-based data exchange, such as through the Geography Markup Language GML,[3] to some extent abandoning the original aspiration of defining spatial computing services independently of encoding formats (Kottman, 2001). The idea of a software-independent essential model of geospatial computing (Cook and Daniels, 1994) was not pursued, and the language of geographic information standards became one of software technologies and data transfers, rather than one of spatial information contents, questions, and computations. While this may have been a good choice for existing GIS technology vendors and user communities, it resulted in voluminous standards with heavy doses of jargon and acronyms, severely limiting outreach to and adoption by new communities.

One reason for this situation is the lack of theory on how to structure spatial computations in order to serve actual application needs. Attempts at devising a theoretical basis for GIS operations (Tomlin, 1990b; Albrecht, 1998) produced useful groupings of GIS functions, but have not attained a theoretical power and simplicity comparable to Codd's relational algebra for databases. Thus, the obligation to improve the situation is primarily that of researchers, not of vendors and standards bodies.

Relational algebra itself, although efficient for tabular data, is not sufficient for spatial data, since the nature and structure of spatial data is different from that of tables. Spatial databases have been an essential component in GIS and related information systems for two decades now, providing efficient data structures to index and query spatial data. However, from the user's perspective, systems such as PostGIS,[4]

[2] `http://semantic-web-journal.org` – All URLs cited in this article were accessed on December 4, 2015.

[3] `http://www.opengeospatial.org/standards/gml`

[4] `http://postgis.net`

Oracle Spatial,[5] and Spatial Hadoop[6] offer only spatial extensions of SQL, forcing their users to formulate their questions in the language of tables, which does not sufficiently capture the spatial nature of the data. Specialized graph and array databases, such as SciDB,[7] can be used for spatial computing, but focus on one particular view of data, without providing the necessary additional abstractions for asking spatial questions independently of data structures. The same comment applies to the spatial extensions of the statistical package R (Lovelace and Cheshire, 2014). Finally, spatial computing libraries such as GDAL[8] provide specific raster and vector views of contents. While they are powerful for complex data manipulations and format conversions, these tools tend to lock users into particular representational choices and burden them with implementation details.

By contrast, our mission is to enable question-based spatial computing through information content and quality abstractions. These abstractions are provided by the core concepts of spatial information and the computations on them, as summarized in the next section.

4 Core concepts and computations for spatial information

Core concepts of spatial information have been defined as concepts to interpret spatial data and to bridge the gap between spatial thinking and spatial computing (Kuhn and Ballatore, 2015). They initially started as ten concepts and were later reduced to the following seven:

1. **Location** – The idea of locating something relative to something else, applicable to instances of the following four content concepts.
2. **Field** – A property with a value for each position in space and time.
3. **Object** – An individual that has properties and relations with other objects.
4. **Network** – A set of objects (nodes) linked by a binary relation (edges).
5. **Event** – Something that happens and involves fields, objects, and/or networks as participants.
6. **Granularity** – The level of detail in some spatial information.
7. **Accuracy** – The correspondence of some spatial information with what is considered a true state of affairs.

Location is a base concept, applicable to the next four concepts, which in turn can be seen as "content concepts." Granularity and accuracy are "quality concepts" that can be applied to the base and content concepts, in order to set and assess the quality of spatial information (see Kuhn, 2012). The core concepts provide a conceptual foundation for spatial computing, which is still evolving. However, merely translating

[5] http://www.oracle.com/database/big-data-spatial-and-graph

[6] http://spatialhadoop.cs.umn.edu

[7] http://www.paradigm4.com

[8] http://www.gdal.org

GIS commands and data structures into core concept terms would not yield useful results. Rather, GIS applications should be reorganized around questions posed in terms of core concepts.

For this purpose, we define a set of core computations for each core concept. These core computations should form a minimal yet powerful and complete set of operations that are applicable across different application domains. Defining a large, open set of operations would go against the central philosophy of the core concepts and would be similar to existing, bloated GIS command sets. Core computations allow for extension by combining with computations of other core concepts and they get implemented through existing GIS functions; this allows for a small set of computations. The definition of core computations is ongoing research, and the initial set formalized in Kuhn and Ballatore (2015) has changed. A new field operations, called *restrict domain* and a new granularity operation, called *coarsen*, will be introduced in Section 5.2.

