Designing a Language for Spatial Computing

"One of the main reasons why software projects fail is the lack of communication between the business users, who actually know the problem domain, and the developers who design and implement the software model." (Ghosh, 2011)

Abstract. We present the design rationale underlying a language for spatial computing and sketch a prototypical implementation in Python. The goal of this work is to provide a high-level language for spatial computing that is executable on existing commercial and open source spatial computing platforms, particularly Geographic Information Systems (GIS). The key idea of the approach is to target an abstraction level higher than that of GIS commands and data formats, yet meaningful within and across application domains. The paper describes the underlying theory of spatial information and shows its evolving formal specification. An embedding in Python exemplifies access to commonly available implementations of spatial computations.

Introduction

If spatial computing is to realize its often-proclaimed potential across disciplines, geographic information science needs to convey a clearer picture of what GIS and related technologies are good for. This picture must focus on information *contents* and user *questions*, rather than on data formats (e.g., raster and vector) and system commands, which dominate the current image of GIS. It should be a value proposition that is meaningful across application domains, while avoiding overgeneralizations (such as reducing all spatial information to a single form) or obscure terminology (such as abstract ontological terms).

Existing abstractions like the geo-atom (Goodchild, Yuan, & Cova, 2007) and the notion of generalized fields (Camara et al., 2014) are helpful in generating such a picture, but they attempt to squeeze all spatial information into a single form, avoiding content distinctions. Similarly, spatial analysis reduces spatial information essentially to products of random point processes. In the absence of a clear but nuanced picture of spatial information and computing, many potential users continue to believe that GIS is primarily used to make and store maps.

The problem of software requiring users to speak a language they may not be familiar with is as old as computing itself. In the early 1980s, computer scientists and psychologists started to describe and address it systematically. Don Norman coined the terms "Gulf of Execution" and "Gulf of Evaluation" (D. A. Norman, 1986) to describe the gap between how users think and how computers require them to talk (execution) as well as how they talk back to users (evaluation). By the early 1990s, usability engineering had become a much-researched part of system

Author copy
Kuhn, W., & Ballatore, A. (2015). Designing a Language for Spatial Computing. AGILE 2015, Lecture Notes in Geoinformation and Cartography. Bacao, F. et al. (Eds.) (pp. 309-326).

design and implementation, generally and in the GIS area (Egenhofer & Kuhn, 1999). The user interfaces of GIS rapidly changed from command line to direct manipulation interfaces, including visual programming languages (VPL).[1]

Yet, this re-organization of GIS languages has largely remained syntactic. The semantic questions, concerning what contents users want to talk *about* at a GIS interface, have hardly been addressed. Consequently, there is still no commonly accepted classification of the types of spatial or geographic information (as opposed to the data types) that are handled by GIS. Organizing GIS commands bottom-up turned out to be too hard (Albrecht, 1998), and ontologists have not yet come up with a top-down structure capturing what is special about spatial information. Geospatial ontologies, on the other hand, specify application domains like hydrology or land cover, without attempting the generalization sought here. The prolonged absence of an answer to what spatial or geographic information is about is further aggravated by *pride in complexity*: researchers and practitioners mastering GIS all too often believe that what they have spent so much effort (and money) on learning is inherently complex and needs to stay that way.

Standardization, meanwhile, has quietly abandoned its original aspiration of defining service interfaces for spatial computing at the content level. In the absence of theories to inform the design of such interfaces, OGC and ISO had to pursue a modern (service-oriented) form of data exchange, based on GML. The idea of a software-independent essential model of geospatial computing fell by the wayside, and the language of geographic information standards became one of software technologies, rather than of information contents. Sadly, this is mainly a result of researchers keeping themselves busy developing and testing industrial software specifications, rather than developing theories of spatial information and computing.

Thus, a quarter century into GIScience research and standardization, our field still fails to communicate what GI and GIS are about. Evidence for this can be found in the heterogeneous organization of textbooks and curricula. As a result, teaching GIS tends to default to software training, and attempts to explain the potential of GIS to, say, an epidemiologist, an economist, or a historian, tend to leave these colleagues with an impression that they better leave GIS to specialists.

Contrast this situation with that in statistics. The two fields play similar cross-disciplinary roles, but statistics has had time and intellectual aspiration to achieve a mature self-image and understanding of its tools. Simplifying somewhat (as one needs to, for this purpose), the field of statistics comes with a set of core concepts, ranging from random variables through distributions and correlations to statistical tests. If one wants to do statistics, these concepts provide access to and transfer between a large and growing variety of statistical computing tools, all of them more or less understanding and speaking these terms.

