Thesis report

The relevance of semantics for GIS workflow synthesis.

Author:
Rogier Meerlo
R.R.Meerlo@gmail.com

Supervisor:
Dr. Simon Scheider
S.Scheider@uu.nl

Responsible professor:
Prof. dr. Stan Geertman
S.C.M.Geertman@uu.nl

1 Contents

List of abbreviations

DAML	- DARPA Agent Markup Language
GIS	- Geographic Information System
OWL	- Web Ontology Language
RDF	- Resource Description Framework
SWfMS	- Scientific Workflow Management System
STL	- Semantic Linear Time Logic
WfMC	- Workflow Management Coalition
WFMS	- Workflow Management System
XML	- Extensible Markup Language

List of figures

List of tables

Abstract

In order to solve geo-analytical problems users of GIS must often compose workflows in GIS software. In order to do so, the user must be aware of the possible functionalities and which tools can be connected in a meaningful way in order to obtain their goal. A proposed solution for assisting GIS users in creating meaningful workflows is the creation of a GIS workflow synthesis engine. This type of support system would allow users to automatically generate a workflow capable of bridging the gap between the available data and the goal specified by the user. The main objective of this thesis is the creation and evaluation of an ontology-based GIS workflow synthesis prototype. This goal was achieved by formalizing the semantic data type ontology and annotating tools and datasets using semantic web technologies. These technical elements were combined in a workflow synthesis engine in order to create a functional GIS workflow synthesis prototype. The quality of such a workflow synthesis system was evaluated by generating workflows for five geo-analytical questions and evaluating each of the first 20 workflows by hand in accordance with a novel workflow evaluation framework. This process was repeated with a limited workflow synthesis prototype that could only take traditional GIS semantics into account in order to provide a reference for the capacity of the semantic data type ontology. The work done within this thesis has proven that a GIS workflow synthesis prototype based on the semantic data type ontology has the potential for creating meaningful workflows, and has a higher capacity for answering spatial questions than a synthesis engine based on the traditional semantics of line, points, polygons, vectors, and rasters.

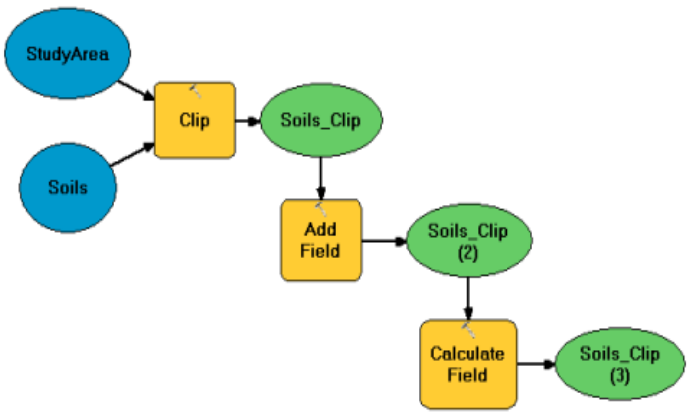
2 Introduction

The term *"Geographic information system"* (GIS) was dubbed by Roger Tomlinson in the year 1968 and first published in 1969 (Tomlinson, 1969). This marked the start for a new field of information technology, one focused on geographic information. Between now and then the field of GIS has developed into a vast and complex field, with technological developments such as geostatistics, geocoding, network analysis, hydrological modeling and much more. The development of such technologies has given the users of GIS a lot of new methods to handle geographic information in order to answer spatial questions. Often users of such technologies do not simply use a single function but compose functionalities into a workflow in order to meet their needs. In order to generate such a workflow, the user must be aware of the possible functionalities and which components can be connected in a meaningful way in order to obtain their goal. Due to the expansive growth of geographic information technology, the necessary knowledge of possible functionalities and proper connections is becoming ever more extensive. Operating in the field of geographic information systems thus often requires an expert with extensive knowledge of GIS in order to create and operate workflows.

GIS is not the first field confronted with expansive growth of technology, and in order to navigate such complexity, multiple fields have turned to Scientific Workflow Management Systems (WFMS). A scientific workflow management system is *a system that supports the specification, modification, run, re-run, and monitoring of scientific workflows using the workflow logic to control the order of executing workflow tasks* (Lin et al., 2008). By using WFMS scientists can get a more clear view on their resources, simplify the process of workflow assembly, store and replicate created workflows and facilitate their execution and monitoring (Chen et al., 2003; Lamprecht, Margaria, & Steffen, 2009; Martin et al., 2005). A familiar example of such a WFMS for GIS users would be the ArcGIS model builder (Figure 1.1). In this example, the complex task of manually scripting a workflow is replaced by the simpler composition of a workflow using a graphic user interface. Using these WFMS users can more easily create, edit and run standardized workflows for ArcMap.

Figure 3. 1 Model builder workflow and Arcpy workflow

```
1
2
3  import numpy
4  import arcpy, arcinfo
5  from arcpy import env
6  from math import radians, sin, cos
7  arcpy.CheckOutExtension("spatial")
8
9  # In order for this tool to work properly the data must be in a projected co
10
11 #scratchspace = raw_input ("what is the drive location used for this project
12 scratchSpace = "C:\\GIMA"
13
14 #creating an temporary file for the exported rasters, it's impossible to sum
15 arcpy.CreateFolder_management (scratchSpace, "TempRasz")
16
17 #arcpy.env.workspace = arcpy.env.scratchWorkspace = raw_input("What the driv
18 arcpy.env.workspace = arcpy.env.scratchWorkspace = "C:\\GIMA\\indepth.gdb"
19 env = arcpy.env.workspace
20 arcpy.env.overwriteOutput = True
21
22 # fc= "C:\\GIMA\\indepth.gdb\\" + raw_input("What is the name of the point f
23 fc= "C:\\GIMA\\indepth.gdb\\TempTrees"
24 arcpy.env.extent = fc
25
26
27 high = 45.0
28 medium = 30.0
29 low = 15.0
```

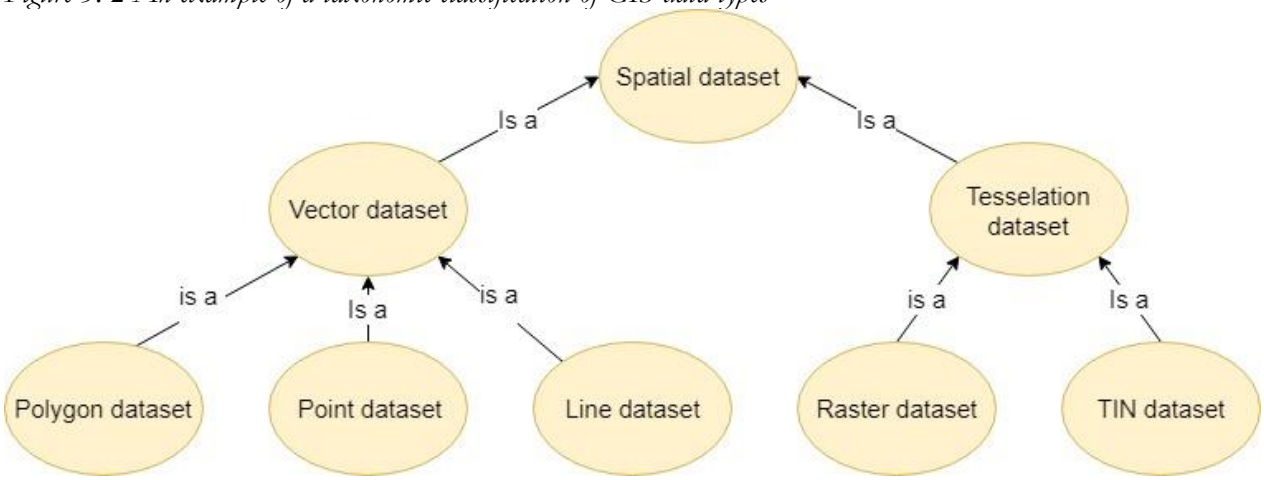


Even though these types of WFMS can aid in multiple aspects of workflow development and alleviate some of the complexity of workflow creation, the creator of a workflow must manually create the workflow. This requires the creator to be of which components exist and how they can be connected in a meaningful way (S Scheider & Tomko, 2016). The time and knowledge-intensive nature of manual workflow creation result in limited usability and scalability and allows for the creation of invalid workflows due to human error (Gil, 2007).

In order to alleviate some of the pressure on human expertise and time, the field of scientific workflows has developed a new type of workflow support system by the name of workflow synthesis (Chun, Atluri, & Adam, 2002; Kasalica & Lamprecht, 2018). The goal of such a system is to automatically compose a workflow that bridges the gap between the available input and the specified outcome. The advantage of such a system is that the user would no longer require extensive expert knowledge in order to know which methods can be combined meaningfully. This is because the necessary knowledge of all available functionalities and their applicability constraints would be captured within the formal framework of the support system. Secondly, such a system would drastically reduce the creation time of workflows and reduce the possibility of human error. The effectiveness of such a workflow synthesis system has already been demonstrated in the field of bioinformatics, allowing users to automatically validate the meaningfulness of a connection between processes, or suggest processes to bridge the gap between data types, for example for the creation of a simple phylogenetic analysis workflow (Lamprecht et al., 2009).

These types of systems are based on the modal logic of Semantic Linear Time Logic (SLTL), which combines relative time with taxonomic classifications of data types and processes (Steffen, Margaria, & Freitag, 1993). A taxonomy is a simple ontology that relates semantic entities in terms of a *"is a"* relation, where lower instances are considered a subclass of higher classes (figure 1.2).

Figure 3. 2 An example of a taxonomic classification of GIS data types



For synthesis to successfully bridge the gap between input data types and its goal specification it requires a machine-readable formalization of the taxonomy of the domain knowledge and a machine-readable formalization of annotated data and services according to the specified semantic properties (Lamprecht, Naujokat, Margaria, & Steffen, 2010).

Most GIS software packages in the field of GIS have a program-specific WFMS such as ESRI Modelbuilder, Geocortex for ESRI or wxGUI Graphical Modeler for GRASS (Esri, n.d.-a; OSGEO, n.d.; VertiGIS, n.d.). These modules offer an easy to use graphic user interface for easier workflow construction and allow for the storage, sharing, editing, use and re-use of workflows. However, these WFMS do not support workflow synthesis. This might be because map-making, as well as spatial analysis, are full of semantic intricacies that are complex to

formalize as the construction of GIS workflows goes beyond simply fitting data types to inputs and outputs (Hofer, Mäs, Brauner, & Bernard, 2017) and requires a working understanding of the domain knowledge (S Scheider & Tomko, 2016; Stasch, Scheider, Pebesma, & Kuhn, 2014). It might be that this inherent complexity has prevented GIS software creators from creating a synthesis module for their WFMS. However outside the world of software packages some successful work has been done in regards to the automatic chaining of geo-functionalities and geo-data based on semantic descriptions (Athanasis, Kalabokidis, Vaitis, & Soulakellis, 2009; Lemmens et al., 2006; Lutz, 2007; Yue, Di, Yang, Yu, & Zhao, 2007). A common conclusion among their works is the fact that successful workflow synthesis requires a formalized ontology of the domain's semantics and a rich annotation of the available data and tools (Lamprecht et al., 2009).

The field of bioinformatics already has such an ontology by the name of EDAM (Ison et al., 2013). EDAM is extensive enough to facilitate workflow synthesis, but as of yet the field of GIS misses such an ontology (Kasalica & Lamprecht, 2019). Yet there is no need to work in a complete knowledge vacuum. The development of GIS was accompanied by the development of the field of Geographic Information Science, also called Geographical Information Science (GIScience). GIScience is still a pretty new face in the world of science. Being only defined as a field of science by the British geographer Michael Goodchild in the 1990s (Goodchild, 2010). Geographic information science is a subset of information science that studies the nature and properties of geographic information according to scientific principles (Goodchild, 1992). The scientists within this field have made notable progress in regards to the formalization of geospatial knowledge (Albrecht, Derman, & Ramasubramanian, 2008a; Lehmann et al., 2015; S Scheider & Tomko, 2016; Simon Scheider, 2019). Multiple formalized ontologies have been created for different research project (Fitzner, Hoffmann, & Klien, 2011; Lemmens et al., 2006; Lutz, 2007; Stasch et al., 2014). However, these ontologies are often specialized either in the discovery, comparison, composition or execution of operations and workflows. So far there isn't a singular ontology that can cover the full spectrum necessary for automatic workflow composition (Hofer et al., 2017).

Even though the current GIS ontologies cannot fully cover the entirety of the domain's knowledge, they can still contribute to the capabilities of workflow synthesis. By formalizing an ontology, a synthesis engine is capable of reading the ontology and interpreting the meaning and relations of the semantics within the ontology. When this formalized ontology is combined with annotated GIS tools and datasets in a synthesis engine it is possible to create a GIS workflow synthesis prototype. With this prototype, users are capable of bridging the gap between the input data and the goal specification, creating a novel tool for GIS users.

An essential part of the workflow synthesis prototype is the selected ontology as it provides the basis for the entire structure of the prototype. GIS doesn't have a fully developed workflow synthesis ontology like EDAM. Instead, this thesis will implement the semantic data type ontology by Simon Scheider (2019). By formalizing said ontology and annotating GIS tools and datasets in accordance with the included semantics and combining them in a synthesis engine. As this ontology isn't capable of covering the entire domain knowledge necessary for workflow synthesis it is necessary to evaluate the selected ontology and resulting prototype. Ontologies can be evaluated by measuring their functional dimension. This is done by matching the implementation of the ontology against a experts judgment (Gangemi, Catenacci, Ciaramita, & Lehmann, 2005). If the implementations of the ontology are judged to be valid the ontology can be said to be valid. In the case of an ontology meant for GIS workflow synthesis, the implementation of the ontology is the generated workflows. The expert's judgments consist of the manual evaluation of the generated workflows by a GIS expert. However, there is no standardized framework to evaluate GIS workflows. As such this thesis will create its own novel evaluation framework in order to provide the experts judgement in a standardized fashion. The evaluated workflows will be generated by implementing the workflow synthesis prototype for a

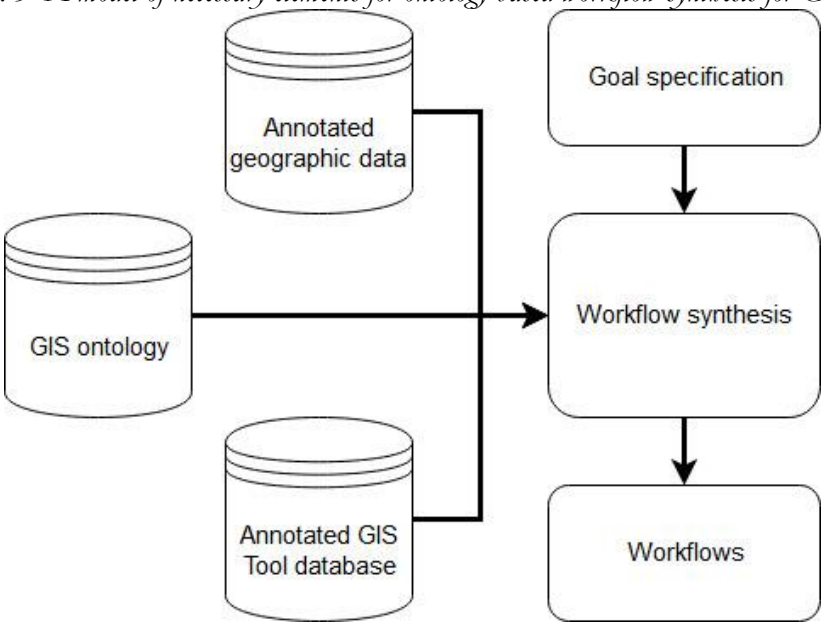
set of scenario's based on the creation of a livability atlas for the municipality of Amsterdam. by evaluating the resulting workflows it is possible to evaluate and validate the used ontology and demonstrate it's relevance for workflow synthesis, while also demonstrating the validity of the prototype.

3 Research objectives

The aim of this thesis will be threefold, the first goal is the creation of a novel GIS workflow synthesis prototype that is capable of bridging the gap between data states and goal specifications. The creation of such a system requires the identification of the necessary technical elements for workflow synthesis (figure 1.3) and the availability of such elements for the field of GIS. Any element that isn't readily available will have to be created for the purpose of this study. These elements will only be developed in so far that they are sufficient for a workflow synthesis prototype and will not cover the full extent necessary for fully functional workflow synthesis.

The geographic database, GIS tool database, GIS ontology, and the goal specification are all based on the semantics used in the domain of GIS. As such this study will aim to make some headway into researching what semantics are relevant for workflow synthesis. The relevant semantics will be those aspects of geographic data or GIS functionalities that influence the validity of a connection between data and tools. For the purpose of this study, the number of semantic properties will be limited to what is relevant to the particular GIS scenarios tested with the workflow synthesis prototype. This is because the creation of a holistic GIS ontology would be very time consuming and outside the scope of this study. Based on the selected semantics, an annotated tool database, an annotated geographic database, and a machine-readable ontology will be created. These elements will be combined in a synthesis engine in order to create a GIS workflow synthesis prototype. The number of tools and datasets will be limited to a small sample of all possible GIS tools and datasets types as the annotation process are time-consuming and the synthesis prototype only needs a limited set of data and tools in order to prove the viability of such a system.

Figure 3. 3 A model of necessary elements for ontology-based workflow synthesis for GIS.



The second goal of this thesis is the implementation of the workflow synthesis prototype in a limited set of scenarios' in order to test the prototype in a realistic setting. The scenarios for this trial run will be based on the hypothetical creation of a liveability atlas for the municipality of Amsterdam in ArcGIS pro (Esri, n.d.-b). The composition of a liveability atlas will be run on two different versions of the workflow synthesis prototype. One version will be based on the full ontology and will take all semantics included in the ontology into account. The second version will be based on a limited ontology that only takes standard geodata types such as point, line, polygon, and raster into account. The full ontology used in this thesis will be based on the semantic data type signatures ontology by Simon Scheider (2019). For the purpose of this study, elements of the data types signatures ontology that relate to events and networks will not be taken into account as the operationalization of these types was out of scope.