Defining core concepts and core computations in this way creates Abstract Data Types (ADTs) for GIS users. This kind of data abstraction will enable users to formulate spatial questions to be answered by existing GIS. The real world entities captured in a GIS application can be seen as instances of the different concepts, which then suggests computations to be applied in GIS projects.

Spatial questions may be answered directly with one or a set of core computations, but in most cases they need to be decomposed into simpler questions, each of which could be answered by one or a set of operations. This process can be seen as a case of *problem decomposition*, which in existing GIS platforms has to be performed in terms of data structure manipulations. By contrast, with the core concepts, users can ask questions and find answers by using a smaller set of computations organized around spatial concepts, not their representations. This difference is the essence of our approach to question-based computing.

The goal of our work is not to reinvent or replace existing GIS or any other spatial computing platforms, but to make the computations easier to discover, understand, and use. From an object-oriented implementation perspective, core concepts and core computations can act as wrappers for existing GIS and library functions. To achieve this goal, we specify and test the core computations in a Python implementation, articulating our approach through a real-world case study.

5 A case study from economics

MIT economist Matt Lowe provides a striking recent example of the difficulties and frustrations that a domain scientist encounters when using GIS (Lowe, 2014). Lowe uses nocturnal luminosity observed by satellites as a proxy measure of development and welfare, noting that light density has a fairly strong correlation with local economic activity.

He proposes computations on four pieces of spatial information: (1) satellite imagery that captures global nocturnal luminosity, (2) a global map of gas flares, (3) the boundaries of all the countries in the world, and (4) roads in China. Gas flares

are of industrial origin and outside of Lowe's economic scope of interest, so they are intentionally excluded from the luminosity analysis. As for the data layers, the first is imported as raster data and the following three are imported as vector data.

Ultimately, Lowe wants to answer the spatial question: *What was the level of economic activity near roads in China in 1994, based on nighttime luminosity as a proxy?* In his description of the ArcGIS procedure he has used, and based on the ArcPy scripting library, Lowe repeatedly puzzles over the amount of complex details involved in executing this seemingly simple spatial analysis (Lowe, 2014). In order to address the reasons for this complexity and then a remedy, we first present Lowe's procedural approach and then our alternative approach based on core concepts driven by spatial questions.

5.1 Solution with procedural GIS

The procedural steps that Lowe uses to answer his spatial question are as follows:

1. Erase gas flares from the data of countries
2. Calculate the average luminosity
3. Clip the luminosity data (output of 2) to the output of step 1
4. Extract the data in China from the output of step 3
5. Create a buffer around the China roads
6. Clip the output of step 4 to the extent of buffered roads
7. Clip the luminosity data to the extent of the output of step 6
8. Create a grid
9. Apply a zonal mean operator to each grid cell to calculate mean luminosity

This recipe of nine steps is a typical example of the procedural approach currently necessitated by spatial computing: in order to obtain an answer to a question, users have to apply a sequence of steps on specific data formats, which are not trivial to locate in the complex toolboxes of modern GIS.

An examination of Lowe's analysis illustrates how users without substantial expertise have difficultly finding, understanding, and correctly applying GIS procedures. Some of Lowe's analysis choices reveal confusions or inefficiencies that a GIS user might face. For instance, to determine the luminosity inside China, excluding the gas flares, he first erases the gas flares from the global map of countries, then clips the luminosity data of the world to the output of the previous process, and again clips the resulting data set to the extent of China. Whereas first clipping the global map of countries to the extent of China and then clipping the luminosity data to the extent of China would have been a more expedient technique.

The final results of Lowe's procedure is shown in Figure 1. Dark areas have no luminosity, while light areas have high luminosity. Cross-hatched areas are outside of the road buffer and have been excluded from analysis.