---

[1] Industrial products include *ModelBuilder* in ArcGIS and the *Workflow Designer* in Autodesk Map 3D.

This paper presents a novel approach to the problem of describing spatial information and computations above the level of GIS data formats and commands. It proposes to use the previously defined *core concepts of spatial information* (Kuhn, 2012) as a vocabulary for a language of spatial computing, supplying a high-level view of operations on these concepts. We first discuss what domain the language should be for (section 2), then recall the core concepts (section 3) and specify them through their operations (section 4), preparing for a sketch of an ongoing implementation in Python (section 5). Conclusions and a discussion of future work wrap up the paper (section 6).

What Domain?

While it remains to be seen whether "geographic" or "spatial" is the better scope for a language of spatial computing, our work rests on the assumption that most computational techniques developed for geographic applications are applicable to other spaces as well. For example, fields or networks are both useful structures across many scales, dimensions, and applications. Therefore, we refrain from any attempts to limit the application domains, and thus the scope of the domain-specific language, to geographic or other spaces. Still, the spaces considered here primarily are those of human experience in one to three dimensions. The applications are primarily geographic (dealing with neighborhoods or river catchments, for example) or indoor environments (dealing with floors, rooms or hallways), but other spaces remain in scope. We refer to this broad scope as our domain and to areas like ecology or economics as applications. Figure 1 provides an overview of the conceptual and technology layers involved in our proposal.

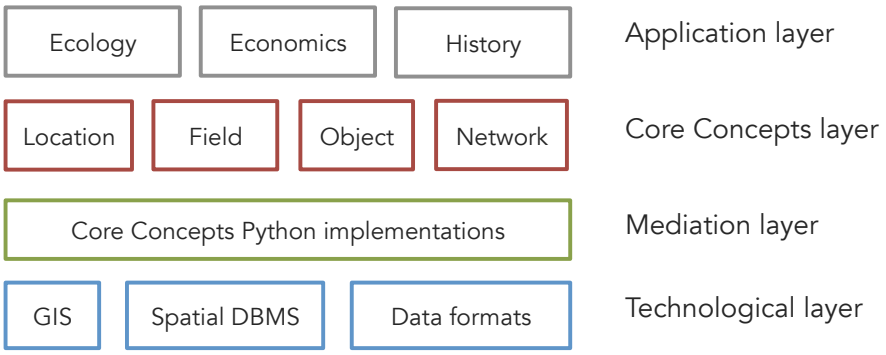

Figure 1: Overview of the spatial computing domain

Originally, we planned to map the core concepts to domain specializations (for example, translating the core concept of network to the domain specialization of

road network), but we found it beneficial to preserve the language's generality and abstraction even within applications. A domain mapping remains a valid option for any specialized domain language in which conceptualizations can be fixed. However, it is a characteristic of spatial computing that different conceptualizations co-exist, for example, of roads as networks or complex objects. Thus, we envisage a scenario where domain practitioners express their spatial questions in the proposed high-level language, specializing their entities when necessary, and operating at a cross-domain conceptual level.

Core Concepts of Spatial Information

This section summarizes the core concepts of spatial information in their latest form of seven concepts–see also (Kuhn, 2012). These connect spatial thinking to computing and provide a high-level vocabulary with which to ask and answer questions about phenomena in space and time. Spatial information is spatiotemporal by default in today's practice, so that the concepts do not distinguish spatiotemporal from spatial information.

A remark on the underlying notion of *information* is in order. Information answers questions. It exists only in the minds of people and not in computers, nor in the world outside of human minds or computers. Spatial information gets generated by humans observing the world with a reference to location and extent. Concepts of spatial information are the elements from which humans build mental representations of the world. Computers hold *data* about the world, recording human or technical observations and used to answer questions. It is often convenient to use one and the same term for these three aspects of information. For example, the term *object* can be used to refer to something physical in the world (say, a house), its mental representation (your idea of that house), and the data describing it (in a cadastral database).

The seven core concepts of spatial information comprise five concepts of information *content* and two concepts of information *quality*. The latter can be considered meta-information concepts, as they apply to all content concepts. Table 1 shows an overview.

<table>
<tr><th colspan="5">Core Content Concepts</th><th>Core Quality Concepts</th></tr>
<tr><td rowspan="2">Location</td><td rowspan="2">Field</td><td rowspan="2">Object</td><td rowspan="2">Network</td><td rowspan="2">Event</td><td>Granularity</td></tr>
<tr><td>Accuracy</td></tr>
</table>

Table 1: Overview of the core concepts of spatial information

This small set of core concepts of spatial information inevitably misses some ideas or relegates them to sub-concepts or non-core status. Other views of what is

"core" can be found in (Golledge, 1995) or (Janelle & Goodchild, 2011). Picking core concepts is mainly a terminological and granularity choice with which one decides on how to talk about spatial information. Their suitability needs to be judged through applications, for the purpose of which we are designing the domain language around them.