The third and final goal of this thesis is the evaluation of the generated workflows in order to evaluate the synthesis prototype and the functional dimension of the full ontology and the limited ontology. This will be done according to a novel evaluation framework that is based on the identification of workflow error types. If the workflow is judged to be incorrect according to the expert the cause is categorized according to the error types defined in the framework. The type of errors defined within this thesis shall be limited to an initial draft of errors types identified during the manual evaluation of workflows, as a full taxonomy of all the possible error types within a GIS workflow lies outside of the scope of this thesis. By comparing the error rate within each category of errors for both sets of workflows it is possible to identify the capacity of each ontology to answer questions within a GIS. By comparing the error rates of each ontology we can compare their respective capacity and test the relevance of the semantics that are included in the full data types signature ontology.

3.1.1 Research questions

From these research objectives, the following main research questions can be defined. In order to answer these main research questions, this study answers the following sub-questions. Further indented questions are considered further sub-questions:

How can an ontology-based workflow synthesis prototype for automatic GIS workflow composition be created?

- How can GIS workflow synthesis be realized using semantic web technology?
    - What technical elements are necessary for GIS workflow synthesis?
    - What is the current status of these elements for the field of GIS?
    - How can we create the necessary technical elements for GIS a workflow synthesis prototype?

How can synthesized workflows be evaluated for their capacity to answer geo-analytical questions?

- What workflow evaluation methods are available for workflows?
- What type of errors can be made in the workflows resulting from workflow synthesis?

What is the capacity of the semantic data type ontology to answer geo-analytical questions?

- How can ontologies be evaluated?

4 Theoretical framework

The aim of this chapter is to place the research done in this thesis within its scientific context. This chapter will be divided into three distinct sub-chapters. The first sub-chapter will provide the theoretical background for the concept of processes and workflows and provide a short summary of the development of workflow support systems leading up to workflow synthesis. The goal of this chapter is to clearly define the type of workflows used in this thesis and to put the development of workflow synthesis within its context. It also aims to prove the relevance of workflow support systems by illustrating the need for such systems within the field of science. The second sub-chapter will consist of the description of semantic-based workflow synthesis and the technical elements necessary for it. The goal of this chapter is to provide the scientific basis for workflow synthesis that will be used in the methodology section. The final sub-chapter will introduce the data types ontology (Simon Scheider, 2019) used within this thesis and describe the propositions made within it with formal logic. The goal of this chapter is to show the reasoning behind the ontology and to illustrate some of its implications.

4.1 Workflows and workflow support systems

Workflows have existed ever since man started to create things. A very early example of such a workflow would be the caveman attaching a piece of sharpened stone to a wooden stick to create a spear. Later in history, the execution of workflows was followed by the formalization of workflows. This formalization was initially focused on the field of business. In this field, an entire set of ideas, tools, and technologies for workflows have been developed to meet the needs of businesses (Barga & Gannon, 2007). This body of work is a good starting point to understand the concept of workflows.

The concept of workflow evolved from the concept of process in the business world. The concept of the process was created by the desire to increase efficiency within businesses by separating work activities into well-defined tasks, roles, rules, and procedures. Initially, the processes were all executed by humans, but with the development of new technologies, certain processes could also be executed by machines and computers (Georgakellos & Macris, 2009). These processes can be categorized into three different categories: material, information, and business (Medina-Mora, Winograd, Flores, & Flores, 1993).

The material process is a process that is rooted in the physical world. Any activity that involves physical things being moved or changed in its state constitutes a material process. An example of such a process would be the excavation of metal ore from a mine (figure 4.1).

Figure 4. 1 Example of material processes in a workflow.

However, processes in a workflow aren't just simply limited to interaction with the physical domain. The second type of process is the information process, this process focusses on the interaction of actors and computers with information. This interaction can constitute a wide arrange of actions such as the creation, retrieval, storage, manipulation, transference and analysis of information. An example of such a process would be the placement of an order (Figure 4.2). Although a physical paper carrying the representation of an order might be material, the goal of such a process is the transfer of the information on the paper.

Figure 4. 2 Example of information processes in a workflow.

The final type of process is the business process. This type of process captures the purpose of the process instead of the action itself. Business processes are market-centered descriptions of an organization's activities implemented as information processes and/or material processes. The business process is engineered to fulfill an idea such as satisfying a certain need in return for payment (Georgakellos & Macris, 2009).

Figure 4. 3 Example of business processes in a workflow.

The processes that will be described in this thesis are limited to processes involved with the usage of GIS. *Geographic information system (GIS) is a system designed to capture, store, analyze, manage and present spatial or geographic data. The processes used in GIS applications allow users to create interactive queries, analyze spatial data information, edit in maps, and present the results of all these operations* (Clarke, 1986; Maliene, Grigonis, Palevičius, & Griffiths, 2011). The definition of GIS clearly exemplifies that GIS focusses on information processes. So for the purpose of this thesis, we will approach GIS workflows as consisting solely of information processes. By linearly combining multiple processes a workflow can be created as can be seen in diagram 2.1, 2.2 and 2.3.

4.2 Workflow

Workflows are closely related to business processes and have been intensively studied for the field of business (Barga & Gannon, 2007; Ludäscher, Weske, McPhillips, & Bowers, 2009). However, there is little agreement on the definition of workflows within the field of business. When people talk about workflows they might be referring to a business process, the specification of a business process, software that implements and automates processes or software that supports participants of a business process with the coordination of the project and or the collaboration between the participants (Georgakellos & Macris, 2009). The definitions also differ in terms of the granularity of workflows. Certain definitions define a workflow as a part of a business process, while others define a workflow as the entirety of a business process. Other definitions define a workflow as consisting of multiple business processes (table 2.1). This ambiguity in terms of workflow makes it difficult to draw on the workflow research done for the field of business.

table 4. 1 Examples of workflow definitions categorized according to workflow granularity.

Workflow Granularity	Definition
Part of a business process	A workflow is a work that is recast as a series of people-based transaction and a series of workflows form a business process (Frye, 1994).
The entirety of a business process	A workflow is a collection of tasks organized to accomplish some business process (Georgakellos & Macris, 2009) Workflow is the process by which individual tasks come together to complete a transaction (Transaction is defined as a clearly defined business process) within an enterprise (Action Technologies Inc, 1993).
A set of business processes	Workflows are defined as activities involving the coordinated execution of multiple tasks performed by different processing entities (Rusinkiewicz & Sheth, 1995).

Attempts have been made to standardize the workflow definitions and terms, mainly since 1993 by the Workflow Management Coalition (WfMC). WfMC defines workflow as "*the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"* (Workflow Management Coalition, 1999). This definition became a commonly accepted standard that provides a basis for many workflow approaches. Besides the definition of workflows, the WfMC also created a set of reference models and standards. In order to meet the needs of commercial enterprises an entire industry of tools and technologies devoted to workflow management has been created (Barga & Gannon, 2007). So the field of the business workflow can be considered a fully mature and well-established field (Ludäscher et al., 2009). However, despite the fact that these tools, technologies, standards, and models are readily available, the scientific community has not widely adopted these standards and instead created its own workflow paradigm (Barga & Gannon, 2007). This is due to the fact that there are distinct differences between business workflows and scientific workflows and their usage in their respective field.

4.2.1 Scientific workflows

An alternative paradigm for workflows is the scientific workflow paradigm first coined by Vouk and Singh in 1996 (Vouk & Singh, 1996). "*A scientific workflow is a formal specification of a scientific process, which represents, streamlines and automates the steps from dataset selection and integration, computation and analysis to final data product presentation and visualization* (Lin et al., 2008). Scientific workflows serve a dual function, as detailed documentation of the scientific method used for an experiment and as a re-usable executable specification for future research. The workflows are composed of information processes such as data movement, data transformation, data analysis, and data visualization. The constructs used in such a workflow are often determined by the workflow system in which the workflow was created (Deelman, Gannon, Shields, & Taylor, 2009).

This paradigm differs from the business paradigm as it focuses on the specification and automatization of a workflow instead of only the automatization. It also differs as the type of processes captured within this paradigm mainly consists of information processes and doesn't leave as much room for human tasks as these can't always be fully automated (Sonntag, Karastoyanova, & Deelman, 2010). As such the scientific workflow definitions are more in line with the idea of workflows within a GIS environment and will be the definition used for this thesis.

4.2.2 Workflow usage

The difference in nature between business workflows and scientific workflows and their area of implementation also results in differences in the usage of the workflows. One of the differences between business workflows and their scientific counterpart is the focus of business workflows on security and integrity of a sequence of actions. Business workflows serve customers as a service, this necessitates a robust execution (Barga & Gannon, 2007; Sonntag et al., 2010). Due to the focus on security and integrity business workflows are less dynamic and evolving in nature compared to scientific workflows. Business workflows are often predefined and executed in a routine fashion (Gil, 2007; Sonntag et al., 2010). In comparison, the nature of scientific work is often exploratory. Due to this fact, the processes in the workflow are often modified as the research proceeds or might even be the subject of the research itself and be modified as an experiment (Barga & Gannon, 2007; Bechhofer et al., 2013).

It is also quite common for scientific workflows to be stored and shared. This is done to aid other researches in their research but also to provide a basis for the reproducibility of research (Bechhofer et al., 2013; Deelman et al., 2009; Ludäscher et al., 2009). Workflow reuse and sharing have several advantages, such as allowing for the improvement of workflow quality through incremental development by multiple independent users. It also allows for an increase in efficiency, as secondary or tertiary users of workflows don't need to create a workflow from scratch (Goderis, Sattler, Lord, & Goble, 2005). Not only can shared workflow be re-used they can also be used as sub-workflows for other research in order to increase the scope of potential research (Zhang, Tan, Alexander, Foster, & Madduri, 2011). This incremental/evolutionary workflow development allows for a more efficient scientific process, where one can build on the progress of others more easily (Garijo et al., 2014). The second advantage of sharing workflows is that it facilitates the reproduction of scientific work in order to evaluate the results. A key property of science is repeatability (Gil et al., 2007), if an experiment doesn't yield the same results when repeated it can be considered invalid. The publishing of workflows facilitates the efficient verification of research results as the workflow doesn't have to be reconstructed in its entirety (Bechhofer et al., 2013). There is also a difference in the production of business and scientific workflows. Business workflows are often produced by professional software and business flow engineers, whereas scientists often produce their own scientific workflows (Barga & Gannon, 2007).

4.2.3 Limitations of unassisted workflow composition

The often unaided creation of workflows by scientist has some clear limitations for the usage of workflows in the field of science. Scientific workflows are often constructed by the scientists themselves and even though they might be experts at their own field they often lack expertise in the field of workflows (Barga & Gannon, 2007). For the execution and specification of workflows scientist often create ad hoc scripts that can handle the iterative nature of workflows. In order to create these scripts, a scientist must have knowledge of the processes, the execution environment, and the scripting language. This is quite a large demand for skill and as such not every scientist is capable of creating ad hoc scripts.

The nature of ad hoc scripts also interferes with the desired reusability and shareability of workflows. Within each script, the data and execution environment need to specified for each process, and the data must be placed in the exact location specified. When the workflow is shared the data often won't be on that exact location and the execution environment might be incomplete, resulting in workflow errors. So even when the script already has been constructed, the secondary or tertiary scientists often won't be able to understand error conditions or repair failures within the scripts (Gil, 2007). The ad hoc nature of the script also means it lacks a clear structure making it difficult to extend the workflow, especially by people not directly involved in the creation of the script.

When the execution of a workflow doesn't require a complex iterative execution scientist can also specify workflows by hand in text editors and execute them manually. Although this method requires less technical know-how it is very limited in terms of scalability and efficiency as the manual execution of workflows is quite labor-intensive.

The previously specified nature of scientific workflows demands that workflows can be easily saved, used, reused, edited and shared. However the nature of unassisted workflow composition doesn't easily allow for such things due to the complexity of ad hoc scripts and the inefficiency of manually executed workflows, this can stand in the way of reusing workflows (Garijo et al., 2014).

4.2.4 Workflow management systems

In order to support scientists in the creation, storage, sharing, and execution of scientific workflows, scientific workflow management systems (SWfMS) were developed. This type of system was based on Workflow Management Systems (WFMS). WFMS was originally developed for the business world. The Workflow Management Coalition defines a WFMS as *"A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications"* (Workflow Management Coalition, 1999)". WFMS have made it possible to bridge the gap between workflow specification and execution, meaning that they used the specification in order to execute the workflow. By doing so WFMS could support document management, imaging, application launching and/or human coordination, collaboration and co-decision (Georgakopoulos, Hornick, & Sheth, 1995). Initially, the field of WFMS was only small with an estimated market value of $100 million in 1991 (McCready, 1992) but the industry rapidly expanded to an estimated market value of 5.2 billion in 2018 (Zion Market Research, 2019). This growth went accompanied by the development of many new products for WFMS, including the development of workflow management systems specifically designed for scientific workflows.

A scientific workflow management system is *a system that supports the specification, modification, run, re-run, and monitoring of a scientific workflow using the workflow logic to control the order of executing workflow tasks* (Lin et al., 2008). Scientific workflow management systems are a quite recent development only emerging since the last two decades (Kasalica & Lamprecht, 2018) Just like the difference between business workflows and scientific workflows the focus of an SWfMS is slightly different to the focus of WFMS. SWfMS are often specialized in catering to the needs of a specific scientific domain or software package. Some examples of known SWfMS are Kepler, Pegasus, Taverna and VisTrails (Altintas et al., n.d.; Deelman, Mehta, Kim, Gil, & Ratnakar, 2007; Freire & Silva, 2012; Wolstencroft et al., 2013), and many more are currently under development. These systems simplify the assembly of complex scientific workflows (Kasalica & Lamprecht, 2018).

In order to specify workflows, SWfMSs often use Graphical user interfaces where human coordinators set the task and interdependencies conditions of the workflow (Chun et al., 2002). This interface is a lot easier to use and less technical than having to code the specifications of the ad hoc script manually. Once the workflow has been specified it can be stored. This is often done in a proprietary data format of the specific SWfMS. This stored version of the workflow can easily be opened in the same system in order to make edits to the workflow without having to specify the entire workflow again. This type of editing facilitates the experimental nature of scientific workflows and allows for easier development of workflows through multiple iterations. The fact that these type of systems store the workflow in a digital format also allows for the sharing of workflows, as any scientist with the same management system can easily open the workflow document to observe and evaluate the shared workflow or possibly edit the workflow if so desired. SWfMSs are often integrated with an execution engine allowing the users of such a

system to run or re-run the workflow. This integration saves time as the user of such a system can also verify the workflow by running it. This creates a quick and easy loop between composition, testing, editing, and execution.

The introduction of SWfMS such as model builder for ESRI has aided in relieving some of the work pressure of creating workflows by offering an easy to use graphical user interface for workflow construction and by allowing workflows to be saved, shared, used reused and edited. These features reduce the required knowledge of the processes used in the workflow, the execution environment, and scripting languages, allowing for easier creation, editing, use, reuse and sharing of workflows.

4.2.4.1 Limitations of scientific workflow management systems.

The usage of scientific workflow management systems supports scientist in their usage of scientific workflows, however, these systems still have their limitations. The first limitation of scientific workflow management systems is that fact that creators of workflows still must be aware of which component exist, which components can be connected and how they can be connected in a meaningful way in terms of the input and output data types and formats (Kasalica & Lamprecht, 2018; S Scheider & Tomko, 2016). This expertise is costly and with the ever-increasing complexity of workflows caused by the growing size of databases used and the technological advancement of tools the user of such an SWfMS would need ever increasingly more knowledge.

A way to mitigate the necessity of workflow creation is the sharing of workflows but this method also isn't without its limitations. There are three types of reuse based on the person executing the reuse of the workflow. Reuse by third parties who the workflow author never met, reuse by collaborators and personal reuse (Goderis et al., 2005). In order for third parties or collaborators to reuse or edit the workflow, the execution environment has to be copied exactly (Garijo, Garijo, De Informática, & Gil, 2012). There is also little interoperability between SWfMS, thus a workflow created in one system often can't be executed in a different execution engine or SWfMS. There have been efforts to bridge the interoperability between different SWfMS. However the translation between SWfMSs sometimes loses detail as the capabilities of each and every SWfMS are not fully captured in the translation framework (Oliveira, de Oliveira, & Braganholo, 2015). This limits the reusability of workflows because it becomes difficult or impossible to combine workflows that were created in different SWfMSs without human translation or translation through third-party applications. Despite the limitations of reuse between these types of systems, there is still a large push toward the sharing of workflows. In the case of workflow sharing between collaborators, the workflow is often shared by sending the file containing the workflow through an email or giving the collaborator access to an online database containing the workflow specification. However, the sharing of workflows to third parties has to deal with a bit more complexity as the workflows need to be discoverable for external parties. Several projects have been launched to make workflow specifications more easily searchable and sharable (Ubels, 2018). These projects have met with some success but are still limited by the complexity of the field. One of the difficulties of these projects is the formalization of workflow descriptions. In order for workflows to be re-used they need to be found. The current methods for searching workflows leave a lot of ambiguity in terms of workflow description (Garijo et al., 2012). Any found workflow also needs to be checked for correct usage of processes and data. This means that the user of SWfMS still needs expert knowledge in order to construct workflows or to verify the validity of a workflow for reuse.