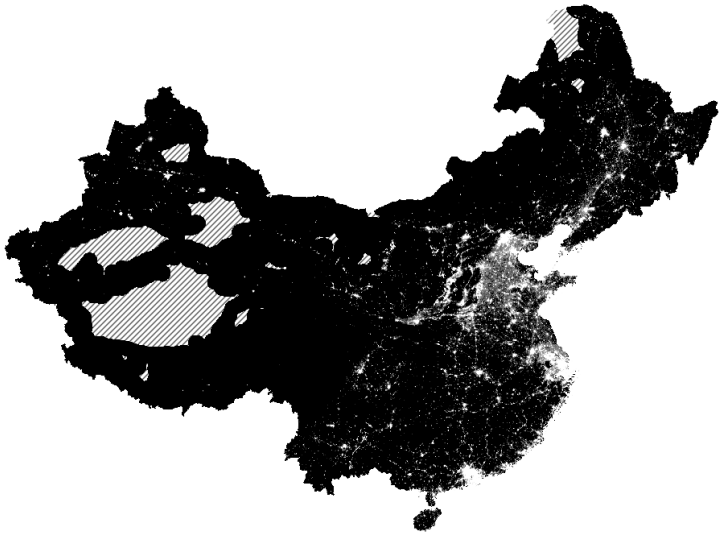

Fig. 1 The final result of Lowe's ArcPy procedure

5.2 Solution based on spatial questions

In order to demonstrate the potential of core concepts for simplifying spatial computing, we propose an alternative approach to answer Lowe's question. The main question is decomposed into three smaller questions that can be answered with one or two computations each. This captures the idea of question-based computing, in which computations are designed to answer specific spatial questions and are therefore more "declarative" than procedural. Figure 2 shows the ArcPy scripts developed by Lowe on the left with the alternative solution based on core concepts, also written in Python, on the right.

We utilize Python for several reasons. Python is popular among application programmers and users, has the capacity to implement ADTs, and is interoperable with existing GIS tools and libraries. For the back-end implementation of our approach, we work with ArcPy to keep consistency with the original solution and to compare on a fair and relevant basis. In fact, the computations used here, act as wrapper for ArcPy functions.

Before starting the actual computations, the preliminary step is always to provide the required input data. This process is conventionally done by loading data layers as raster or vector data into the chosen GIS. This perspective tends to lock users into conceptualizations associated with the data formats, limiting the set of available computations to those defined for each data type and contributing to unnecessary format conversions. For instance, in order to clip the grid to the extent of the luminosity data in China, Lowe had to convert the luminosity raster to vector

ArcPy

```
# Get China less gas flares polygon
arcpy.Select_analysis("countries_nogas", "china1.shp",
   "\"NAME\" = 'China'")
# Average two satellites for 1994
outRaster = (Float("F101994")+Float("F121994"))/2
outRaster.save("FXX1994")

# Use buffer tool and roads to make polygon of China
# close to roads, then clip china1 to this
arcpy.Buffer_analysis("a2010_final_proj", "roadbuff.shp", "0.5
   DecimalDegrees", "FULL", "ROUND", "ALL", "")
arcpy.Clip_analysis("H:/Research/Data/Lights/china1.shp",
   "H:/Research/Data/Lights/roadbuff.shp", "china2.shp", "")

# Clip each lights raster to extent of china2
rasterList = arcpy.ListRasters("F*")
for raster in rasterList:
   arcpy.Clip_management(raster, "-179.9999 -90.0 180.0
     83.62741", "G"+str(raster[1:]),
     "H:/Research/Data/Lights/china2.shp", "",
     "ClippingGeometry")

# Create grid to extent of one of new light rasters
arcpy.CreateFishnet_management("ch_grid.shp", "73.55416
   18.15416", "73.5541 28.15416","0.1", "0.1", "0", "0",
   "134.77916 53.5625", "NO_LABELS", "G101992", "POLYGON")
arcpy.RasterToPolygon_conversion("G101992", "G101992p.shp",
   "NO_SIMPLIFY", "Value")

# Process: Clip grid to perimeter of polygon
arcpy.Clip_analysis("H:/Research/Data/Lights/ch_grid.shp",
   "H:/Research/Data/Lights/G101992p.shp", "china_grid.shp",
   "")

# Zonal statistics on each year
rasterList = arcpy.ListRasters("G*")
for raster in rasterList:
   arcpy.gp.ZonalStatisticsAsTable_sa("H:/Research/Data/Lights/
     china_grid.shp", "FID", raster,
     "l"+str(raster[5:])+".dbf", "DATA", "MEAN")
```