Location

The concept of location serves to ask and answer *where* questions. It is the most fundamental concept of spatial information (Golledge, 1995). Location is neither an attribute nor an object, but a *relation*. We locate something by relating it spatially to something else (Donnelly, 2005). Thus, there is no absolute location of anything, but there are infinitely many relative ones. When talking about *a* location, one means either a place or a position (see below).

Location information relates *figures* (being located) to *grounds* (locating the figures). What gets located plays the role of *figure*, what locates it plays the role of *ground* (Talmy, 1983). When we say "Santa Barbara is in California," Santa Barbara is chosen as the figure and California as the ground. Location information is always the result of a human judgment, assigning figure and ground roles to states of entities and relating them spatially. The roles of figure and ground can be assigned to states of objects (e.g., streets) or network elements (e.g., intersections).

Places are commonly used grounds. They often carry names, such as Santa Barbara, California, the Pacific Ocean, or North America. Like all grounds, they are entities in space *and* time, as the examples demonstrate. A place may start and cease to play its role as a ground—consider Yugoslavia, which was a place during a part of the 20th century. Place-based location information is normal for human communication and has become a key resource in spatial computing.

Figures get related to grounds through *spatial relations*. Locating Santa Barbara *in* California or *by* the Pacific Ocean uses prepositions to express the qualitative relations of containment and contact; locating it by geographic coordinates uses relations expressed as angular distances from the equator and a meridian.

Positions express spatial relations quantitatively, through distances from the grounds established by coordinate systems. For example, Wikipedia[2] says that Santa Barbara has the coordinates 34°25'33"N 119°42'51"W, stating angular distances from the equator and prime meridian of the WGS'84 coordinate system. In the vertical dimension, heights use an ellipsoid or geoid as ground and distances from it as positions. With the grounds conventionalized through coordinate systems, positions appear to be independent entities, but they remain relations, referring to default grounds.

[2] http://en.wikipedia.org/wiki/Santa_Barbara,_California

Field

Field information answers questions about the *value of an attribute* anywhere in a space of interest. The field concept is an elegant mathematical idea, originally developed in physics to explain gravity and widely adopted for spatial information models in geography, biomedicine, and other areas. It is, of course, the first of two fundamental views of spatial information, the second being the concept of an object (Couclelis, 1992; Galton, 2004). Fields capture phenomena that can be described by a property with a single value at any position in a space of interest. Geographic examples include temperature and wind fields.

Mathematically, every field is characterized by a *continuous function* from positions to values, meaning that small changes in positions map to small changes in values. The domain of the field function captures the space of interest. Positions can be spatio-temporal, although time often gets separated from space and modeled as snapshots. The positions as well as the values can be discrete, still allowing for continuous functions between them (Rosenfeld, 1986).

In practice, the continuity condition on the function is sometimes ignored, retaining only the functional relationship between positions and values. Land cover or land ownership are examples of phenomena requiring this broader definition, as their values do not change smoothly. Such a generalized "field" concept has been captured in the *geo-atom* of (Goodchild et al., 2007)and standardized in the *coverage* specification of the Open Geospatial Consortium as "digital spatial information representing space-time varying phenomena" (Baumann, 2010).

Object

Object information answers questions about properties and relations of objects. Objects provide the second fundamental way, after fields, of understanding spatial information. They capture individual things extended in space that can be identified and described by properties and relations. Typical geographic examples are buildings and lakes.

The only defining characteristic of all kinds of objects is that they have an *identity*. This allows for tracking their properties and relations over time. Changes can occur in all properties and relations, including spatial relations in the case of mobile objects.

Objects are *bounded*, i.e., they have finite sizes, although their boundaries are not always known or even knowable. Many natural objects do not have crisp boundaries and are better characterized by transition zones between what clearly belongs to them and what does not (Burrough & Frank, 1996). Consider mountains, forests, beaches, or ash clouds; while some positions clearly lie within these

objects, others clearly lie outside, and the transition zones are often vast. Boundaries, crisp or vague, may or may not be necessary for analysis.

The *properties* of objects are attributed to them as a whole and *relations* between objects hold for them as wholes. All properties and relations are either spatial or thematic; temporal aspects of objects (such as their creation) are captured by events in which objects participate. The values of an object's properties and the relations it participates in define its *state* at any time.