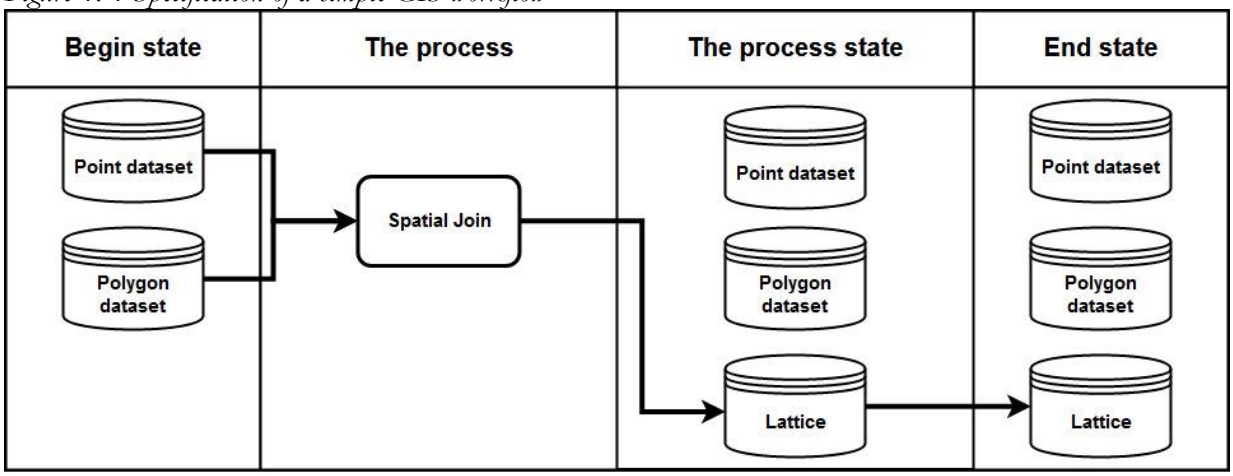
The sharing of workflows can't fully stem the problem of increasingly complex workflow specification. To face this problem the development of SWfMS have turned to a new type of support system for supporting scientists in the creation of scientific workflows, namely assisted workflow composition sometimes also referred to as automated workflow generation or

workflow synthesis. This type of support system shifts the focus from assisting users in the manual construction, validation or retrieval of workflows, to the automatic generation and validation of complete or partial workflows (Chen et al., 2003; Chun et al., 2002; Kasalica & Lamprecht, 2018; Kona, Bansal, Blake, & Gupta, 2008; Martin et al., 2005).

4.3 Workflow synthesis

Workflow synthesis is based on the concept of loose specification. A loose specification is "*a specification which features elements of under specifications"* (Bjørner, 2007). Under specifications allows the creator of a specification to omit certain details. This omission is often done because the element is unknown or to describe a larger class of possible elements. In programming, these type of loose specifications are used to defer the full specification to a later phase of the development. When we look at a workflow we can specify 4 types of elements: the begin state, the processes, the process state and the end state (figure 4.4).

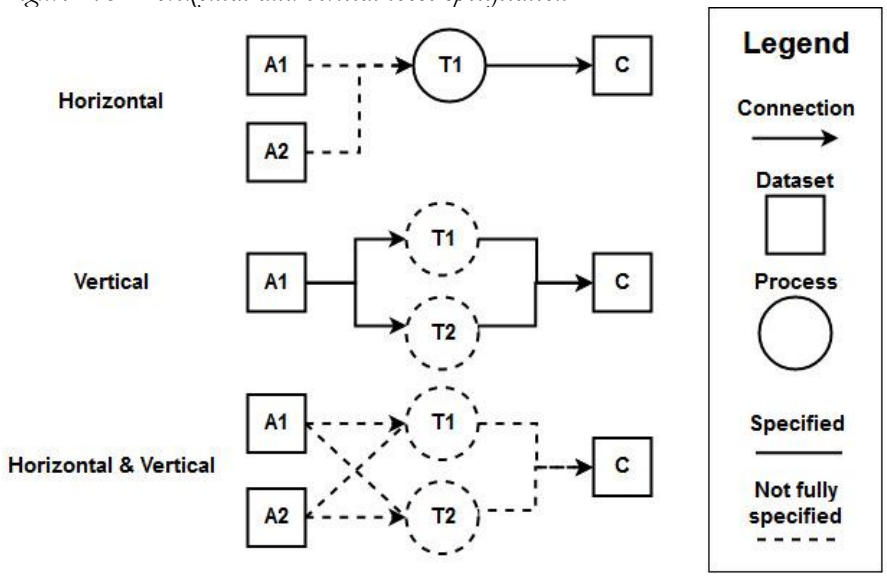
Figure 4. 4 Specification of a simple GIS workflow



The begin state is the specification of the initial state of the system. For GIS it describes the available datasets for workflows. The process is the specification of the implementation of a process in order to create a new system state. It specifies the input of the process, the output of the process and the process itself. The output is added to the new system state in the process state. The end state is simply the final process state.

When specifying a workflow it is possible to under specify any of these elements. Underspecification of workflows can be done along two dimensions. The horizontal dimension (figure 4.5) is the under specification of connections between states and processes. It allows for the creator of a workflow to not specify the input or output of a specific process. The vertical dimension (figure 4.5) is the under specification of processes and states, allowing for the usage of abstract descriptions in the specifications. This allows the creator of a workflow to loosely specify processes or data states. By combining horizontal and vertical under specification it is possible to specify a workflow with neither determined connections nor processes. (Lamprecht, Naujokat, Margaria, et al., 2010). The goal specification for workflow synthesis can be perceived as a fully underspecified workflow with only the end state and begin state known.

Figure 4. 5 Horizontal and vertical loose specification



Workflow synthesis is a method to automatically add elements that were underspecified to a workflow according to the semantic constraints (Lamprecht, Naujokat, Margaria, et al., 2010). In order to achieve this automatic addition of elements, workflow synthesis makes use of the synthesis methodology (Freitag, Steffen, Margaria, & Zukowski, 1995; Steffen et al., 1993). This methodology can handle loose specification along both the vertical and horizontal dimensions. The actual synthesis itself is based on the modal logical SLTL (Semantic Linear-Time Logic), which combines relative time with taxonomic classifications of data types and processes (Steffen et al., 1993). SLTL combines static, dynamic and temporal constraints. The static constraints are the taxonomic expression over the data types or classes that denote if the types of data match with the processes or classes of processes. The dynamic constraints are the taxonomic expressions over the processes or classes of the process taxonomy. The temporal constraints are covered by the modal structure of the logic. When all the necessary taxonomies are imported the synthesis algorithm interprets the synthesis problem specification and automatically generates all process compositions that satisfy the given specification in order to generate valid workflows for the underspecified workflow (Lamprecht, Naujokat, Margaria, et al., 2010).

In order for a synthesis to successfully bridge the gap between states, it requires a machine-readable formalization of the domain knowledge (figure 4.6). This formalization includes a machine-readable formalization of the domain's terminology in the form of an taxonomic classification, sometimes also referred to as a light ontology. It also includes a machine-readable description of the available processes and data. The data must be annotated in accordance with a formal description of data types in the ontology. These data types provide the basis for the inputs of processes. Finally in order for the synthesis to function it requires a machine-readable formalized description of processes that are annotated with inputs and outputs in accordance to the ontology (Kasalica & Lamprecht, 2018; Lamprecht et al., 2009).

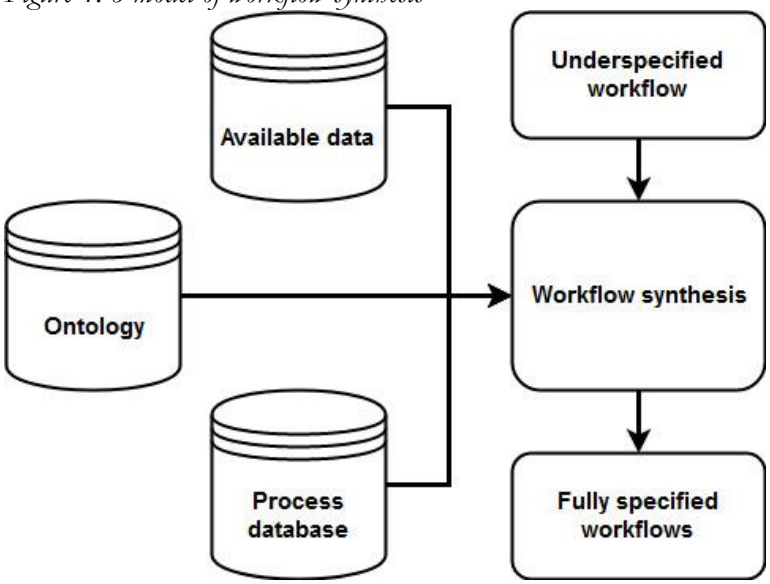
Synthesis programming can consider a problem locally or globally. When a problem is considered locally the synthesis can only take the output of the last process into account in order to formulate an appropriate process to connect to the upcoming process. However, if a synthesis problem is approached from a global perspective the synthesis engine can use all the resources within the system. As GIS workflows often require multiple inputs and sometimes require parallel processes in order to create the necessary resources for a workflow. As such, GIS synthesis should be approached as a global synthesis problem.

Using SLTL, synthesis can use these formalizations to automatically match data to possible processes. It keeps generating possible combinations in accordance with the restrictions

until a combination of data and processes is found that bridges the gap created by the underspecification. If only the begin and end state of a workflow are specified the workflow is underspecified in both a vertical and horizontal sense as no tools are selected and no connections are designated. When specifying the synthesis problem as a global problem the synthesis engine can create all possible workflows for the begin state that lead to the end state in accordance with the set limitations.

If the necessary elements of the formalized domain knowledge can be gathered for the field of GIS it should be possible to create a workflow support system based on synthesis that can create any available workflow for a set goal specification based on the available data and GIS tools. The advantage of such a system would be that the user of it would no longer require extensive expert knowledge in order to create a valid GIS workflow. The knowledge of all available functionalities and their applicability constraints would be captured within the formal framework. Secondly, such a system allows for better usability and scalability, as it would drastically reduce workflow creation time and the possibility for human error. The effectiveness of such a support system has already been demonstrated by its usage in the field of bioinformatics, allowing users to automatically validate the meaningfulness of a connection between functionalities, or suggest functionalities to bridge the gap between features (Lamprecht et al., 2009; Lamprecht, Naujokat, Steffen, & Margaria, 2010).

Figure 4. 6 model of workflow synthesis



4.3.1 GIS ontology

The basis for the formalization of domain knowledge lies within the creation Underspecification an ontology. The word ontology is used differently depending on what community you are in. In the field of information science, an ontology is used to formally represent knowledge (Zúñiga, 2001). This formalization of knowledge is based on a conceptualization. A conceptualization is made up of the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that they hold between them (Gensesereth & Nilsson, 1987). The conceptualization thus is an abstract, simplified view of the world. Due to the limitations of language every knowledge-based system is forced to use some form of conceptualization in order to make implicit knowledge explicit (Guarino, 1998). An ontology within the field of information science is "*an explicit specification of a conceptualization*" (Gruber, 1995).

Later the idea of an ontology for the field of information science was expanded upon by specifying more elements. Zúñiga (2001) Narrowed down the definition of Gruber by providing

a definition of specification and a definition of conceptualization. Zúñiga defined a specification as *"an axiomatic theory made explicit by means of a specific formal language. The ontology is designed for at least one specific and practical application. Consequently, it depicts the structure of a specific domain of objects, and it accounts for the intended meaning of a formal vocabulary or protocols that are employed by the agents of the domain under investigation"*. Conceptualization was defined as "*the universe of discourse at work in every possible state of affairs for the particular domain (or domain space) of objects that are targetted by the ontology"*. In the rest of the thesis, we shall use the definition of Gruber for its brevity but keep the additions of Zúñiga as an underlying clarification.

There are four types of ontologies that can be considered for the field of information sciences: top-level ontology, domain ontology, task ontology, and application ontology. The level of an ontology is based on its generality (Guarino, 1997).

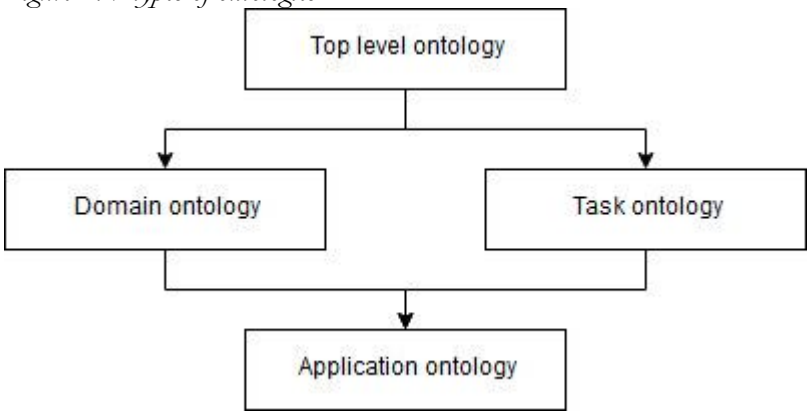
Top-level ontologies this type of ontology describes very general concepts like space, time, matter and objects. These type of ontologies are so broad that they supersede particular domains. Because of this, it is possible to create an unified-top level ontology as these types of ontologies don't conflict with domain-specific ontologies.

Domain and task ontologies These type of ontologies described the vocabulary related to a generic domain (GIS, or bioinformatics) or a generic task or activity like (drawing maps or counting antibodies). This is done by introducing specialized terms that are derived from the top-level ontologies.

Application ontologies These type of ontologies describe concepts depending both on a particular domain and task. These concepts often correspond to roles of entities within the domain within a certain activity within the domain. Examples of such an entity would be vector datasets.

These ontologies can be seen as derived from each other. Domain and task ontologies are derived from the top-level ontologies and application ontologies are derived from the domain and task ontologies (figure 4.7).

Figure 4. 7 types of ontologies



Within an information system ontologies can play two different roles. The ontology can provide the architecture for an information system, or can be used to drive the information system itself (Guarino, 1998). The ArcGIS desktop application is an excellent example of an information system designed according to an ontology. The tools and applications have been designed with the formal description of the field of GIS in mind, but it doesn't actively use the GIS ontology in its operations. In order for the workflow synthesis system to evaluate the validity of workflows for automatic workflow composition, the ontology must be actively checked. When the ontology

is actively used in the execution of an information system it is considered an application ontology. Unfortunately, there is a very limited set of readily available application ontologies available for reuse in information systems (Guarino, 1998).

4.3.2 Ontologies for GIS workflow synthesis

some successful work has been done in regards to the automatic chaining of geo-functionalities and geo-data based on semantic descriptions (Athanasis et al., 2009; Fitzner et al., 2011; Lemmens et al., 2006; Lutz, 2007; Yue et al., 2007). One of the essential elements of workflow synthesis for any domain is capturing the semantic intricacies of the terminology used within the domain in a formalized ontology. However, the ontologies used in these studies were specialized either in the discovery, comparison, composition or execution of operations and workflows. As of yet the field of GIS doesn't have a geo-analytical ontology that can be used to drive workflow synthesis (Hofer et al., 2017). This is due to the fact that map-making, as well as spatial analysis, are full of semantic intricacies, as creating workflows in GIS goes far beyond simply fitting data types to inputs and outputs (Hofer et al., 2017). In order to properly apply GIS applications, extensive knowledge of the semantics of the terminology of GIS is needed (S Scheider & Tomko, 2016; Stasch et al., 2014). Input and output parameters of an operation provided by the current descriptions are not sufficient for automated workflow composition (Yue et al., 2007). As of yet, it's not clear which semantics need to be contained in a GIS ontology to address the full spectrum of workflow synthesis for GIS (Hofer et al., 2017). The current method of discerning the validity of semantics is mainly based on theory and literature, using argumentation and/or formal logic to prove the validity of each of the semantics and their relations in the ontology.

4.3.3 ontology evaluation

There is a clear need to evaluate available ontologies for the development of ontologies themselves and for their use and re-use within semantic technologies. However, as it stands there is no global approach for evaluating ontologies. The available methods can be categorized into three district types of measures: structural measures, functional measures and usability-profiling (Gangemi et al., 2005).

Structural measures: this measure focuses on the syntax and representation of ontologies as graphs. This type of measure can identify topological, logical and meta-logical properties by context-free metrics. Examples of such measures would be the ontology depth, width or the type of logical relations used within the ontology.

Functional measures: This measure is related to the intended use of a given ontology, which in the case of an application ontology is the specification of a given conceptualization. the measures of evaluation are related to the quality of the practical implementation of the ontology.

Usability-profiling: this type of measure focusses on the ontology profile, which addresses the communication context of an ontology. The ontology profile consists of the metadata and annotation of the ontology and its elements. The metadata can consist of the authorship, price, license, etc.. Annotation contains information about structural, functional or user-oriented properties of an ontology and its elements.

For this thesis, the evaluation of the ontology will be approached from a functional perspective as we are mainly interested in the creation of a functional GIS workflow synthesis prototype, which requires an application ontology that can serve our intended function. In order to evaluate

the function of an ontology, it is necessary to find a way to measure the extent to which the ontology mirrors a specific expertise. This is also referred to as the matching problem. By measuring the extent to which the implemented ontology matches the expertise it is possible to calculate the functional measures such as recall, precision, specificity, and sensitivity. This is done by posing competency questions. A competency question is a question designed to be answered by both the implementation of the ontology and by an expert. If the implementation is capable of providing the same answer as the expert it is considered to be a valid ontology. The results of the implemented ontology in this theses are the synthesized workflows. By generating workflows in order to answer a spatial question it is possible to create a set of workflows that can be evaluated by a GIS expert. If the workflows are valid according to the expert the ontology can be considered valid from a functional perspective.

4.3.4 Workflow evaluation

The need to evaluate the quality of workflows is not without precedent. Business workflows have been defined in workflow models for quite some time, and in order not affect business objectives negatively these models needed proper definition, analysis, verification, and refinement before enacting the model (van der Aalst, Hirnschall, & Verbeek, 2002). However, there is only a very limited body of work in literature that covers workflow verification and evaluation. In the available literature, workflows are almost always approached as Petri nets. Petri net is a mathematical modeling language for the description of distributed systems based on the concepts of place, transitions, and tokens (Clempner, 2017; van der Aalst, 1998). One of the central pillars of this mathematical model is the concept that the execution of a process comes at the cost of its input. However, this doesn't match the type of workflows found in GIS workflows, where the data keeps on existing no matter how many times it has been used as input. Most of the criteria developed in this field deal with the validity of workflows in terms of resources and a such have no real applicability on GIS workflow evaluation. As such, there is no readily available body of literature for GIS workflow evaluation.