Core Concepts

```
# load input data
china_boundary = MakeObject('data/china.shp')
china_lights_1 = MakeField('data/lights_1994a.tif')
      .restrict_domain(china_boundary, 'inside')
china_lights_2 = MakeField('data/lights_1994b.tif')
      .restrict_domain(china_boundary, 'inside')
gas_flares = MakeObject('data/china_flares.shp')
roads = MakeObject('data/china_roads.shp')


# What is the luminosity in year 1994 in China, excluding
gas flares?
average_luminosity = china_lights_1
                   .local(china_lights_2, 'average')
luminosity_noGas = average_luminosity
                   .restrict_domain(gas_flares, 'outside')

# What is the luminosity within 0.5 degrees from roads?
 roads_buffered = buffer(roads, 0.5, 'Decimal Degrees')
 luminosity_around_roads = luminosity_noGas
                   .restrict_domain(roads_buffered,'inside')

# What is the mean luminosity in a 0.1 by 0.1 degree area?
final_results = luminosity_around_roads.coarsen(0.1,0.1))
```

Fig. 2 Comparison between the procedural and core concepts solutions to the China luminosity study (Lowe, 2014).

data. The decision to convert from raster to vector is encouraged by the chosen GIS framework and not inherent (in fact rather contrary) to the question.

Alternatively, core concepts allow users to decide which concept is most appropriate for framing the problem and interpreting the data sets, given the questions to be answered. For instance, a user could treat the Chinese road data as a network or instead as set of road objects. The only computation that needs to be performed on the road data involves determining which areas lie within a certain distance from roads. Therefore, the most appropriate (simplest) choice is to treat the roads as a set of objects to which a buffer operation can be applied. Thus, the road data are interpreted and subsequently loaded as a set of objects.

Similarly, China and the gas flares are each interpreted as objects and the two raster files containing the luminosity data for 1994 as fields. The following Python code is used to load these data. In Lowe's solution, these operations were performed in a pre-processing stage.

```
china_boundary = makeObject('data/china.shp')
china_lights_1 = makeField('data/lights_1994a.tif')
                .restrict_domain(china_boundary, 'inside')
china_lights_2 = makeField('data/lights_1994b.tif')
                .restrict_domain(china_boundary, 'inside')
gas_flares = makeObject('data/china_flares.shp')
roads = makeObject('data/china_roads.shp')
```

A considerable number of steps applied by Lowe (four out of nine) are spent on limiting the spatial coverage (extent) of his data. He uses "select by attribute" in the fourth step and "clipping" in the third, sixth, and seventh steps. Through the core concepts, limiting coverage is a much easier task achieved by restricting the domain of a field. By definition, each field has a domain for which it is defined (Kuhn and Ballatore, 2015) and when creating a new instance of a field, its domain is inferred from the extent of the loaded data. Since clipping in fact only reduces the domain of a field, it is conceptually easier to restrict the domain by the new boundary, avoiding the need to modify or create any values. Our `restrict_domain` computation restricts the domain of a field in this way. In fact, there is not even a need for any computations in such a domain restriction, as one can just add or subtract spatial extents to the domain definition. In the second and the third commands of the above code, the domain of the fields (that are being created from a tif file) are restricted to the inside of the boundary of China, at the time of creation.

Once the data have been interpreted and loaded as core concept instances, we can start asking spatial questions. Lowe's overall question is naturally subdivided into three simpler questions:

- What is the average luminosity in 1994 in China, excluding gas flares?
- What is the luminosity within 0.5 degrees from roads?
- What is the mean luminosity at a coarser (0.1 by 0.1 degree) granularity?