Parts of objects can themselves be treated as objects, and *complex objects* get aggregated from simpler ones. *Feature*s are parts of the surfaces of objects and can be considered special cases of objects. For example, while lakes may be seen as three-dimensional objects, they are also often treated as two-dimensional parts of the earth's surface.

The large variety of objects in any domain suggests a need to classify them. Object *types* are typically defined based on shared properties and relations. For example, water bodies may be classified into those with standing or flowing water, with lakes as a further subclass of standing water bodies.

Network

Networks hold information on connections between objects. They are used to answer a broad range of questions about connectivity, such as whether an object is reachable from another object, what the shortest path is between them, how central an object is in a network, where the sources and sinks are of something flowing through the network, or how fast something will spread and where it will spread to. Network applications benefit from modeling networks as graphs and from the vast choice of algorithms and implementations coming with this. As a consequence, the concept of networks is most broadly applied across disciplines.

The objects connected through a network are called its *nodes*. The variety of objects that can play the role of nodes is unlimited. They include physical objects (such as people or places), as well as mental and social ones (such as concepts, web pages, or companies).

Nodes get connected by any binary relation of interest, forming a network's *edges*. Every pair of nodes participating in the relation is connected by an edge. Edges may have physical realizations, as do roads, or they may be modeled as abstract connections only, as in the case of social relations. Edges, and with them whole networks, can be *directed*. For example, a road segment may be seen as a directed connection from one place to another. Applications often use a single attribute to characterize a relevant property of edges. A numeric attribute is called the *weight* or *impedance* of the edge, a nominal attribute its *color*. For example, travel distance (measured by length or time) can serve as weight, with longer connections having greater weights; colors can express different types of edges, such as travel modalities (walking, driving).

A *path* in a network is a sequence of nodes where each consecutive pair of nodes is connected by an edge. A node is reachable from another node, if there is a path connecting the two. A path returning to its origin is a cycle. Sometimes it is useful to think of a path as the sequence of edges connecting the nodes. A network is connected if it has a path between every pair of nodes.

A network can be *embedded* in a surface or a three-dimensional space, so that its nodes have positions and its edges shapes. Embeddings on the earth's surface or in a building are typical geographic examples. Many more network properties are defined in the literature (see, for example, (Newman, 2010)).

Event

Event information answers questions about what has happened, is happening, or may happen. For example, one might want to know where two people first met, how long it takes to get through a traffic jam, or whether it will rain tomorrow. While static maps show snapshots of unfolding events, computer models and their visualizations can now represent events and relationships between them. Events are rapidly becoming an important subject of spatial information, reflecting the fact that information about changes in our dynamic world is increasingly available, even in real time.

Events are individual portions of processes. For example, considering weather and traffic as processes, a rainstorm is an individual weather occurrence and a traffic jam is a single, limited occurrence of heavy traffic. All events are temporally bounded, i.e., of finite duration. Their beginnings and endings are not always known, just like for the spatial boundedness of objects.

Events have an *identity* and are described by temporal and thematic properties and relations. For example, a rainstorm may be named and characterized by its duration and the accumulated rainfall. Commonly used *temporal properties* of events are their duration or temporal bounds. The main temporal *relations* between events are precedence, co-occurrence, and posteriority. For example, a rainstorm can precede a traffic jam, co-occur with an electricity blackout, and follow (be posterior to) gusts of wind.

The key *relation* between events and the other core concepts is that of *participation*. Events involve fields, objects, and networks as participants, which often get changed through their participation. For example, rainstorms change temperature and humidity in their area, and traffic jams affect the choice of routes through road networks.

Parts of events can themselves be treated as events, and *complex events* can be aggregated from simpler ones. Part-whole relations are as central to event models as they are to object models. For example, a particular rainstorm may be part of a storm system sweeping a region and there may be questions about rainfalls or damages in individual storms as well as about their overall accumulation.

Granularity

Granularity information answers questions about the amount of detail in spatial information. For example, one may want to know how precisely a smart phone can locate itself, what cell size underlies a global climate model, what the smallest recorded buildings are on a map, whether gravel roads are included in a road network, or how long a snow storm must last to be recorded as a blizzard. Granularity characterizes all content concepts of spatial information.

Granularity is a concept of information *quality*, used to describe information itself rather than things in the world. Every piece of spatial information comes with some granularity, implicitly or explicitly, i.e., with a level of detail it captures. While the term granularity is sometimes used in a narrower sense to describe the level of detail in conceptualizations (but not in representations external to the mind), it is used here to mean the level of detail affecting any stage of producing or using spatial information—starting with conceptualization, and going through observation, representation, integration, and analysis, to visualization.