Following the lack of literature is the absence of a standardized framework to evaluate errors synthesized GIS workflows. As such the expert's evaluation of the synthesized workflows thus will be based on the tacit knowledge of the expert. This thesis shall aim to provide some constructs to evaluate workflows by categorizing the errors encountered in automatically generated workflows during the manual evaluation of these workflows. These constructs will form the basis for our own novel workflow evaluation framework. Such a framework could provide a valuable tool for future evaluation of workflows and provide creators of application ontologies for workflow synthesis a method to test the functional quality of their ontology.

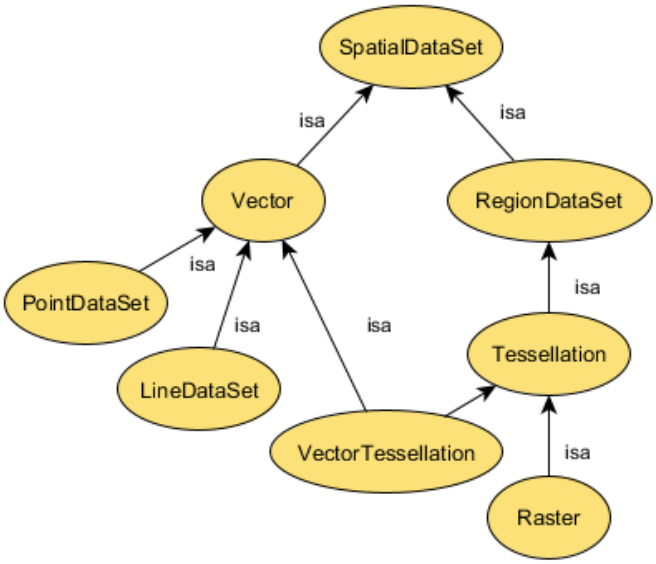
4.4 Semantic data type signatures otology

The ontology that will be implemented in order to create a GIS workflow synthesis prototype is the semantic data type ontology proposed by Simon Scheider (2019). His ontology was specifically designed with GIS workflow synthesis in mind. This chapter will describe the relevant semantics of his ontology and the relations between the semantics. The relations will be described using description logics for OWL. His approach was based on the idea that abstract data types allow for GIS operations to be constrained to the type of data that can be run meaningfully. In his paper, he tries to formulate some of the necessary semantics to capture the data types for geospatial analysis. He discusses the implications of each semantic for geospatial analysis and suggests a lightweight formalization in OWL based on the Description logic notation for OWL. The semantics for his research are based on geometric layer types, core concepts, and attribute semantics.

4.4.1.1 Geometric layer types

In his publication, Simon Scheider suggests that the first relevant semantic should be the geometric properties of layers (figure 4.8). Geometric properties are those properties that can be derived from the geometry of a solid body or particle. Layers within GIS can have distinct geometries that can be distinguished. Normally layers in GIS are distinguished according to the most prominent geodata types namely: raster or vector. In his publication, he argues that the distinction between raster and vector is extraneous, as there is trivial translation between raster to vector and vice versa, but also because often ignoring this distinction between raster and vector is important for geographic analysis. Instead, he mainly differentiates between geometric layer types based on them being a vector or a tessellation. He defines a tessellation as *a tilling of the plane into regions which are jointly covering the plane and mutually non-overlapping*. Tessellation enables GIS users to describe an area without any gaps in the plane and is thus a representation of the concept of a field. However, he does realize the relevance of raster datasets for the creation of workflow synthesis so maintains raster as a subtype of a tessellation. Since rasters adhere to all the requirements of tessellations but are just a special type of tessellations that is limited to a square tilling of the planes.

Figure 4. 8 Hierarchy of geodata types based on geometric properties of layers with OWL class descriptions



Definition 1 RegionDataSet ≡ SpatialDataSet∩∀HasElement.RegionData

A RegionDataSet is a spatial dataset with RegionData elements. In his model, Raster is a subclass of Tessellation, which is a subclass of RegionDataSet

Axiom 1 Raster ⊂ Tessellation ⊂ RegionDataSet

Definition 2 Vector ≡ SpatialDataset ∩ ¬ Raster

In order to keep the distinction of vector and raster as incompatible data formats, he defines anything that isn't a raster dataset as a vector dataset.

Definition 3 Linedataset ≡ Vector∩∀HasElements.LineData
Definition 4 Pointdataset ≡ Vector ∩∀HasElement.PointData

PointDataset and LineDataSet are special types of Vector data, being distinct from each other and from RegionDataSets.

Definition 5 VectorTessellation ≡ Vector ∩∀ HasElements.VectorTessellation

Next, he states that tessellations can also be created with vectors, but not by line or point datasets

Axiom 2. RegionDataset∩LineDataSet∩PointDataset⊆ ⊥

4.4.1.2 Spatial core concepts and attribute distinctions

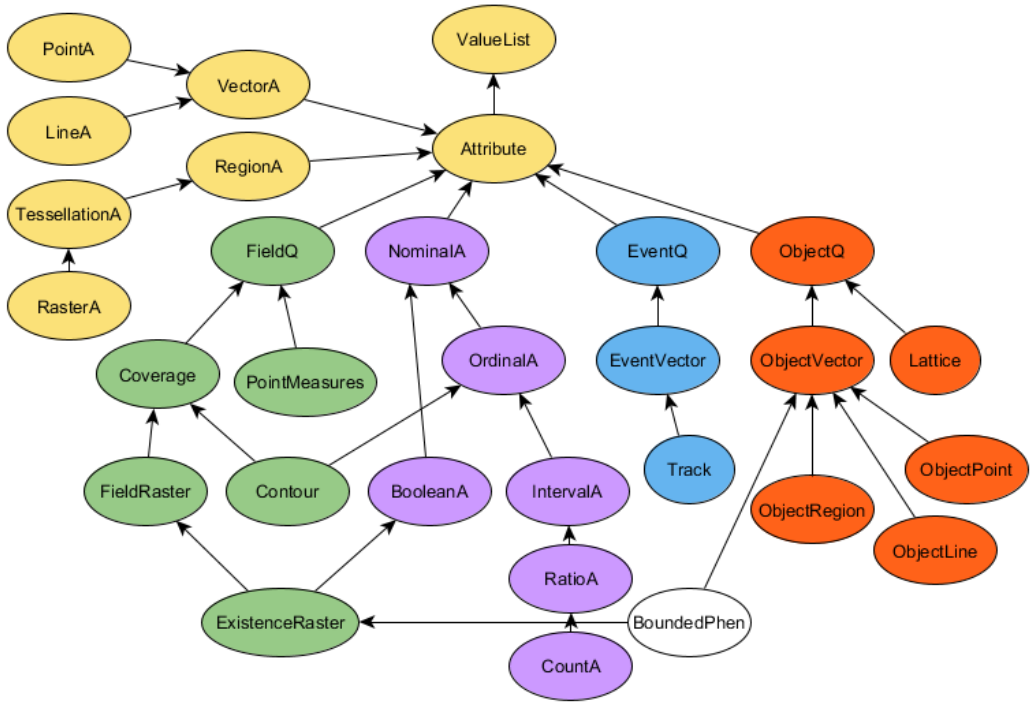
The other two elements that were critical for his creation of semantic data type signatures were spatial core concepts and attribute semantics (figure 4.9). The core concepts that are taken into account were introduced by Kuhn (Kuhn, 2012) as a conceptual and computational interface to GIS for multiple disciplines. Kuhn defined 10 core concepts of spatial information being: location, neighborhood, field, object, network, event, granularity, accuracy, meaning, and value. For the work of Scheider, the most relevant core concepts are field, object (table 4.2), network and event. For this thesis the semantics that are derived from the core concepts of event and network were not included as these concepts ad a lot of complexity and as such are outside the scope of this thesis.

table 4. 2 Core concepts for geography according to Kuhn (2012)

Core concept	Meaning
Field	Fields describe phenomena that have a scalar or vector attribute everywhere in the space of interest, for example, air temperature on the earth's surface.
Object	Objects describe individuals that have an identity as well as spatial, temporal, and thematic properties. An example of objects would be, houses in a street.

Spatial core concepts are about the Thematic contents of geodata. Field locations and object have qualities and each of the core concepts needs to be able to reflect these qualities. Therefore, he defined the core concepts as a type of attribute and not a type of geodata. The qualities of these core concepts have their own unique mathematic properties that they don't share with the core concept. By using a tessellated representation of a field, which he refers to as a coverage. This coverage has the special property that parts of its regions have the same attribute value as the entire region, this unique feature is called self-similarity and allows us to reconstruct the underlying values at any location within the field. Objects do not have such properties as their values are related to their entirety making them not self-similar. Another important thing to take into account that these type of core concepts need to be captured in values. This is done at different measurement scale levels and they determine the type of numerical operation that can be applied to the data.

Figure 4. 9 abstract types of spatial attributes, combing geometric layer types, spatial core concept, and statistical attribute types.



Definition 6 Attribute ≡ ValueList∩∃ofDataSet.SpatialDataset

Following his logic of core concepts and attribute distinctions being data types and not a type of geodata he defines attributes as a value list of some SpatialDataSet. Each of the core concepts represented in this ontology is its own class of attributes.

Axiom 3. FieldQ ⊆ Attribute; EventQ ⊆ Attribute; ObjectQ ⊆ Attribute

By representing core concepts as attributes instead of types of geodata you can combine them freely. This way the concept of field can be described by both point data and region data. Finally, each attribute also needs to be captured in a type of value. Levels of measurement for spatial attributes are a chain of sub-assumptions since lower measurements are special cases of higher levels. For example, a boolean is a special case of a (bi)nominal attribute. These levels of measurement determine the mathematical operations that can be executed in a meaningful way (Stevens, 1946).

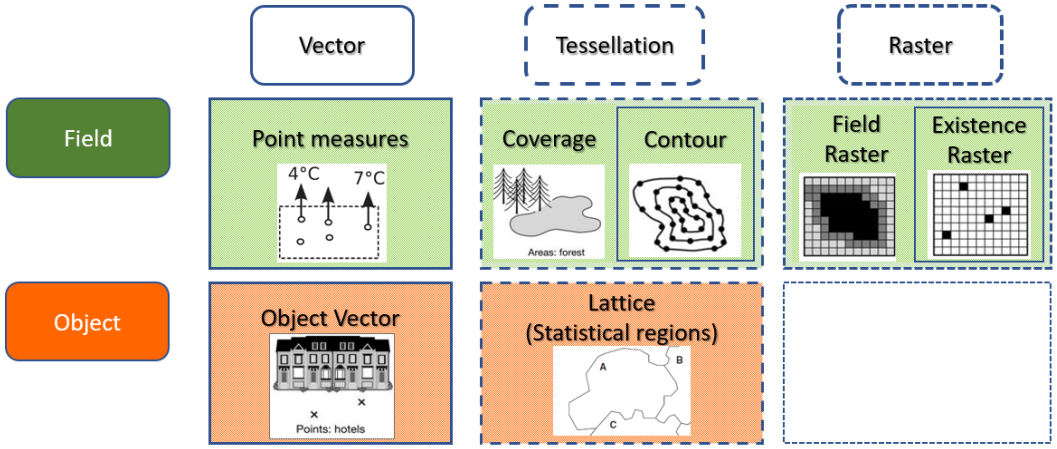
Axiom 4. IntervalA ⊆ OrdinalA ⊆ NominalA ⊆ Attribute
Axiom 5. BooleanA ⊆ NominalA

4.4.1.3 Synopsis of abstract data types

The distinctions above can be freely combined resulting in a total of 72 different data type signatures. Instead of describing them all Scheider creates 7 categories that are most relevant for geographic operations concerning fields and objects (figure 4.10). We shall shortly discuss each of these definitions and their associated properties as they will often be used within the annotation of both the tool and data database, and their properties are very relevant for

evaluating the validity of certain workflows.

Figure 4. 10 Core concept data types



Point measures ≡ FieldQ ∩ ∃OfDataSet.PointDataSet

According to Scheider the simplest way of representing a field is in terms of pointwise samples of measurements. He defines this data type as point measures. This type of data represents a field quality by using vector points. Since the underlying datasets is a vector dataset and not a tessellation this type of data cannot be directly used to estimate the value of the field quality for each location within the extent of the data.

FieldRaster ≡ FieldQ ∩ ∃OfDataSet.Raster

A field raster is a very common way of representing a field quality as it's a tessellation and as such can be used to defer the field quality of each location within the extent. As a field quality, it is inherently self-similar.

ExistenceRaster ≡ FieldRaster ∩ BooleanA

An existence raster is a raster that shows the existence of a kind of phenomena, which includes all core concepts for each location in space. The value "true" asserts the existence of the phenomenon at the cell locations within the raster.

Coverage ≡ FieldQ ∩ ∃OfDataSet.VectorTesselation

A coverage is a more general way of representing a field, by not following the strict tessellation of a raster but instead tessellating based on the values of the field quality. As Coverage are field qualities they are also self-similar.

Contour ⊆ Coverage ∩ OrdinalA

A contour is a type of coverage whose values are ordinally scaled, denoting quality intervals of continuous fields such as Height isoline maps.

ObjectVector ≡ ObjectQ ∩ ∃ ofDataSet.VectorTessellation

In contrast to field bases types, object-based attributes are not self-similar. In addition, the boundaries of their geometries become meaningful in the sense that they represent actual objects, instead of intersections of values such as the coverage. Objects are not tessellated and as such there can be "gaps" in the dataset where no objects and no quality is located. An object vector is an attribute of vector data which represent object qualities.

Tessellation≡ ObjectQ ∩ ∃ ofDataSet.VectorTessellation

Because the boundaries of an object are meaningful on their own, we can summarize data within the boundaries. An example of such a summary would be summarizing the average housing value within a municipality. An important distinction of this type of tessellation is that a lattice is not self-similar and as such the mean isn't the same for every location within the lattice.

5 Methodology

This chapter will outline the methodology for this research as well as describing the used technologies used within the research. The goal of this thesis is threefold, the first goal is the creation of a novel GIS workflow synthesis prototype that is capable of bridging the gap between data states and goal specifications as currently, no such workflow support system exists for GIS. The second goal of this thesis is the implementation of the workflow synthesis prototype in a limited set of scenarios in order to test the prototype in a realistic setting. The third and final goal of this thesis is the evaluation of the generated workflows in order to evaluate the synthesis prototype and the functional dimension of the full ontology and the limited ontology.

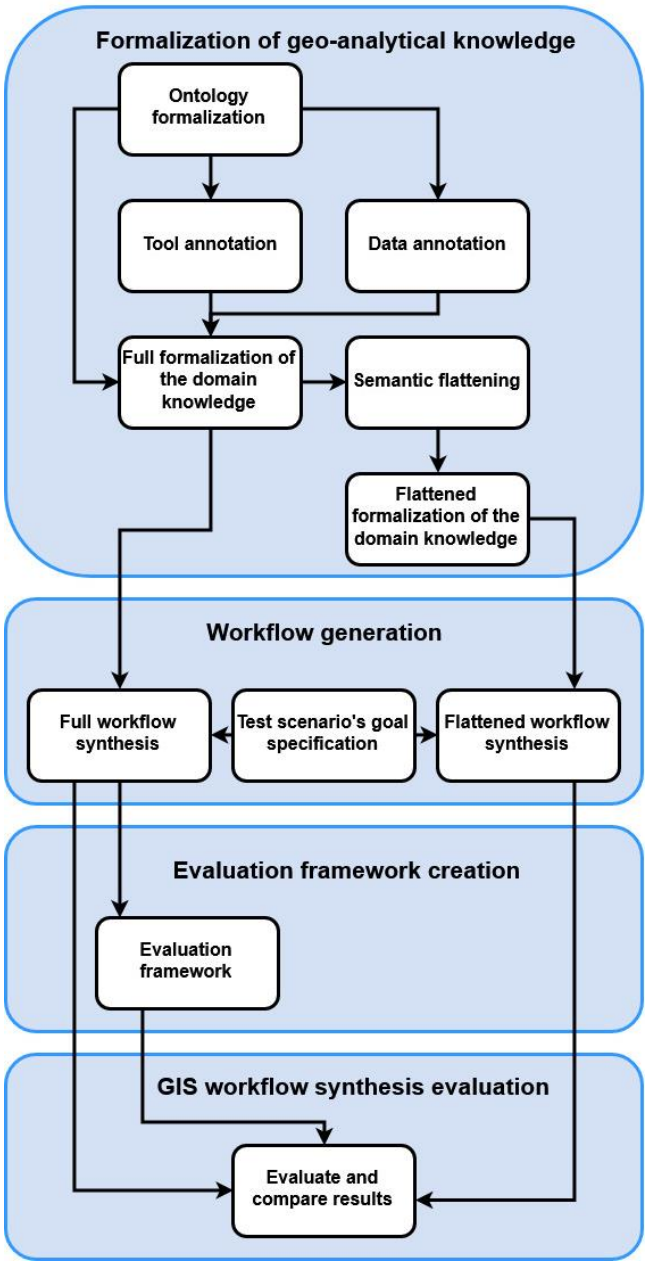
In order to achieve these goals, the methodological steps specified in figure 5.1 will be undertaken. All the steps have been divided into stages that refer to a specific research category. The first phase of the research is the formalization of the geo-analytical knowledge. The goal of this phase is to create a limited formalization of the domain of GIS in order to facilitate the workflow synthesis in stage two.

The second stage is the creation of a set of workflows by combining goals specifications with the domain formalization in a workflow synthesis engine. The workflows will be created in accordance with the goal specifications that aim to emulate realistic GIS research.

In the third stage, the workflows generated in stage two will be used to identify error types in workflow generation. Based on these error types a workflow evaluation framework will be created that can identify errors and describe the validity of sets of workflows.