In order to answer the first question, the luminosity in the year 1994 is averaged from two satellite data sources for that year. Then the domain of this field gets restricted to the region outside of the gas flares. As the luminosity data have been interpreted as fields, a local field operation can be used for averaging the values. Local operations are well-known from map algebra (Tomlin, 1990a). To restrict the domain of the resulting field to the outside of gas flares, the `restrict_domain` function is applied.

```
average_luminosity = china_lights_1
                        .local(china_lights_2, 'average')
luminosity_noGas = average_luminosity
                        .restrict_domain(gas_flares, 'outside')
```

Answering the second question requires determining the luminosity near roads (within a 0.5 degree buffer). Buffering is a core computation on objects, requiring objects and the buffer distance as input. After applying it, the domain of the luminosity field can be reduced to the buffered region.

```
roads_buffered = buffer(roads, 0.5, 'Decimal Degrees')
luminosity_around_roads = luminosity_noGas
                        .restrict_domain(roads_buffered, 'inside')
```

To address the third question, the luminosity field can simply be coarsened. Lowe defined a vector grid to aggregate the raster into larger cells. This approach is computationally expensive and necessitates a raster to vector conversion on the luminosity data. Our alternative solution uses the concept of granularity, as a quality concept applicable to all content concepts, including fields. Two core computations

for granularity, *coarsen* and *refine*, allow for decreasing and increasing, respectively, the granularity of fields, objects, networks, and events (here shown for the case of fields):

```
final_results = luminosity_around_roads.coarsen(0.1, 0.1)
```

To do the same process in ArcPy, Lowe had to undertake a multi-step process: convert the luminosity raster to a polygon layer, clip the luminosity polygons to the extent of the grid, and then apply zonal statistics.

5.3 Implementation

To implement the core computations, we have constructed an hierarchy of abstract and concrete classes that leverages the classes defined by Kuhn and Ballatore (2015) and existing spatial computation platforms (currently only ArcPy). Abstract class constructors generate subclass instances to appropriately type and handle the loaded data. This removes common but unnecessary choices and parameters from user commands. The CcField abstract class, for example, points to the GeoTiffField subclass (hidden from the user), an instance of which is generated when a user loads ".tiff" data as a field. Within the same GeoTiffField subclass, functions for retrieving the domain and for reading in raster data allow `local` to execute and optimize the map algebra local operation in a number of ways (see section 5.2).

5.4 Comparison and discussion

Our core concept solution is not replicating the step-wise procedure that Lowe had to undertake, which was chaining together tools operating on data formats. It is also an improvement over a standalone use of ArcPy, which restricts users to fragmented step-wise operations on raster and vector data. As shown, the procedure used by Lowe contained nine analytical steps, some consisting of multiple operations. In contrast, our method based on core concepts has only three analytical steps, with five operations in total, thus being 45% shorter overall. A further improvement is that our computations should be simpler to understand, as they contain fewer parameters, the user does not have to deal with format conversions, and the steps follow a logical order established by the three sub-questions. We also expect overall command spaces to end up smaller and more usefully organized by the core concepts, though the current state of our work cannot yet demonstrate this effect.

6 Conclusions

This paper proposed a new approach to spatial computing based on asking questions rather than performing data manipulation procedures. Inspired by the notion of Abstract Data Types (ADT) from computer science, and leveraging the idea of the core concepts of spatial information and the language designed based on them, our approach allows for answering spatial questions by using core computations defined for each core concept.

In a first attempt of putting this approach into practice, we used a case study from economics, in which a domain expert had used ArcGIS (through its scripting language, ArcPy), to determine aggregated nocturnal luminosity in China near roads. We answered this spatial question by interpreting the input data in terms of core concepts and performing core computations on them. We see the case study as a realistic scenario to test the idea of question-based spatial computing and the underlying core concepts and computations for spatial information. Despite the encouraging results, this case study is only an initial step, and more real-world scenarios are needed to demonstrate the power and to identify the potential weaknesses of our approach.

Furthermore, a spatial question can either be answered directly by a core computation or the user decomposes it into smaller, answerable questions. Formulating a method to decompose spatial questions into smaller units will be pursued in future research, while further developing the core computations. We believe that this is an ambitious but achievable goal that will expand the range of spatial questions answerable through the core concepts and computations, without the need to grow their number.

This research also helped us extend and improve the formal specifications of core concepts and modify the computations defined on them. For instance, the granularity concept did not have a rigid formal specification beforehand, but now has a clearer definition and two core computations (refine and coarsen). The method proposed in this paper ends up being remarkably shorter than the published GIS solution and appears easier to understand. It uses just over half the number of computations used by Lowe.