Commonly used alternative terms for granularity, in addition to level of detail, are *resolution* and *precision*. The term *scale* has so many different and often conflicting meanings that it is best used as an informal notion only, but it refers primarily to granularity. Other related terms, for instance *discrimination* or *precision* (in its second sense of repeatability of measurement), refer to measuring instruments, not to information, but they impose lower limits on granularity. Finally, the terms *spacing* and *support* have more specific technical meanings in measurement processes.

Spatial and *temporal* granularity are extents in space and time and therefore, in principle, measurable quantities. Yet, variations throughout the observation-to-visualization life cycle of information make it hard to come up with a generally applicable measure for granularity. Given that all information is rooted in observation, granularity is best understood starting from observations and then defining measures for derived information, for example for field or object information. When talking about observations, granularity is usually referred to as *resolution* (Degbelo & Kuhn, 2012). It results from the fact that the change observed through observation necessarily has a minimum below which it cannot be detected. *Thematic* granularity is considerably harder to formalize. There is not always a clear ordering of thematic classes, nor is there always a numeric measure for thematic granularity.

Accuracy

Information is accurate if it describes something correctly. Accuracy can only be determined for a given granularity and with respect to some reference information

that is considered (more) accurate. Accuracy has been a long-standing concern for the theory and practice of spatial information (Burrough & McDonnell, 1998). While the emphasis has often been on locational accuracy, the concept of accuracy applies to all information about fields, objects, networks, and events. Accuracy and granularity together are the main indicators of the quality (or certainty) of spatial information.

The degree of accuracy is measured by the difference between the reference information and the information in question, either statistically or for individual values. For example, one may determine the accuracy of a building footprint obtained from aerial photography by comparing it to ground survey data. Inaccuracy in information results from errors in observation or analysis. Accuracy is limited by observation procedures and gets propagated through analysis.

The difference between the mean of repeated observations and a hypothetical true value is called *bias*. With adequate measuring equipment, the results of repeated measurements or calculations distribute regularly around the true value. This property is a consequence of understanding measurement as a random process.

Accuracy is often associated with the absence of *systematic* errors, such that measurement is affected only by *random* errors. Systematic errors get minimized or eliminated by calibration, which should reduce systematic errors to a level below the granularity of the information. As a consequence, best practice of reporting information requires stating only as many digits after the decimal as the combined effect of granularity and (in)accuracy permit.

Core Spatial Computations

The ambition behind defining core concepts was to reduce the complexity of spatial information to a few powerful notions that are meaningful across applications. The practical impact of this idea, however, comes from reducing the complexity of *computations* to a similarly low number. For this purpose, simply reorganizing existing GIS commands around the core concepts would not be sufficient. Instead, the approach taken in this work is to define *core computations* for each core concept. These operators constitute the semantic primitives of our language of spatial computing, which can then be combined to express more complex computations. For example, questions about the location of objects get answered by combining the operators for Location and Object. The operators can take many syntactic forms, depending on the chosen embedding of the language. This section presents a simplified algebraic specification derived from a Haskell embedding; the next section presents a Python embedding.

The length of this paper does not allow for a detailed discussion of the operators, but they should be largely self-explanatory for GIScience researchers. For a better understanding, readers should consult the concept descriptions in the previ-

ous section. We present the operator signatures in the form of a table, omitting the Haskell axioms specifying their semantics. The signatures are written following a standard practice in software engineering: the name for the operator followed by parameters for the input types (combined as a cross product) and for the output type (after the arrow). Note that the type parameters are further constrained in the full specifications. For example, the figure and ground parameters can be instantiated by objects, nodes, or edges.

	Operators	Comments
Location	isAt: figure x ground -> Bool	the contact relation
	isIn: figure x ground -> Bool	the containment relation
	position: figure -> point	a point positioning the figure
	bounds: figure -> shape	a shape bounding the figure
Field	new: [(pos,val)] x ([(posxval)] x pos -> val) -> field	interpolating a field from a list of values
	bounds: field -> shape	the domain of the field
	getValue: field x pos -> val	the value at a position
	local: field x (val -> val') -> field	computing new values at all positions
	focal: field x (pos -> val') -> field	computing new values from neighborhoods
	zonal: field x (pos -> val') -> field	computing new values from zones
Object	get: object x (object -> value) -> value	get the value of a property of the object
	is: object x object x (objectxobject -> Bool) -> Bool	are the two objects in the given relation?
	same: object x object -> Bool	are the two objects the same?
Network	nodes: network -> [node]	all nodes in a network
	edges: network -> [edge]	all edges in a network
	addNode: network x node -> network	add a node to a network
	addEdge: network x edge -> network	add an edge to a network
	degree: network -> node -> Int	degree of a node in the network
	connected: network x node x node -> Bool	are two nodes connected?
	shortestPath: network x node x node -> [node]	the shortest path between two nodes
	distance: network x node x node -> Int	the network distance (as number of nodes)
	breadthFirst :: network x node x Int -> [node]	all nodes at distance from a node
Event	when: event -> Date	the time of an event as a date
	within: event -> Period	the time of an event as a period
	same: event x event -> Bool	are the two events the same?
	during: event x event -> Bool	is the first event happening during the second?
	before: event x event -> Bool	is the first event happening before the second?
	after: event x event -> Bool	is the first event happening after the second?
	overlap: event x event -> Bool	does the first event overlap the second?