In the final stage, the validity of each respective set of workflows is compared in order to compare the validity of both ontologies and prove the relevance of the data type signatures ontology. It also serves as a test and to prove the viability of workflow synthesis for GIS.

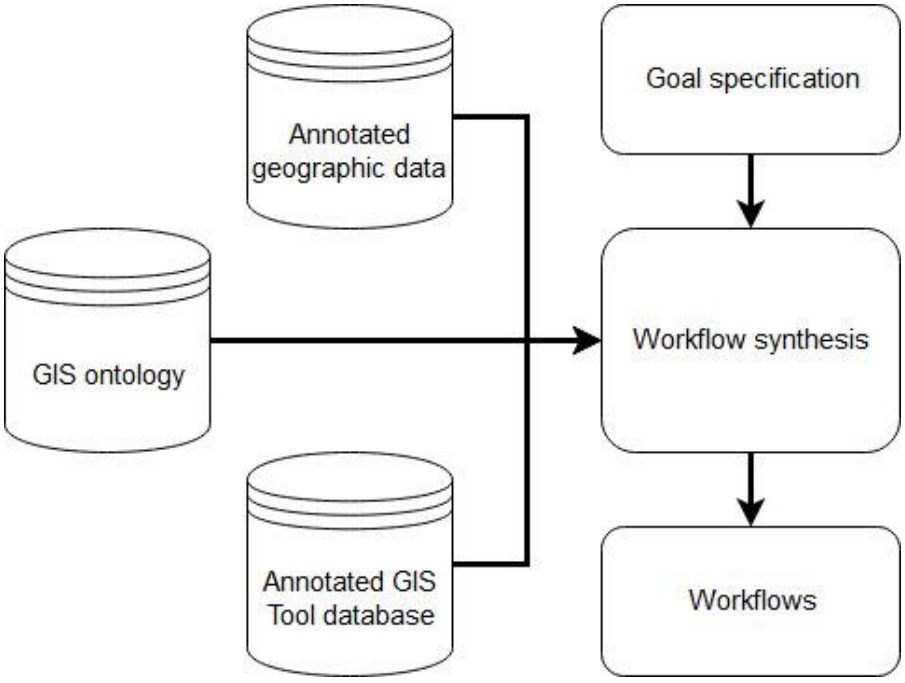
Figure 5. 1 methodology overview



5.1 Formalization of geo-analytical knowledge

In order for a synthesis to successfully generate workflows, it requires a machine-readable formalization of the geo-analytical knowledge (figure 5.2). This formalization includes a machine-readable formalization of the domain's terminology in the form of an taxonomic classification, sometimes also referred to as a light ontology. It also includes a machine-readable description of the available processes and data. The data must be annotated in accordance with a formal description of data types in the ontology (Gruber, 1995). These data types provide the basis for the inputs of processes. Finally in order for the synthesis to function it requires a machine-readable formalized description of processes that are annotated with inputs and outputs in accordance to the ontology (Kasalica & Lamprecht, 2018; Lamprecht et al., 2009). For the purpose of this study, we shall implement the ontology of Simon Scheider (Simon Scheider, 2019) in order to create a formalization of the geo-analytical knowledge, as this ontology was specifically designed for the automation of GIS workflow composition.

Figure 5. 2 Model of workflow synthesis for GIS workflows



5.1.1 Ontology formalization

The first step of formalizing geo-analytical knowledge is done by defining the classes and the relations between those classes based on the semantics proposed in the work of Simon Scheider (2019) in a machine-readable fashion. There are multiple ontology development and query languages, and more are currently being developed. Most of these formal languages are based on the eXtensible Markup Language XML. The most notable examples of these languages are Resource Description Framework (RDF), DARPA Agent Markup Language (DAML) and the Ontology Web Language (OWL). The most important elements to evaluate their suitability as a formal language for the creation of an ontology is their tool support, their expressive power, and reasoning support. DAML has a very limited suite of support tools and RDF doesn't have enough expressive power for this type of ontology creation (Munir & Sheraz Anjum, 2018), thus this thesis will formalize the ontology using OWL.

OWL is a semantic web language designed to represent the rich and complex knowledge

about things, groups of things and relations between things that are necessary to create an ontology. OWL is a computational logic-based language. This type of language can be used by computers to verify the consistency of knowledge or to make implicit knowledge explicit with reasoners. The latest and most expressive version of OWL is OWL 2 that was published in 2009. One of the main features of this language is its possibility to create unique classes in contrast to RDF where each resource is simply an instance of the class rdf:resource. OWL allows for the creation of other classes through the use of owl:class.

A class has some unique properties compared to a resource that are essential for the creation of a structured formalization as it allows the user of OWL to describe the relation of an entire class of instances in connection to another class of instances (Table 5.1). For example the predicate rdfs:subClassOf states a relation where all instances of the subject class are also part of the object class. This type of relationship would allow for the automatic referral of a more specific class to a broader class. When you describe a specific object of the terminology as a class you can then describe the structure of these classes within the domain of GIS using the relational predicates.

table 5. 1 listing of owl predicates enabling class axioms

Owl	*Description*
rdfs:subClassOF	allows one to say that the class extension of a class description is a subset of the class extension of another class description.
owl:equivalentClass	allows one to say that a class description has exactly the same class extension as another class description.
owl:disjointWith	allows one to say that the class extension of a class description has no members in common with the class extension of another class description.

The type of structures that can be described in OWL aren't limited to class axioms. Other types of structures include the possibility to set restraints on the properties of instances of a certain class or setting relations of certain properties to other properties and many more. This thesis won't go into detail as to what all the possibilities of OWL are in terms of structuring classes as this out of the scope of this thesis

In order to build the ontology, creators of ontologies often turn towards ontology tools. An essential piece of this toolset is the ontology builder. For the purpose of creating ontologies in the field of geospatial applications, there is a clear preference for Protégé over all other tools (Albrecht, Derman, & Ramasubramanian, 2008b). Protégé is an open-source ontology editor and a knowledge management system. Just like many workflow management systems protégé provides a graphic user interface to define ontologies. The defined ontologies can be validated in terms of consistency within protégé by using deductive classifiers. These same deductive classifiers can also infer new information based on the structure of an ontology. An added advantage is that protégé is free and has many extensions for OWL, multiple API, multiple export formats and several editing tools. Even though protégé might be too complex for domain experts (Luscher, Burghard, & Weibel, 2007) it provides a good tradeoff between complexity and capabilities meaning that while it's not the most extensive ontology builder it provides users with many capabilities for a reasonable amount of learning (Albrecht et al., 2008b). As such this thesis shall formalize the ontology using Protégé. The created ontology shall be far from sufficient for perfect semantic-based automatic workflow composition. But the goals of this thesis is not to provide a fully functioning workflow synthesis system but just a prototype to demonstrate the possibility of workflow synthesis for GIS and to evaluate the capacity of different ontologies to answer spatial questions.

5.1.2 Data and tool annotation

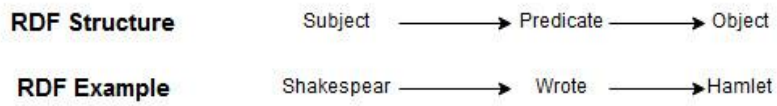
The second and third step of formalizing the domain's knowledge is the creation of a GIS tool database and a geographic data database that both have been annotated in accordance with the ontology (figure 5.3). In order for the synthesis engine to interpret these databases, they must be formulated in a machine-readable method.

The method used in the thesis for the formalization of the databases is based on the semantic web. The semantic web was an addition created for the world wide web as the WWW wasn't expressive enough to share data in a structured fashion (Bizer, Heath, & Berners-Lee, 2009), resulting in a loss of meaning when the data was moved from its original context. By using linked data it was possible to give information well-defined meaning, enabling computers and people to work in cooperation (Berners-Lee & Hendler, 2001). *Linked data is data published on the web in such a way that it is machine-readable, it's meaning is explicitly defined, it is linked to other external datasets and can, in turn, be linked to or from external datasets* (Bizer et al., 2009).

5.1.2.1 Resource description framework

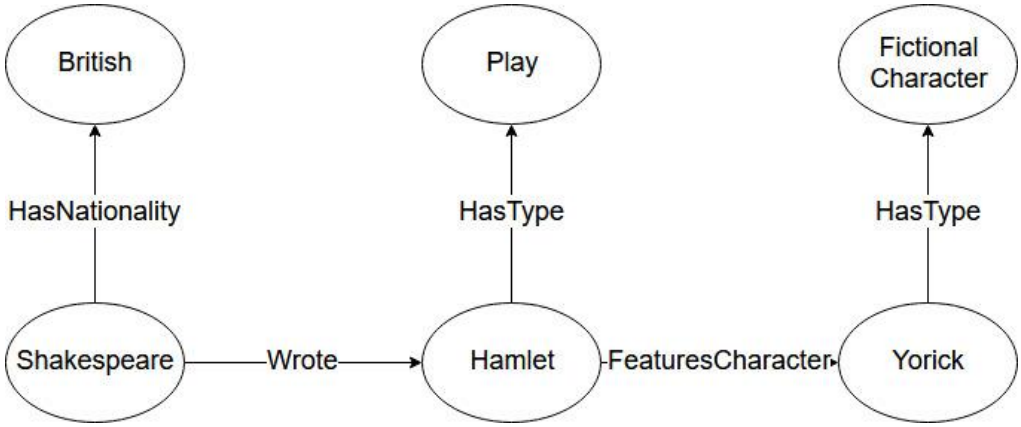
Linked can be structured according to RDF (Resource Description Framework). RDF is a data model that represents a semantic network. RDF is made up of a directed network with qualified edges with which to structure and link data that describes things in the world. A qualified edge specifies a specific relation with a certain direction. These qualified edges allow RDF to describe a network of relations that are heterogeneous in nature, as every edge can have a different meaning (Bizer et al., 2009). Each thing in RDF is specified by a Unique Resource Identifier (URI). the semantic web uses URIs as names for things instead of the names themselves. This is done in order to remove ambiguity. So instead of referring to Shakespeare RDF refers to the URI *http://www.workingontologist.org/examples/chapter3/Shakespeare*. The first part of the URI describes the domain or as it is also known in RDF the namespace in this case http://wwwk.workingontologist.org. The namespace often describes the owner or publisher of the dataset. The second part of the URI is the identifier and indicates a unique resource. To create the directed network RDF uses so-called subject – predicate – object triples. The subject and object of a triple are both Unique Resource Identifiers (URI) that each identifies a resource. The predicate specifies how the subject and object are related and is also represented by a URI. For example, an RDF triple can state that Shakespeare - wrote – Hamlet (Figure 5.3).

Figure 5. 3 Shakespeare RDF triple example



Triple's aren't self-contained denotations of information and relations. Instead, triples can be combines to become a graph like information structure. When the initial RDF triple in diagram 3.1 is combined with another triple such as "Hamlet - FeaturesCharacter – Yorick", the RDF becomes a graph-like structure (figure 5.4) where we can discover new things by following the graph. When you start with Shakespeare in figure 5.2 you can discover that Shakespeare wrote Hamlet, next you can discover that Hamlet is a play and features a character named Yorick. In the example of figure 5.4, multiple other triples have been added to further illustrate the graph-like structure.

Figure 5. 4 Example of RDF made up of multiple triples.



In order to store linked data, it needs to be serialized. There are multiple serialization formats for RDF. The original serialization format is RDF/XML. This method of serialization follows the same structure as the triple and is made up of a resource, a property, and a property value. The resource is anything that can have a URI such as *Shakespeare*, the property is a resource that has a name such as *wrote* and the property value is the value of a property such as *hamlet*. The property value can be another URI but it can also be a numerical value or a string. The resource, property and property value form a statement also known as the subject, predicate, and object of a statement.

This method of serialization has a strict structure, the structure shall be described according to the example in figure 3.5. The first line of the RDF is the XML declaration followed by the root element of RDF documents <rdf:RDF>. These root elements allow for a quicker description of the URIs domain, allowing for a shorter formalization. In the example xmlns:rdf specifies that elements with the rdf: prefix are from the namespace "xmlns:rdf="http:www.w3.org/1999/02/22-rdf-syntax-ns#". The <rdf:description> element contains the description of the resource identified by the rdf:about attribute. The indented elements below this such as <wo:wrote> are the properties and the URI that follows the rdf:resource is the property value. When a resource has been fully described the description is ended by </rdf:description>. When all the necessary triples have been described the document is closed by a final </rdf:RDF> declaration.

Figure 5. 5 an example of RDF/XML serialization

```
<?xml verssion="1.0"?>

<rdf:RDF
    xmlns:rdf="http:www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:wo="http://www.WorkingOntyologist.org/Examples/Chapter3/"
    xmlns:tp="https://www.w3.org/2012/09/odrl/semantic/draft/doco/"
>
    <rdf:Description rdf:about="http://www.workingontologist.org/examples/chapter3/Shakespeare">
        <wo:wrote rdf:resource="http://www.workingontologist.org/examples/chapter3/Hamlet">
        <wo:HasNationality rdf:resource="http://www.workingontologist.org/examples/chapter3/British
    </rdf:Description>
    <rdf:Description> rdf:about="http://www.workingontologist.org/examples/chapter3/Hamlet">
        <tp:HastType rdf:resource="http://mappings.dbpedia.org/index.php/OntologyProperty:Play"
        <wo:Features rdf:resource="http://www.workingontologist.org/examples/chapter3/Yorick"
    </rdf:Description>
</rdf:RDF>
```

This strict structure makes the triples machine-readable, however, it comes at the cost of legibility and writability. Especially if you compare the legibility of the graph with shortened URIs to the RDF/XML formalization. An alternative to the RDF/XML serialization of RDF is

Turtle. This method of serialization provides a trade-off between ease of writing, ease of parsing and readability. As most of the RDF serialization in this thesis will be written by hand the turtle will be used throughout the rest of this thesis (Figure 5.6). The turtle serialization doesn't start with an XML declaration but instead starts defining root elements with @prefix, this allows the usage of prefixes for often used namespaces. The other improvement in terms of legibility and writability is the fact that for turtle serialization you no longer need to specify the resource using rdf:about, instead any URI that is on the first row of the indentation is considered a resource with its own description. The second improvement is the fact that the resource is directly followed by the property and turtle doesn't require the specification of the property value with rdf:resource. Any extra property, property value combinations can be added to the resource by simply placing them indented beneath the first triple. The improved writability of this serialization reduces the chance of error and the speed of manual serialization, however certain programs don't accept turtle serialization and require a translation to RDF/XML.

Figure 5. 6 an example of turtle serialization

```
@prefix rdf: <http:www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix wo: <http://www.WorkingOntyologist.org/Examples/Chapter3/>
@prefix tp: <https://www.w3.org/2012/09/odrl/semantic/draft/doco/>

<http://www.workingontologist.org/examples/chapter3/Shakespeare> wo:wrote wo:Hamlet
    wo:HasNationality wo:British


<http://www.workingontologist.org/examples/chapter3/Hamlet> a <http://mappings.dbpedia.org/index.php/OntologyProperty:Play>
    wo:Features wo:Yorick
```

5.1.2.2 Annotated database creation

For the purpose of this study, we will need a set of geographic data to annotate in order to run the workflow synthesis. The data and tools will not be randomly selected but instead, be selected based on their representation of the seven distinct categories of abstract data types formulated in the work of Simon Scheider (2019). The data for this study will mainly be collected from the geoportal or the city of Amsterdam, but if certain data types cannot be found on that specific portal other geo-portals may be used as long as the data is relevant to the city of Amsterdam. The data gathered from these portals will be annotated in accordance with the principles of RDF. An example of such an annotation is presented in figure 5.7. This is an annotated dataset of the sports facilities in the city of Amsterdam. It denotes the fact that the dataset has the geometric properties of an object point datasets, that the values included in the points are nominal and that those values are intensive. Each dataset is also accompanied with a comment for easy identification of the specific dataset.

Figure 5. 7 an example of an annotated GIS dataset

```
<https://maps.amsterdam.nl/sport/> a ccd:ObjectPoint.
<https://maps.amsterdam.nl/sport/> a ccd:NominalA.
<https://maps.amsterdam.nl/sport/> a exm:IRA.
<https://maps.amsterdam.nl/sport/> rdfs:comment "this point dataset
```

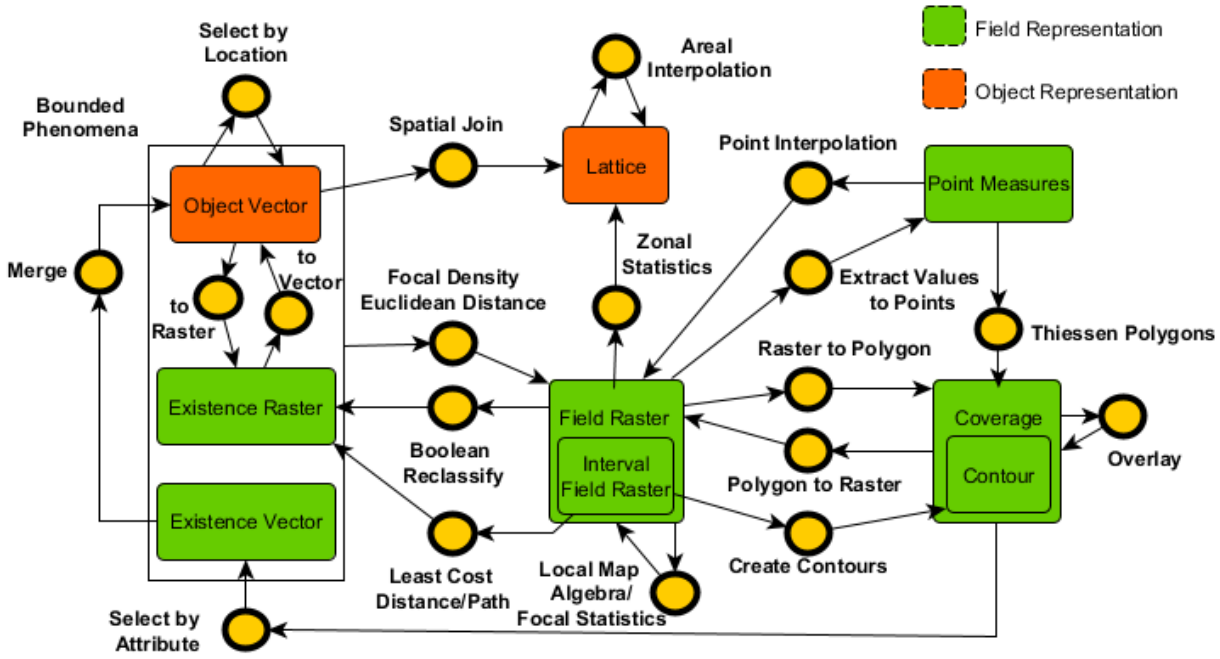
5.1.2.3 Annotated tool database creation

For the purpose of this study, we will also need a set of GIS tools in order to bridge the gap between data states. These tools will not be randomly selected but instead, be selected based on their representation of transformations between the core concepts of abstract data types (Figure 4.3). For this study, we shall at least select one tool for each possible transition. This is done to guarantee a more representative set of tools and to maximize the possible range of interaction between data types. The instances of tools in this database need to be annotated with a

description of valid input and their respective output that can be interpreted by the workflow synthesis engine.