Our ADTs were implemented in Python, currently using the ArcPy library, because of Python's interoperability with existing GIS tools and its growing usage within and outside GIScience. The core computations implemented for each concept are not yet complete, and defining new operations as well as modifying existing ones continues, based on additional case studies. Also, the method used in this paper has its limitations, as some of the operations were not fully compatible with ArcPy, requiring several workarounds. This problem will be addressed in future work leveraging multiple platforms beyond ArcPy, including commercial and open source GIS. Another line of future research will study transitions from one core concept to another, and their implementation as constructor functions.

We foresee our research to eventually lead to an open source Application Programming Interface (API) for spatial computing, based on core concepts and computations and dispatching to commercial or open source spatial computing platforms. This API will provide higher-level access to existing commands on these

platforms and thus support a broader community of users, ranging from GIS experts to domain users or application programmers without technical GIS training.

Acknowledgements We gratefully acknowledge the contributions of Thomas Hervey, Sara Lafia, Michael Wang, and others at the UCSB Center for Spatial Studies for helping shape and refine this idea and its implementation. We also acknowledge Professors Rich Wolski and Chandra Krintz from the Computer Science department at UCSB, who have been challenging us to apply the question-based approach to this kind of case study. We thank the anonymous reviewers for their insightful comments, which led to improvements in the paper.

References

Ahlqvist, O. (2005). Using uncertain conceptual spaces to translate between land cover categories. *International journal of geographical information science*, 19(7):831–857.

Albrecht, J. (1998). Universal analytical GIS operations: A task-oriented systematization of data structure-independent GIS functionality. In Onsrud, H. and Craglia, M., editors, *Geographic information research: Transatlantic perspectives*, pages 577–591. Taylor and Francis.

Ames, D. P., Horsburgh, J. S., Cao, Y., Kadlec, J., Whiteaker, T., and Valentine, D. Hydrodesktop: Web services-based software for hydrologic data discovery, download, visualization, and analysis. *Environmental Modelling & Software*, 37.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Cook, S. and Daniels, J. (1994). *Designing object systems*, volume 135. Prentice Hall, Englewood Cliffs.

Dale, N. and Walker, H. M. (1996). *Abstract data types: specifications, implementations, and applications*. Jones & Bartlett Learning.

Gao, S. and Goodchild, M. F. (2013). Asking spatial questions to identify gis functionality. In *Computing for Geospatial Research and Application (COM. Geo), 2013 Fourth International Conference on*, pages 106–110. IEEE.

Guttag, J. V. and Horning, J. J. (1978). The algebraic specification of abstract data types. *Acta informatica*, 10(1):27–52.

Kottman, C. (2001). White Paper on Trends in the Intersection of GIS and IT. *Open GIS Consortium*.

Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, 26(12):2267–2276.

Kuhn, W. and Ballatore, A. (2015). Designing a Language for Spatial Computing. In Bacao, F., Santos, M. Y., and Painho, M., editors, *AGILE 2015: Geographic Information Science as an Enabler of Smarter Cities and Communities*, pages 309–326. Springer, Berlin.

Kuhn, W., Kauppinen, T., and Janowicz, K. (2014). Linked data-A paradigm shift for geographic information science. In *Geographic Information Science*, pages 173–186. Springer.

Liskov, B. and Zilles, S. (1974). Programming with abstract data types. In *ACM Sigplan Notices*, volume 9, pages 50–59. ACM.

Lovelace, R. and Cheshire, J. (2014). Introduction to visualising spatial data in r.

Lowe, M. (2014). Night Lights and ArcGIS: A Brief Guide. [Online; accessed Nov-2015] `http://economics.mit.edu/files/8945`.

Tomlin, C. D. (1990a). *A map algebra*. Harvard Graduate School of Design.

Tomlin, D. C. (1990b). *Geographic information systems and cartographic modeling*. Prentice Hall, Englewood Cliffs, NJ.

Tomlinson, R. F. (2007). *Thinking about GIS: geographic information system planning for managers*. ESRI, Inc.