Table 2: Overview of core computations on spatial information

The choice and development of the operators is the subject of ongoing research. Specifications for the quality concepts (granularity and accuracy) are still being developed. The main goal of producing embeddings like those in Haskell and Py-

thon is in fact to test the operators for completeness, consistency, and adequacy in practical GIS projects.

A Python Embedding

To demonstrate applicability in the current technological context, we outline a Python implementation of the proposed language for spatial computing. Python, created in the 1990s in response to the verbosity and complexity of object-oriented languages such as C++, is a mature general-purpose, multi-paradigm programming language used in industry and academia in domains ranging from Web development to high-performance scientific computing. Libraries such as SciPy and PySAL[3] include spatial computing tools, and well-known spatial libraries such as GDAL provide Python bindings.[4] Current GIS display a high degree of interoperability with Python, providing APIs and bindings at multiple levels. Notably, ArcGIS[5] and QGIS[6] have adopted Python as their principal scripting language, making it a suitable testing ground for the proposed approach to core concepts. More generally, the popularity of Python among researchers and practitioners in many domains lowers the adoption barrier of the proposed approach.

The first building block for our domain-specific language consists of its abstract data types (ADT). Starting from the Haskell specifications of core concepts, we defined a set of Python classes, bearing in mind the semantic differences between these two languages. Python is strongly dynamically typed, favors procedural and object-oriented programming, and does not provide a mechanism for type parameterization. By contrast, Haskell is a functional language and strongly statically typed, i.e. all types are known and checked at compile-time. In Python, ADTs are not definable directly, and have to be embedded in concrete *classes*, while Haskell *type classes* are abstract interfaces, providing powerful support for type parameterization.

The proposed Python classes are a lower level of abstraction, between the Haskell specifications and the concrete software components that perform the spatial computation. This way, existing software resources can be harnessed and deployed in the computational workflow, without locking the user into a software-driven conceptualization of their domain knowledge. To provide an illustration of the approach, the Python class that defines the *field* concept is structured as follows (the "Cc" prefix stands for "core concepts"):

---

[3] https://pysal.readthedocs.org, http://www.scipy.org

[4] http://gdal.org/python

[5] http://resources.arcgis.com/en/communities/python

[6] http://pyqgis.org

```
class CcField(object):

    def getValue( self, position ):
        """
        @param position a position in the field
        @return the value of field at position
        """
        raise NotImplementedError("getValue")

    def local( self, fun ):
        """
        Map algebra's local operations, with a
               function to compute the new values
        @param fun a function to be locally applied on the field
        @return a new CcField field
        """
        raise NotImplementedError("local")

    def focal( self, fun ):
        """
        Map algebra's focal operations, with a kernel function to
               compute the new values based on the neighborhood of the position
        @param fun a Kernel function to be applied on the field
        @return new CcField field
        """
        raise NotImplementedError("focal")

    def zonal( self, fun ):
        """
        Map algebra's zonal operations, with a
               function to compute the new values
               based on zones containing the positions.
        @param fun a function to be zonally applied on the field
        @return new CcField field
        """
        raise NotImplementedError("zonal")
    ...
```

This class includes a getter (*getValue*), and standard Map Algebra operations. As Python does not provide an explicit mechanism to define abstract classes, we simulate the abstract nature of the class by raising exceptions at run-time (i.e. *NotImplementedError*), highlighting the fact that the class should not be instantiated directly. This class can be implemented to interface with the technological layer, in this case by handling the popular GeoTiff raster format,[7] reusing the logic defined in the software package GDAL:[8]

```
import gdal # import implementation library

class GeoTiffGdalField(CcField): # subclass of CcField
    def __init__(self, file_path):
        # load GeoTiff from file using GDAL
        self.gdalHandler = gdal.Open(file_path, GA_Update)
```