The annotation is done by considering the implications of each possible combination of semantics for the validity of each tool. For instance, if a tool can only be used with vector datasets containing data at a ratio measurement level the input is annotated as being simultaneously a vector dataset and a ratio dataset. Not only will certain tools be limited in the geometric data type that they can accept, but their outcome will also change depending on the input. For instance, certain tools that only change the geometry of a vector dataset can't change the measurement level, as such their output depends on the measurement level of the input vector dataset. In this case, we can't construct a singular output for the tool as this would sometimes result in incorrect measurement levels for the output. In order to make sure the correct measurement level is denoted in the output, the tool needs to be split into different sub-tools that only accept a subset of the valid input that result in the same semantic data type output. The evaluation of all possible semantic combinations for each tool is done manually which makes it a time-consuming process.

Figure 5. 8 Examples of possible tools that facilitate transformations between the core concepts of abstract data types



For the purpose of this study, the tool selection will be based on the available tools in ArcGIS Pro. This is because ArcGIS pro offers a rich description of each tool on its website and has a broad range of tools that can cover a large range of possible operations within the field of GIS. The tools will be annotated in accordance with the principles of RDF (figure 5.9). The annotation will be done in accordance with a strict structure. Each annotation starts with The URI of the tool being annotated; the URI will refer to the description of the tool on the ESRI ArcGIS Pro website. Next, the predicate tools:implements specifies what exact version of the tool is being implemented with this annotation. This is necessary as tools can have different input, output relations based on the type of input and the parameters of the tool. The predicate rdfs:label gives a small description of the tool in order to make the tool repository more understandable for humans. Next, a unique blank node is specified for the input and output of the particular tool. A blank node is an unspecified resource that matches the set description. By

leaving this node blank any dataset that matches the description can be matched to this tool. Note that a tool can have multiple inputs and each will have to be specified accordingly. The blank nodes are annotated with

Figure 5.9 an example of an annotated gis tool in turtle format

```
######### Spatial join with sum rule
<http://pro.arcgis.com/en/pro-app/tool-reference/analysis/spatial-join.htm>
tools:SpatialJoinSumTessRatio rdfs:label "Sums the attributes at ratio measu

tools:SpatialJoinSumTessRatio wf:input1 _:SpatialJoinSumTessRatio1.
_:SpatialJoinSumTessRatio1 a ccd:ObjectVector. #"Join Features".
_:SpatialJoinSumTessRatio1 a ccd:RatioA.
_:SpatialJoinSumTessRatio1 a exm:ERA.

tools:SpatialJoinSumTessRatio wf:input2 _:SpatialJoinSumTessRatio2.
_:SpatialJoinSumTessRatio2 a ccd:VectorA. #"Target Features"
_:SpatialJoinSumTessRatio2 a ccd:TessellationA.

tools:SpatialJoinSumTessRatio wf:output _:SpatialJoinSumTessRatioOut.
_:SpatialJoinSumTessRatioOut a ccd:Lattice.
_:SpatialJoinSumTessRatioOut a ccd:RatioA.
_:SpatialJoinSumTessRatioOut a exm:ERA.
```

the abstract data types that are valid inputs for the tool and the output is annotated with the resulting abstract data types. This allows the synthesis engine to match any data to the input of the tools and create output accordingly. The tools included in this thesis are listed in table 5.1.

table 5. 2 Overview of the formalized GIS tools.

On	Operation	Operation subtypes	Input type	2nd input type	Output type
Field	Local map algebra		FieldRaster	FieldRaster	FieldRaster
	Focal statistics	Mean, Median, Sum, Variety	FieldRaster		FieldRaster
	zonal statistics	Majority, Sum Mean, Median,	FieldRaster	VectorTessellation	Lattice
	Boolean reclassify		FieldRaster		ExistenceRaster
	Select by attribute		Coverage		ExistenceRaster
	Raster to coverage		Nominal FieldRaster		Coverage
	Raster to contour		Interval FieldRaster		Contour
	Coverage to raster		Coverage		Nominal FieldRaster
	Contour to raster		Contour		Ordinal FieldRaster
	Overlay		Coverage	Coverage	Coverage
	Point interpolation		Interval PointMeasures		FieldRaster
	Extract values to point		FieldRaster	Point	PointMeasures
	Thiessen polygons		PointMeasures		Coverage
	Least cost distance/path		Interval FieldRaster	Bounded Phenomenon	ExistenceRaster
Object	Areal interpolation		Lattice		Lattice
	Select by Location		ObjectVector		ObjectVector
	Spatial join	Mean, Sum	ObjectVector	VectorTessellation	Lattice
	Raster to vector		ExistenceRaster		ObjectVector
	Feature to raster		ObjectVector		ExistenceRaster
	Merge		ExistenceVector		ObjectVector
	Euclidean distance	Vector, Raster	Bounded Phen		Ratio FieldRaster
	Focal density	Vector, Raster	Bounded Phen		Ratio FieldRaster

The workflow synthesis engine used in this thesis cannot directly read RDF and as such, the turtle notation needs to be translated to an XML notation. This is done by a python script that automatically translated the turtle to the appropriate XML format (figure 5.9, python script in the appendix).

Figure 5. 9 An example of an annotated tool in XML format.

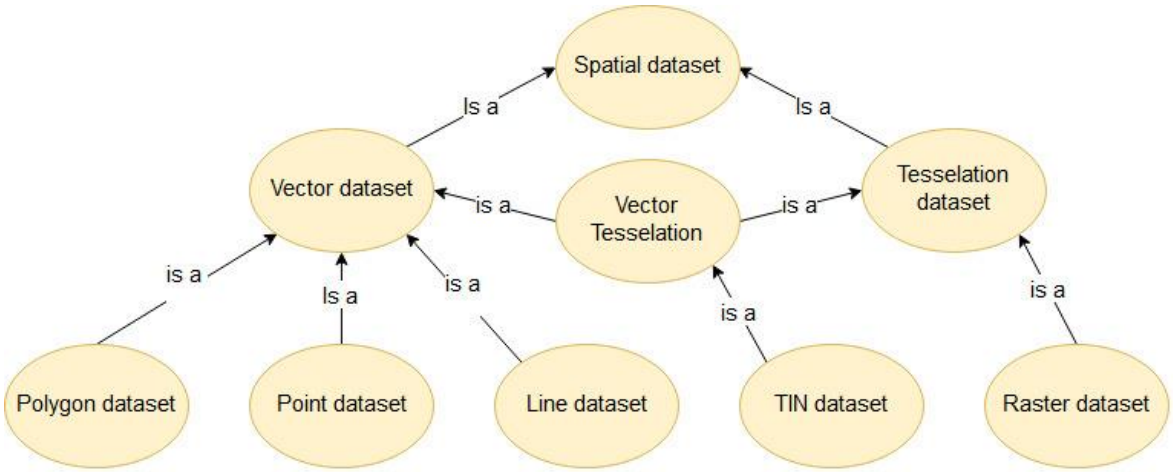
```
<function name="SpatialJoinSumTessCount">
  <operation>SpatialJoinSumTessCount</operation>
  <inputs>
    <input>
      <type>CountAObjectVectorERA</type>
    </input>
    <input>
      <type>TessellationAVectorA</type>
    </input>
  </inputs>
  <outputs>
    <output>
      <type>CountALatticeERA</type>
    </output>
  </outputs>
  <implementation>
    <code>SpatialJoinSumTessCount</code>
  </implementation>
</function>
```

A clear difference in the XML format and the RDF format is the fact that the XML format can only support a single data type as input. But due to the nature of our RDF description, each unique combination of properties can be considered a unique data type. In order to work around this, the python scripts also generated a unique data type for each included combination of data types. These data types can be perceived as unique leaves on a taxonomic tree.

5.1.2.4 Semantically flattening

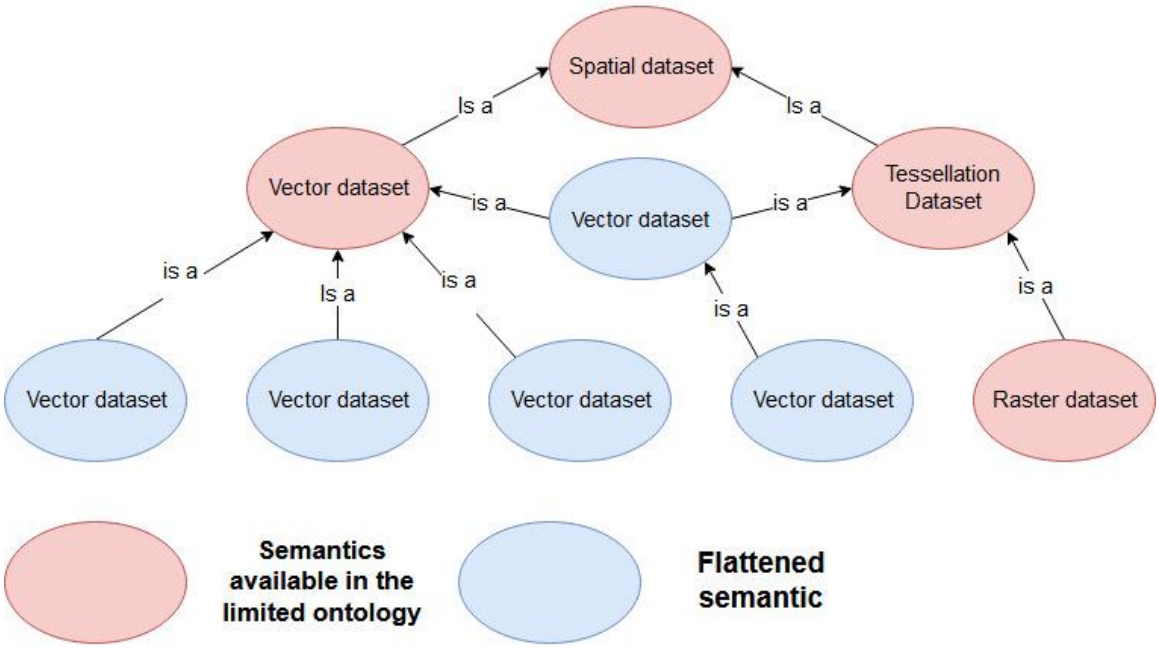
With the ontology, data repository and tool repository in place, it is possible to synthesize workflows based on the created formalization. However, the goal of this thesis was to evaluate the capacity of certain ontologies to answer spatial questions in GIS. In order to test such a thing, it is necessary to compare the capacity of Simon Scheider's ontology with a more limited ontology. This is done by generating and evaluating workflows for that ontology and comparing the validity of the results with a set of workflows that was based on a formalization that only takes data types into account that are used in ArcGIS Pro. In order to create this limited formalization all semantics that aren't included in the limited ontology need to be flattened to their least common subsumer that still exists in the limited ontology. The least common subsumer is the most specific common ancestor of two concepts found in a given ontology. In the ontology, in figure 5.10 the most common ancestor of polygon data and point data is vector data.

Figure 5. 10 pre-flattening geometric ontology



By matching every semantic to their least common subsumer that is still present in the limited ontology we can flatten the ontology and tool/data annotations to a more limited ontology (figure 5.11). in figure 5.11 you can see that the datasets that previously were the polygon and point dataset have been matched to their most specific subsumer and been changed to that specific subsumer. In this study the flattening of the ontology and tool description was done by running the ontology, data repository and tool repository through a Python script that "flattens" the ontology of each of these databases by replacing all annotations with their least common subsumer. Doing so it was possible to create an ontology, data repository and tool repository that only takes the semantics of line, point, vector, raster and tessellation into account as the example in figure 5.11.

Figure 5. 11 post-flattening geometric ontology



5.2 Synthesis engine

The goal of this study was to create a GIS workflow synthesis prototype. In order to create a GIS workflow synthesis prototype the formalized ontology, data repository and tool repository need to be combined with a synthesis engine. For this thesis, a synthesis engine by the name of APE (Automatic Pipeline Engine) is used. This workflow synthesis engine was developed for the research of Vedran Kasalica and Anna-Lena Lamprecht (Kasalica & Lamprecht, 2018). This synthesis engine is based on the Semantic Linear-Time Logic (STL) and was developed specifically for the creation of GIS workflows. The current state of APE doesn't offer a graphic user interface and requires the user to call upon the functionalities through java. The engine does offer certain advanced options such as the restriction on certain workflow patterns (table 5.3).

table 5. 3 possible workflow patterns for APE

Category	**Description**	**No. of parameters**
Force use	Use *"**tool**"* in the solution	1
	Use ***"data type"*** in the solution	1
	If *"**tool**"* is used then use *"**tool**"* subsequently	2
	If *"**tool**"* is used then use ***"data type"*** subsequently	2
	If ***"data type"*** is generated ***"data type"*** must be generated subsequently	2
Limit use	Do not use *"**tool**"* in the solution	1
	Do not use ***"data type"*** in the solution	2
	If ***"tool"*** is used then ***"data type"*** must have been previously generated	2
	If ***"tool"*** is used then ***"tool"*** must have been used previously	2
	If ***"tool"*** is used then do no use ***"tool"*** subsequently	2
	Use ***"tool"*** as the last tool in the sequence	1
	Generate ***"data type"*** as last data in the sequence	1
	If ***"data type"*** is in the system do not generate ***"data type"*** subsequently.	2

However, for this study a bare minimum of workflow patterns was used as any pattern entered should be applied universally for each synthesis problem. The only pattern that matched this description is the limitation that any workflow is only allowed to generate the goal data type once and that the goal data state is the final data state.

5.3 Workflow generation

The second goal of this thesis is the implementation of the workflow synthesis prototype in a limited set of scenarios. These scenarios will serve as the competency questions in order to test the prototype in a realistic setting and to provide the necessary workflows for the evaluation of the prototype and ontologies. In order to achieve this goal, this study will create workflows for a fictional GIS project for both the full and limited prototype. The fictional GIS project will be based on an assignment set out by the University of Amsterdam, using openly available data from the city of Amsterdam at "https://maps.amsterdam.nl/?LANG=nl". The original task set out by the University of Amsterdam was to create a livability atlas for neighborhoods at the level of Postcode Area (PC4) for older people in Amsterdam. In order to create such an atlas, this

thesis shall synthesize 5 competency questions. The workflow synthesis is implemented by fully underspecifying the necessary workflow in both the vertical and horizontal dimension by only stating the begin state and the goal specification. The begin state is created by specifying the semantic properties of the datasets that are initially added to the system. The goal specification is the semantic description of the final dataset that should be capable of answering the spatial question posed. The datasets are annotated to the fullest extent of their respective ontology. For each competency question, 200 workflows will be generated with a maximum length of 4 tools. Of these 200 workflows, the shortest 20 workflows will be selected for evaluation. The choice for the usage of the 20 shortest workflows instead of a random selection was based on the notion that longer workflows allow for more errors and that users of workflow synthesis will prefer shorter workflows. As such the shortest workflows are most relevant for the functionality of the prototype.

5.3.1 Competency questions

1. **What is the number of sports facilities in each PC4 area?**

   begin state: for this question will consist of a point dataset that denotes the location of each sports facility within Amsterdam and a dataset that contains the polygons of each PC4 area in Amsterdam.

   Goal specification: The goal for this question is a summary of the number of points within each pc4 area.

Begin state:	NominalA∩ObjectPoint, TessellationA∩VectorA
Goal specification	CountA∩Lattice
Flat begin state:	Point, Vector
Flat goal:	Tessellation

2. **What is the distance to the nearest voting office for the area of Amsterdam?**

   begin state: for this question will consist of a point dataset that denotes the location of each voting office within Amsterdam

   Goal specification: The goal for this question is a Euclidean distance raster that denotes the distance from the closest voting office.

Begin state:	NominalA∩ObjectPoint
Goal specification	RatioA∩FieldRaster
Flat begin state:	Point
Flat goal:	Raster

3. **What is the average distance to the nearest sports facility within each PC4 area?**

   begin state: for this question will consist of a point dataset that denotes the location of each sports facility within Amsterdam and a dataset that contains the polygons of each PC4 area in Amsterdam.

Goal specification: The goal for this summary of the average Euclidean distance to a sporting facility within each PC4 Area.

Begin state:	NominalA∩ObjectPoint, TessellationA∩VectorA
Goal specification	RatioA∩Lattice
Flat begin state:	Point, Vector
Flat goal:	Tessellation

4. **What is the most common land use in each PC4 neighborhood?**

   begin state: this question will consist of a polygon dataset that denotes the category of land use for each plot of land within the municipality of Amsterdam and a dataset that contains the polygons of each PC4 area in Amsterdam.