[7] http://trac.osgeo.org/geotiff

[8] Created by the Open Source Geospatial Foundation, the Geospatial Data Abstraction Library (GDAL, http://www.gdal.org) is a software package that provides an interoperability layer between a variety of raster and vector data formats.

```
    def getValue(self, position):
        # use GDAL to retrieve field value for position
        return self.gdalHandler.ReadAsArray( position.x, position.y, ... )

    ...
```

Similarly, ArcMap libraries can be harnessed through Python in an appropriate subclass:

```
import arcpy # import implementation libraries
from arcpy import sa

class GeoTiffArcMapField(CcField): # subclass of CcField
    def __init__(self, file_path):
        # load GeoTiff from file using ArcMap
        self.arcRaster = sa.Raster(file_path)
    def getValue(self, position):
        # use ArcMap to retrieve field value for position
        result = arcpy.GetCellValue_management( ... position.x, position.y )
        return result.getValue(0)
    ...
```

These concrete classes act as wrappers, encapsulating the implementation details. This approach enables the user to define operations and queries on fields, only relying on the *CcField* interface, hiding the technological and data layer from view. Similarly, we define the concepts *object* and *network* as:

```
class CcObject(object):
    def bounds( self ):
        """ @return geometric bounds of the object """
        raise NotImplementedError("bounds")

    def relation( self, obj, relType ):
        """ @return Boolean True if self and object obj are in
                    a relationship of type relType """
        raise NotImplementedError("relation")

    def property( self, prop ):
        """ @param prop the property name
            @return value of property in object """
        raise NotImplementedError("property")

    def identity( self, obj ):
        """ @param obj an object
            @return Boolean True if self and obj are identical """
        raise NotImplementedError("identity")


class CcNetwork(object):
    def nodes( self ):
        """ @return a copy of the graph nodes in a list """
        raise NotImplementedError("nodes")

    def edges( self ):
        """ @return list of edges """
        raise NotImplementedError("edges")

    def addNode( self, n ):
```

```
        """ Add a single node n """
        raise NotImplementedError("addNode")

    def addEdge( self, u, v ):
        """ Add an edge between u and v """
        raise NotImplementedError("addEdge")

    ...
```

Existing efficient vector and network data manipulation libraries, such as GDAL and NetworkX,[9] can be tapped in the corresponding implementations. Once the set of core concepts have been implemented and linked to the technological layer, the user can perform spatial computations directly in terms of the concepts.

Selecting the analysis of solar energy collection potentials as a test domain of spatial information, a typical resource consists of Shapefiles with detailed vector data representing building roofs and other viable areas for the installation of solar panels. Using the Python core concepts, the user can load these objects and perform spatial operations on them. In the following example, the user formulates the question: *is the roof of the Poultry building located in a viable area?*

```
import shapefileToObjects from coreconcepts

roofs = shapefileToObjects( "data/Rooftops.shp" ) # load objects
viableAreas = shapefileToObjects( "data/Vareas.shp" ) # load objects
polRoof = roofs[2] # get the roof for Poultry Science building from the array
print polRoof.property('name') # prints "Poultry Science"
# find answer to question
valid = any(map(lambda area: polRoof.relation(area,'within'), viableAreas))
```

Because of their foundational nature, the core concepts can be deployed and assembled to represent a wide variety of domain entities, enabling the user to flexibly model their scenarios and even include multiple conceptualizations. Thus, the energy analyst can define a roof type as a field for some purposes (e.g., modeling its topography) *and* as an object for others, formulating spatial questions that involve both perspectives:

```
class Roof(CcField,CcObject):
    ...

roofA = Roof( 'some_data_source' ) # load a roof
roofB = Roof( 'some_data_source' ) # load another roof

# does roofA have a smaller area than roofB?
answer = roofA.property('area') < roofB.property('area')
# is value of field position (3,5) in roofA higher than
# half of the same position in roofB?
answer = roofA.getValue([3,5]) > roofB.getValue([3,5])/2
```

As we have shown in this section, the embedding of core concepts in Python can provide a widely usable and modular conceptual layer to organize domain-specific

[9] https://networkx.github.io

spatial knowledge. The advantages of this approach will be particularly evident in the context of information integration from different domains, providing a modeling framework as well as a computing toolkit to facilitate communication between GIS practitioners and domain experts.

Conclusions and Outlook

We have outlined the design rationale and an early implementation of a language for the "domain" of spatial computing. Our ultimate goal is a high-level language executable on existing commercial and open source spatial computing platforms, in particular Geographic Information Systems (GIS). So far, we have specified a set of core concepts and, for each of them, a set of core computations. These specifications are now being translated (by hand) into Python scripts, as well as into Haskell data types and foreign function calls, in both cases allowing for calls to commercial or open source GIS platforms.