   Goal specification: The goal for this tessellation is to identify the most common usage of land in terms of surface area for each PC4 area.

Begin state:	NominalA∩Coverage, TesselationA∩VectorA
Goal specification	NominalA∩Lattice
Flat begin state:	Vector, Vector
Flat goal:	Tessellation

5. **What is the estimated temperature within each PC4 area in Amsterdam on Wednesday the 26th of June?**

   begin state: this question will consist of a polygon dataset that denoted the shape of each building and its respective function and a dataset that contains the polygons of each PC4 area in Amsterdam.

   Goal specification: The goal for this summary of the average Euclidean distance to a sporting facility within each PC4 Area.

Begin state:	IntervalA∩ PointMeasures, TesselationA∩VectorA
Goal specification	IntervalA∩Lattice
Flat begin state:	Point, Vector
Flat goal:	Tessellation

5.4 Evaluation framework creation

With the workflows created for the 5 competency questions, it is necessary to evaluate the workflows for workflow errors. The need to evaluate the quality of workflows is not without precedent. However, as there is no readily available body of literature for GIS workflow evaluation this thesis shall aim to provide some constructs to evaluate workflows by categorizing the errors encountered in automatically generated workflows during the manual evaluation of these workflows. These constructs will form the basis for the workflow evaluation framework.

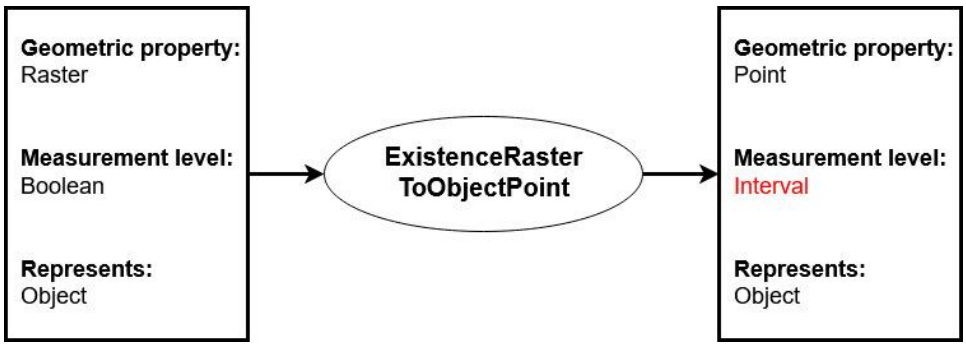
5.4.1 Hard errors

During the workflow evaluation, two distinct types of errors were identified.
The first type of error is a hard workflow error, this type of error renders the workflow completely meaningless for its intended purpose. The workflow becomes meaningless in the sense that none of the generated datasets are capable of answering the competency question. In this category, there are two distinct types of hard errors.

5.4.1.1 Synthesis engine error

The first type of hard workflow error is the synthesis engine error. One of the most important capabilities of the workflow synthesis engine is the proper interpretation of input data types and the proper creation of output datatypes. However, due to the fact that the current method used is only a prototype and the APE synthesis engine is still under development it is capable of making errors. An example of such an error is the erroneous assignment of the interval measurement level to geometric data types when the output doesn't have a specified measurement level. An example of such an error occurred when an object point dataset was created from an existence raster (figure 5.12). A raster needs to hold a boolean value to denote the existence of an object, but an object point by itself already denotes the existence of an object. As such it doesn't need a value and doesn't have a measurement level. However, the synthesis engine can't handle the absence of a measurement level and thus assigns it the interval level. Even though this specific workflow did manage to achieve its goal specification it achieved the goal by generating an erroneous output, which makes the found solution invalid.

Figure 5. 12 an *excerpt of a workflow that demonstrates a synthesis engine workflow error.*



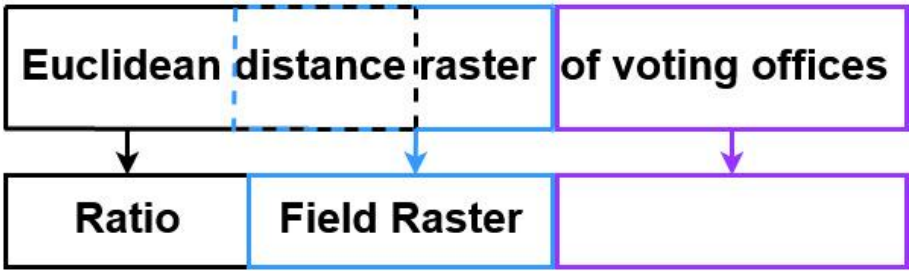
Any workflow that includes a wrongly synthesized output or uses a tool that shouldn't be usable due to the semantic limitations for the purpose of achieving the goal specification will be categorized as containing a synthesis engine error. Even though this type of error creates an invalid workflow for competency questions, the error will not be used to judge the validity of the ontology as the origin of the error was born in the software used and not the ontology.

5.4.1.2 Data type error

The second type of hard error is a data type error. The ontology used for this thesis is based on the semantic data types ontology by Simon Scheider. This ontology is based on three different elements of layers, being: geometric properties, thematic content, and measurement level. However, these three elements are far from the complete documentation of geo-analytical knowledge. As such it isn't possible to fully describe a dataset or goal specification based on this ontology. This limited description leaves room for errors, that result in workflows incapable of answering the competency question. An example of such an error occurs when the synthesis prototype is used to generate a euclidean distance raster for voting offices. Thematic properties

of data such as distance, density, sum, etc. are not included in the ontology and as such the synthesis engine can't interpret them as a goal or as an annotation. Instead, we can only define the semantic properties included in the ontology that would coincide with a Euclidean distance raster of voting offices. For instance, a Euclidean distance will always be a ratio level measurement and a distance raster will always be a field raster. So the most precise goal specification we can create of a Euclidean distance raster of voting offices only conveys part of the intended meaning (figure 5.13).

Figure 5. 13 translation of competency question to goal specification



As such it is possible for the synthesis engine to create a state with a ratio field raster that doesn't answer the proposed competency question as the synthesis prototype will assume that any state that includes a ratio field raster is a valid answer to the question, while the actual answer is more nuanced. It might create a ratio raster based on a different subject than voting offices or it might create a density raster, which answers a different spatial question but still is a ratio field raster. This inability to specify certain differences is well captured in the dataset descriptions (figure 5.14.), where two distinctly different datasets have the same annotation.

Figure 5. 14 Example of an RDF description of two datasets with the same semantic data signature

```
<Https://Maps.Amsterdam.nl/PettingZoos> a ccd:ObjectPoint ;
    a ccd:NominalA.
    a exm:IRA.
    rdfs:comment "this point datasets denotes the location and name of petting zoo's in Amsterdam".

<Https://Maps.Amsterdam.nl/SportFacilities> a ccd:ObjectPoint ;
    a ccd:NominalA.
    a exm:IRA.
    rdfs:comment "this point datasets denotes the location and name of sport facilities in Amsterdam".
```

Any workflow that correctly achieves its goal specification but doesn't actually answer the competency question can be considered to contain a data type error. Meaning that the used ontology simply isn't extensive enough to guide the synthesis engine to the proper goal. These types of errors reduce the validity of the ontology but are excellent for the identification of weak spots within the ontology.

5.4.2 Soft errors

The second type of workflow error is a soft workflow error. This type of error renders the workflow of lesser quality than the optimal workflow but doesn't prevent the workflow from answering the competency question. The reduction of workflow quality is still considered an error, as workflows of lesser quality might be less capable of answering the question correctly or waste resources doing so.

5.4.2.1 redundant tool error

The first type of soft workflow error is a redundant tool error. A property of a fully valid GIS workflow is that each layer generated within the workflow has a connection in terms of usage to the final objective. A tool that isn't connected in terms of output to the final goal specification through a chain of output usage can be considered redundant as the data generated by the said tool isn't used anywhere in order to answer the question.

When automatically generating workflows the synthesis engine combines processes unit the goal is achieved. However, if the goal can be achieved in a singular process the synthesis engine doesn't stop. It tries to generate any combination of workflows that results in the final goal specification. As such it can also combine the one tool solution with different tools that aren't used for the goal specification in order to create extra workflows. The inclusion of such redundant tools creates a lot of unnecessary clutter in a workflow that might increase the execution time or cost of such a workflow. As such we consider any workflow containing redundant tools to have a soft error.

A second error caused by the addition of a random tool is that the same core workflow solution can be synthesized multiple times with just the addition of a random tool. This is caused by the fact that the order of tools is arbitrary as long as they are used before the necessary operation that requires their output as input (Son, Sun Kim, & Ho Kim, 2005). Given the possibility to place the same process at different locations in the temporal timeline of workflow allows for the creation of multiple workflows (figure 5.15) while the actual graph structure of the workflow remains the same (figure 5.16). In figure 5.15 we describe two different workflows as a temporal execution of a workflow. From this perspective, there is a clear difference between the first workflow and the second. The first workflow starts out with the "Coverage to Field Raster" process while the second workflow starts with the "Dissolve" process. However, after they have finished their first process, they execute the other process respectively and finally finish with the same process that is dependent on the completed execution of both processes. It is possible to describe both workflows using a singular workflow graph (Figure 5.13). This is because the two workflows are actually just a singularly unique combination of processes instead of two unique combinations. The graph makes it clear that the workflow is actually two parallel workflows. This difference in temporal placement allows for the generation of multiple workflows that in essence are the same.

Figure 5. 15 Example of APE workflow notation

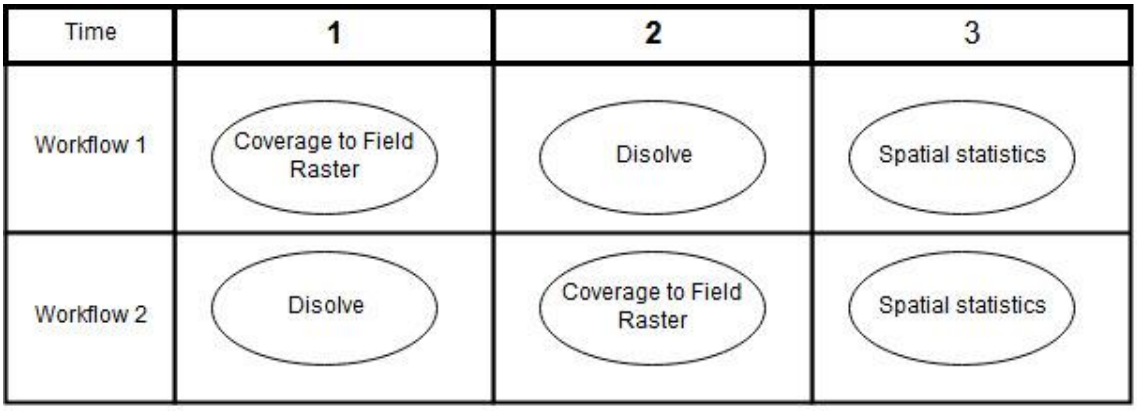
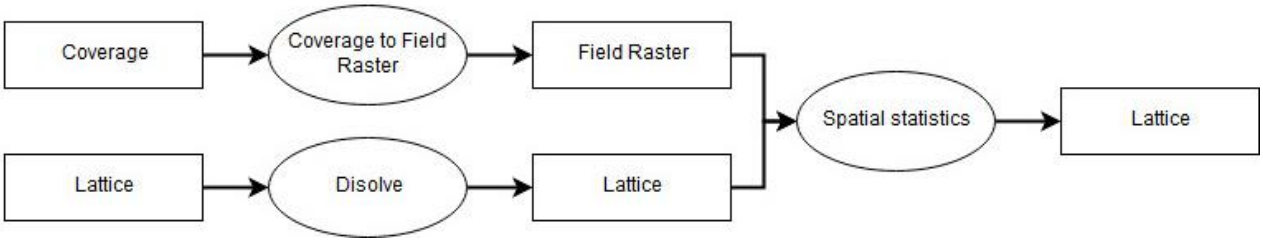


Figure 5. 16 Example APE workflow as a workflow graph.



This problem is exacerbated by the introduction of redundant tools as these are completely independent of any process and can be placed at any time location within the workflow except the last due to the synthesis restriction. This addition of unnecessary tools clutters the workflow and increases the cost/production time of a workflow. As such this type of redundant tool is considered a soft workflow error as the workflow containing such a tool is still capable of answering the competency question but with a reduced quality compared to the optimal solution.

5.4.2.2 Loss of data quality

The second type of soft workflow error is the loss of data quality. Spatial data has a large set of attributes that determine the quality of data (Devillers & Jeansoulin, 2006). For this thesis, the most relevant measure of geographic data quality is
the measure of resolution. For the meaning of geographic data, raster and vector data can be considered interchangeable as it's simply a model representation of the phenomenon (Simon Scheider, 2019). However, the transformations between vector data and raster data can result in a loss of quality in the resolution of the data. For instance, when an object point is transformed into an existence raster the processes lose some of the resolution of the point. This is due to the fact that a raster cannot denote the exact location of an object but only denote its presence within a grid square. This introduces a margin of error for the actual location of the object that simply wasn't present in the original vector representation (Maffini, 1987). So even though an existence raster and an object point might both be valid input for a process, the transformation of an object point to an existence raster would result in less accurate results because the existence raster has a lower resolution. Workflows that unnecessarily lose resolution in their data are considered to be of lesser quality and thus contain soft errors.

Data quality can not only be affected by transformations between representations, but it can also be affected by reclassifying the attributes of the datasets. Within statistics, there are four commonly agreed-upon measurement level for attributes. It is a common practice to always use the highest possible measurement level as this type of data carries the most information. However, it is possible to reclassify each instance of a higher measurement level to a lower measurement level. Some processes accept multiple levels of measurement, meaning that in this case the highest measurement level should be used for maximum information quality. The nature of automatic workflow synthesis allows for the degradation of measurement levels in order to create a new unique workflow. These workflows are considered of lesser quality and are thus considered to contain soft errors.

5.4.2.3 Workflow evaluation framework

The creation of workflows by a synthesis engine can be considered a classification problem. The workflow synthesis engine produces every possible combination of tools that achieves the goal specification based on the input limitations and output description in the tool repository. As such it only reports workflows that have achieved the goal specification and are classified as valid workflows. In order to asses the extent to which the synthesis prototype is capable of answering the competency questions it is necessary to evaluate the workflows that have been classified as

valid. This is done by identifying the errors in each workflow manually and classify them according to the postulated workflow error types.

A common method for evaluating the performance of a classification problem is a confusion matrix (figure 5.17). the confusion matrix differentiates classifications according to two dimensions, being actual and predicted, and positives and negatives. According to this method, classifications of workflows can fall into four different categories.

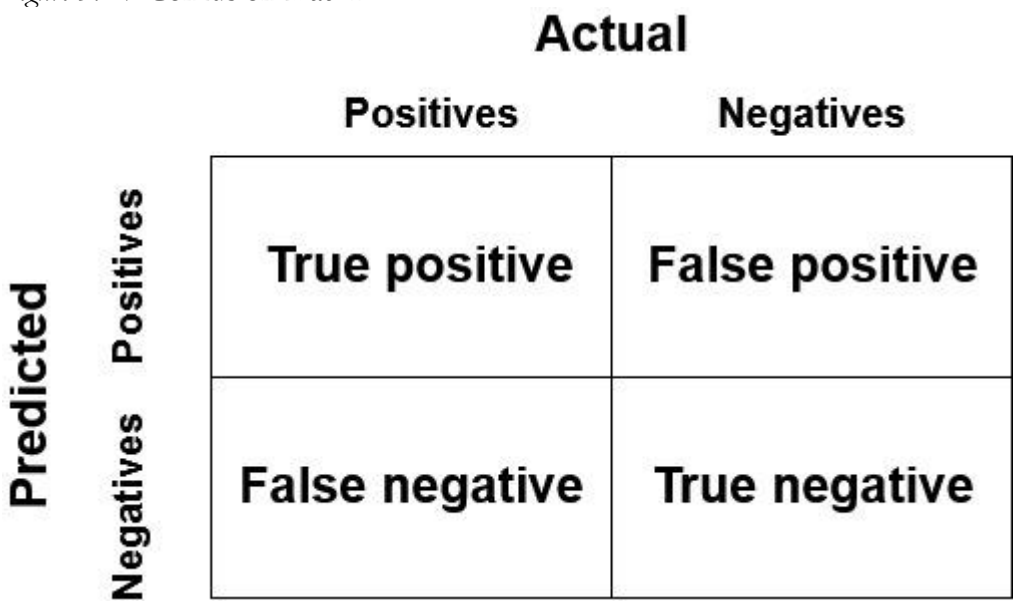
True positive: If the workflow was valid according to the expert and was predicted as such by the workflow synthesis engine it's a True Positive (TP).

False negative: If the workflow was valid according to the expert, but the workflow synthesis engine determined the workflow to be invalid it's a False Negative (FN)

False positive: If the workflow was invalid according to the expert but predicted as a valid workflow it's a False Positive (FP)

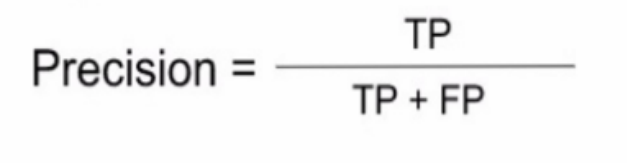
True negative: If the workflow was invalid according to the expert and predicted as such by the workflow synthesis engine it's a True Negative (TN)

Figure 5. 17 Confusion matrix



With this confusion matrix, it normally is possible to measure four different metrics of classification being accuracy, precision, recall, and specificity by combining TP, FP, FN, and TN in specific ways. However due to the fact that the workflow synthesis engine only reports the predicted positive workflows and doesn't report the negatively predicted workflows we only have true positives and false positives. The only metric that can be calculated with these measures is the precision of the workflow synthesis engine. Precision is calculated by dividing the number of correct workflows through the total of the workflows that were predicted to be correct (figure 5.18).