The paper first described the theory of core concepts of spatial information that underpins the language, which includes five core content concepts (*location*, *field*, *object*, *network*, and *event*), complemented by two core quality concepts (*granularity* and *accuracy*). After providing a formal specification of a set of spatial computations relying on these concepts, we outlined an ongoing embedding in Python, showing how this package can function as a mediation layer between the core concepts and existing technological layers that encode the data and perform the computations.

More interdisciplinary research is needed to achieve our vision. The Python embedding will be stress-tested in more realistic scenarios, providing feedback to revise both the formal specifications and the software embedding, but possibly also the concept selection. Use cases set in different domains, ranging from ecology to economics and history, will help demonstrate the cross-domain transferability of the core concepts within and beyond the traditional scope of GIS applications.

A different route to take with this idea is to design and implement Application Programming Interfaces (API) on top of spatial data repositories and spatial database management systems. For example, popular open data like OpenStreetMap or the US Census TIGER data would benefit from some generic computing layer through which to query and analyze them from certain perspectives (for example, seeing them as representing networks or sets of objects).

Acknowledgments

Contributions to the Python embedding and testing from Michel Zimmer, Marc Tim Thiemann, and Eric Ahlgren as well as funding from the UCSB Center for Spatial Studies are gratefully acknowledged.

References

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

Baumann, P. (2010). The OGC web coverage processing service (WCPS) standard. *Geoinformatica*, *14*(4), 447–479.

Burrough, P. A., & Frank, A. U. (1996). *Geographic objects with indeterminate boundaries*. London: Taylor&Francis.

Burrough, P. A., & McDonnell, R. (1998). *Principles of geographical information systems*. Oxford, UK: Oxford University Press.

Camara, G., Egenhofer, M. J., Ferreira, K., Andrade, P., Queiroz, G., Sanchez, A., … Vinhas, L. (2014). Fields as a Generic Data Type for Big Spatial Data. In *Geographic Information Science* (pp. 159–172). Springer-Verlag.

Couclelis, H. (1992). People manipulate objects (but cultivate fields): Beyond the raster-vector debate in GIS. In A. U. Frank, I. Campari, & U. Formentini (Eds.), *Theories and methods of spatio-temporal reasoning in geographic space* (pp. 65–77). Berlin: Springer-Verlag.

Degbelo, A., & Kuhn, W. (2012). A Conceptual Analysis of Resolution. In *GeoInfo - XIII Brazilian Symposium on GeoInformatics, November 25-28 2012, Campos do Jordão, Brasil* (pp. 11–22).

Donnelly, M. (2005). Relative Places. *Applied Ontology*, *1*, 55–75.

Egenhofer, M. J., & Kuhn, W. (1999). Interacting with Geographic Information Systems. In M. F. Goodchild, D. J. Maguire, D. W. Rhind, & P. Longley (Eds.), *Geographical Information Systems: Principles, Techniques, Applications, and Management* (2nd ed., Vol. 1, pp. 401–412). New York, NY: Wiley.

Galton, A. (2004). Fields and Objects in Space, Time, and Space-time. *Spatial Cognition & Computation*, *4*(1), 39–68.

Ghosh, D. (2011). DSL for the uninitiated. *Communications of the ACM*, *54*(7), 44.

Golledge, R. G. (1995). Primitives of spatial knowledge. In T. L. Nyerges, D. M. Mark, R. Laurini, & M. J. Egenhofer (Eds.), *Cognitive aspects of human-computer interaction for geographic information systems* (pp. 29–44). Springer.

Goodchild, M. F., Yuan, M., & Cova, T. J. (2007). Towards a general theory of geographic representation in GIS. *International Journal of Geographical Information Science*, *21*(3), 239–260.

Janelle, D. G., & Goodchild, M. F. (2011). Concepts, Principles, Tools, and Challenges in Spatially Integrated Social Science. In *The SAGE Handbook of GIS and Society*. SAGE Publications.

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

Newman, M. E. J. (2010). *Networks*. Oxford University Press.

Norman, D. A. (1986). Cognitive Engineering. In D. Norman & S. Draper (Eds.), *User centered system design* (pp. 31–61). Hillsdale, NJ: Lawrence Erlbaum Associates.

Rosenfeld, A. (1986). "Continuous" functions on digital pictures. *Pattern Recognition Letters*, *4*(3), 177–184.

Talmy, L. (1983). How Language Structures Space. In H. L. Pick & L. P. Acredolo (Eds.), *Spatial Orientation* (pp. 225–282). New York and London: Plenum Press.