Figure 5. 18 Precision calculation



For the purpose of this thesis, precision is a measure that tells what the proportion of the predicted workflows was correct. A fully functional workflow synthesis should have a precision of 100% as each presented workflow should be a fully functional method for answering the posed question.

The goal of the geo-analytical knowledge formalization is to capture all the intricacies of workflow generation within a single formalized framework in order to produce meaningful workflows. By calculating the precision for the geo-analytical knowledge formalization we can evaluate its capacity for answering spatial questions. The higher the precision of the ontology the better the capacity for answering questions. The precision will be calculated for each type of error

5.4.2.4 Ontology evaluation

In order to test each ontology, we specified several competency questions and ran these in both the full and flattened synthesis prototypes. By evaluating each workflow manually and identifying each type of error within a workflow it becomes possible to calculate the precision for each type of error for both the full and the flattened geo-analytical knowledge formalization. This thesis will calculate the precision for each respective error type and for the workflow as a whole. Doing so we can evaluate each respective ontology and their capacity for answering spatial questions, while also demonstrating the viability of workflow synthesis for GIS.

6 Results

In order to evaluate the capacity of the semantic data types ontology for answering spatial questions two different versions of the GIS workflow synthesis prototype were created. The full version was based on the complete semantic data types ontology (figure 4.9), with the exception of elements that were related to networks or events. It included all the GIS tools specified in table 5.1. The second version was the flattened version, which was created to serve as a reference version based on the semantics normally included as limitations for GIS software. In order to create this flattened formalization, all semantics that weren't included in the limited ontology were flattened to their least common subsumer that still existed in the limited ontology. Doing so, it was possible to create an ontology, data repository and tool repository which only take the geometric semantics of line, point, vector, raster and tessellation into account. For each of these prototypes, the competency questions specified in paragraph 5.3.1 were translated to a begin state and goal specification for each respective version of the prototype. The begin state and goal specification were entered into the synthesis engine, the resulting workflows have a distinct sequence of the form *begin state data types, tool, data type, tool, data type, tool, goal specification*. As the synthesis problem was approached as a global problem the data types can be reused along the pipeline. We shall present several examples in order to illustrate the synthesized workflows.

Question 1: What is the number of sports facilities in each PC4 area?

Workflow 1: *NominalAObjectPoint(T0.0) TessellationAVectorA(T0.1) SpatialJoinCountTess(M1) CountALattice(T1.0)*

This is the first output when both the formalized begin state, and goal specification were entered into the full workflow synthesis prototype. The workflow uses a spatial join with the count parameter on the nominal object points that represent the sports facilities and vector tessellations that represents the PC4 areas, in order to produce a lattice of counts of these facilities within each PC4 area. Coincidently this workflow is the most optimal way of answering the specified synthesis question and contains no workflow errors.

Question 2: What is the distance to the nearest voting office for the area of Amsterdam?

Workflow 2: *NominalAObjectPoint(T0.0) EuclideanDistanceObjectVector(M1) FieldRasterRatioA(T1.0)*

This is the first output when both the formalized begin state, and goal specification were entered into the full workflow synthesis prototype. The workflow uses a Euclidean distance tool with the object points that represent voting offices in the city of Amsterdam in order to generate a distance raster for voting offices. Coincidently this workflow is again the most optima way of answering the specified synthesis questions and contains no workflow errors.

For each question, the first 20 synthesized workflows were manually evaluated for the occurrence of workflow errors. The identified errors were categorized according to the workflow error types in the evaluation framework specified in chapter 5.4. The results of this evaluation were summarized with the precision metric. The value of the precision metric indicates the occurrence of a specific error type. If the value is 0 every synthesized workflow has at least one error in that specific category, if the value is 1 not a single workflow contains that specific error.

6.1.1 Hard workflow error results

Hard workflow errors are errors that render the workflow unable to answer the competency question. After the manual evaluation of the workflows, the hard errors were summarized into the precision metric. Table 6.1 lists the results of the workflow evaluation for hard workflow errors.

Table 6. 1 1 Hard error precision for both workflow synthesis prototypes

Ontology version	Number of workflows	Average workflow length	Precision: synthesis error	Precision: data type error	Precision: hard error
Full ontology	100	3.04	0.94	0.73	0.68
Flattened ontology	100	1.85	1	0.18	0,18

For both the full and flattened ontology 100 workflows were evaluated, with 20 workflows for each specific question. On average the workflows from the full ontology prototype were considerably longer in length, with an average length of 3.04 tools compared to the 1.85 tools in the flat version. On average 6% of the workflows synthesized by the full ontology prototype contained synthesis errors, while the flat version didn't contain any synthesis errors at all. An example of such a synthesis error was generated for the second competency question.

Question 2: What is the distance to the nearest voting office for the area of Amsterdam?

Workflow 3: *NominalAObjectPoint(T0.0) ObjectPointToExistenceRasterBoolean(M1) BooleanAExistenceRaster(T1.0) ExistenceRasterToObjectPoint(M2) IntervalAObjectPoint(T2.0) FocalStatisticsObjectVector(M3) FieldRasterRatioA(T3.0)*

This is the 9th workflow generated for the synthesis problem. The workflow uses a vector transformation to transform the point dataset representing the voting offices into a raster set that denotes the existence of objects. Afterward, it uses a raster transformation to transform the existence raster into a point dataset. However, due to an error in the workflow synthesis prototype, the returned point dataset suddenly has an interval measurement level. This incorrectly assigned interval measurement level allows for the usage of a focal summarization, which denotes the average value of voting offices within an area. As the voting offices only have a name at a nominal measurement level, there isn't any data to average. This makes the workflow impossible and counts as a synthesis error.

The second type of hard error is the data type error. For this error the full ontology prototype has a precision of 0.73, this means that 73% of all synthesized workflows don't contain any data type errors. For the flattened ontology, this number is much lower, with only 18% of all synthesized workflows not containing any data type errors. An example of such a synthesis error was generated for the second competency question.

Question 2: What is the distance to the nearest voting office for the area of Amsterdam?

Workflow 4: *NominalAObjectPoint(T0.0) FocalStatisticsObjectVector(M1) FieldRasterRatioA(T1.0)*

This is the second workflow generated for this synthesis question. The workflow uses the focal statistics tool with the object points representing voting offices in order to generate a density raster denoting the local density of voting offices. Although the workflow achieves the goal of a Field raster at a ratio measurement level, it still is incapable of answering what the distance is to the nearest voting office. This is due to the fact that the goal specification can not fully capture the necessary elements needed to fully specify the goal as it can only specify geometric properties, measurement levels, and core concepts. As such it leaves room for alternative solutions that share a data type specification but can not answer the geo-analytical question. Any such workflow can be said to be caused by underspecification of the goal, which in turn is caused by a lack of semantics.

In order to answer a spatial question, the workflow cannot contain a single hard error. For the full ontology, 68% of the workflows didn't contain a single hard error and as such were capable of answering the competency question to some extent. The flattened ontology performed much worse with only 18% of the workflows not containing any hard workflow error.

6.1.2 Soft workflow error results

The second type of error is a soft workflow error. This type of error renders the workflow of lesser quality than the optimal workflow but doesn't prevent the workflow from answering the competency question. Just like the hard workflow errors, the soft workflow errors were summarized in a precision metric after being manually evaluated. Table 6.2 lists the results of the workflow evaluation for soft workflow errors.

Table 6. 2 Soft error precision for both workflow synthesis prototypes

Ontology version	Number of workflows	Average workflow length	Precision: redundant tool	Precision: loss of data quality	Precision: soft error
Full ontology	100	3.04	0.27	0.82	0.18
Flattened ontology	100	1.85	0.62	0.95	0.58

For the soft workflow errors, the same set of workflows was evaluated as for the hard workflow errors. On average the full ontology prototype included at least 1 or more redundant tools in 73% of the workflows, resulting in a precision of only 0.27 for redundant tools. Surprisingly the flattened ontology has a much better performance with only 42% of the workflows containing a redundant tool, resulting in a precision of 0.62. An example of such a redundant tool error was generated for the first competency question.

Question 1: What is the number of sports facilities in each PC4 area?

Workflow 5: *NominalAObjectPoint(T0.0) TessellationAVectorA(T0.1) ObjectPointToExistenceRasterBoolean(M1) BooleanAExistenceRaster(T1.0) SpatialJoinCountTess(M2) CountALattice(T2.0)*

The workflow uses a vector transformation on the nominal object points that represent the sports facilities, resulting in an existence raster that denotes the presence of one or more sports facilities at a location. Secondly, the workflow uses a spatial join with the count parameter on the nominal object points that represent the sports facilities and vector tessellations that represents the PC4 areas, in order to produce a lattice of counts of these facilities within each PC4 area. The initial transformation of the object points to an existence raster wasn't necessary for the creation of the lattice and can be considered a redundant tool as its output wasn't used in any way, shape or form to create the final goal specification.

The error of data quality loss is much less prevalent than redundant tools. Off the workflows generated with the full ontology prototype, only 18% of the workflow included the unnecessary reduction of data quality, this results in a precision of 0.82. The flattened ontology does perform better than the full ontology when it comes to data loss, with only 5% of the workflow including an unnecessary loss of data quality, resulting in a respectable precision of 0.95 for data quality loss. An example of such a data quality loss error was generated for the second competency question.

Question 2: What is the distance to the nearest voting office for the area of Amsterdam?

Workflow 6 *NominalAObjectPoint(T0.0) ObjectPointToExistenceRasterBoolean(M1) BooleanAExistenceRaster(T1.0) EuclideanDistanceObjectVector(M2) FieldRasterRatioA(T2.0)*

The workflow transforms the object points that represent voting offices to an existence raster that denotes the presence of one or more voting offices within a grid cell. This reduces the precision of the location from a point to an entire grid. Next, the workflow calculates the euclidean distance based on the existence raster, resulting in a raster denoting the distance to the nearest voting office. Although this method is valid for answering the posed question, the question could have been answered more accurately is the euclidean distance was based on the point dataset instead of the existence raster.

In order to answer the spatial question in the most efficient way, the workflow should not contain any redundant tools or reduce any data quality unnecessarily. For the full ontology, only 18% of the workflows do not contain any soft workflow errors. The full ontology performed much better in this category with an average of 58% of the workflows not containing any soft errors.

Table 6. 3 total error precision for both workflow synthesis prototypes

Ontology version	Precision: hard error	Precision: soft error	Precision: any error
Full ontology	0.68	0.18	0.07
Flattened ontology	0.18	0.58	0.05

If we were to evaluate the capacity of answering spatial questions in the most strict fashion possible, any workflow containing any type of error would be considered invalid. If we apply this principle to the full ontology workflows only 7% of the workflows can be considered valid and for the flattened ontology only 5%.

6.2 Ontology evaluation

By comparing the results of the workflow synthesis with the opinion of an expert it is possible to evaluate the functional capacity of the semantic data type ontology. A perfectly functional ontology that covers the entirety of geo-analytical knowledge should have the capacity to answer geo-analytical questions without any error. However, as can be seen in table 6.3 the full semantic data type ontology doesn't have a precision of one and creates workflows that contain both soft and hard workflow errors.

However, when we compare the data type ontology with the flattened ontology that is supposed to serve as a baseline, its performance is drastically better when it comes to the avoidance of hard errors. This is due to the fact that the semantics data type ontology is designed for specifying data set properties that allow for meaningful usage of tools. By introducing semantics such as geometric properties, core concepts, and measurement levels it becomes possible to exclude tools that cannot be connected meaningfully. This drastically improves the quality of the generated workflows as there is less illegal tool usage in them. In the generated workflows only a couple instances of illegal tool usage were found and those were mainly the result of a synthesis error in which the synthesis prototype couldn't interpret the semantics properly. However, despite the fact that the tools used within the workflow are used properly there remain workflows that are incapable of answering the specified questions.

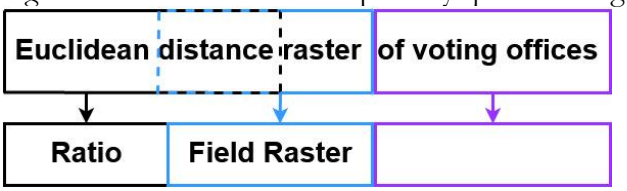
The cause of these errors lies with the fact that the semantic data type ontology was created for meaningful usage of tools, but wasn't designed with the goal of answering spatial questions. In order to answer specific spatial questions, the goal needs to specified further. Currently, the ontology has no way of specifying concepts such a distance, density, sums, majority or mean. As such it can only specify the type of data that would be present in such a measure but can't demand the measure itself. An example of such underspecification can be demonstrated with the second competency question.

2nd Question: What is the distance to the nearest voting office for the area of Amsterdam?

In order to answer this question, we would need a Euclidean distance raster of voting offices. However, the ontology currently has no way of defining distance as a data type let alone euclidean distance, nor can it define data states relative to being generated by a specific dataset. As such we can only define the goal state by mentally generating our own closest semantic subsumer of our goal (figure 6.1). The closest semantic included in the data types ontology to distance is a ratio measurement level, as distance always has a ratio measurement level. The closest semantic included in the data types ontology to distance raster is a field raster. The third element defining the Euclidean distance raster as originating from a calculation based on voting offices doesn't even have a semantic subsumer in the data type ontology. As such the goal definition becomes so broad that it allows room for error. Any other possible tool usage that leads to the same data type might also be considered valid for the goal specification but in reality,

be unable to answer the geo-analytical question. This lack in specificity allows for most of the data type errors in the full ontology workflow synthesis prototype.

Figure 6. 1 translation of competency question to goal specification



Despite this limitation, the semantic data type ontology is still capable of answering the geo-analytical question in 68% of the cases, which is an impressive rate of success, especially when compared to the flattened semantics that only manages to answer the geo-analytical questions in 18% of the cases. As such we can conclude that the inclusion of the semantics captured within the semantic data type ontology bears a significant influence on the capacity of workflow synthesis to answer geo-analytical questions. As such the semantic data type ontology can be considered to be valid. However, the ontology itself is far from complete as a representation of geo-analytical knowledge as it still generates errors when implemented.

The occurrence of soft data errors are of secondary concern to the capability of an ontology for answering geo-analytical questions but still bear some relevance as the inclusion of soft errors lowers the quality of the synthesized solution. Within the implementation of the full ontology, the occurrence of soft errors mainly stems from the usage of redundant tools. with an occurrence rate of 73% compared to the rate of 38% in the flattened implementation. The full version has a significantly higher proportion of redundant tool errors. At first glance one would assume the full ontology lacks the expressive power compared to the flattened ontology to prevent redundant tools. However, as it stands neither has any semantics that define the usage of tool output for the final goal specification. So neither has any expressive capabilities to limit redundant tool usage. As such the lower rate of redundant tool errors is merely an artifact of the average workflow length. The average workflow length in the flattened version is a lot shorter due to the fact that more tools are considered valid and the goal specification is less specific. In the flattened workflows, we can see that this allows for the easier combination of tools and the quicker achievement of the goal specification with a more diverse range of tools.

As the competency questions were formulated with the idea of generating at least 20 workflows for each question the flattened ontology has more room for generating unique tool combinations that don't include a redundant tool, no matter how incorrect the method factually is. The full ontology has more tool limitations, limiting the unique number of tool combinations that can achieve more specific goal specifications. Once the unique tool combinations of the shortest length had been synthesized the synthesis engine generated the same combination only with the addition of an extra redundant tool. As such the validity of the data type ontology promoted the occurrence of redundant tool errors in the generated workflows by reducing the number of shortest possible workflows. The same artifact occurred for the reduction of data quality. In order to reduce data quality you need an extra tool to reduce said quality, as the flattened ontology was capable of creating more incorrect short solutions the occurrence of data quality reduction in the first 20 workflows was observed to be significantly less with an occurrence rate of 5% in the flattened ontology compared to the occurrence rate of 18% in the full ontology version.

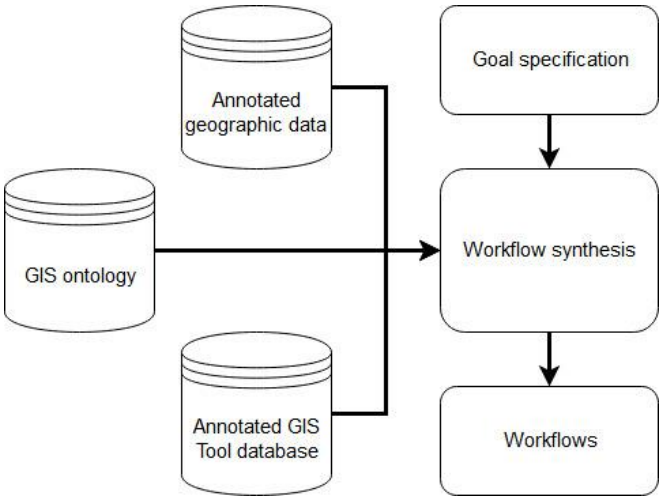
7 Conclusion

The next step for scientific workflow support systems in the field of GIS is the creation of a GIS workflow synthesis engine. This type of system would allow for the automatic bridging of data states to goal specifications. This type of systems may aid the users of GIS to quickly create meaningful workflows for their geo-analytical goals and aid non-expert users in their usage of GIS. For this purpose, the APE GIS workflow synthesis prototype was created. Helping develop this prototype led to a clear answer to the first of the main research questions of this research:

> *How can an ontology-based workflow synthesis prototype for automatic GIS workflow composition be created?*

In order to create a GIS synthesis workflow prototype four different technical elements are required; a machine-readable ontology, an annotated tool database, an annotated geo-data database and a synthesis engine (Figure 7.1).

Figure 7. 1 model of necessary elements for ontology-based workflow synthesis for GIS.



The geo-analytical ontology serves as a formalization of the geo-analytical knowledge within the domain of GIS and allows the synthesis engine to interpret the relations between the semantics that were used to describe the goal specification, the tool annotations, and the data annotations. The GIS ontology also serves as the basis for the annotation of the tools and data. As such, the semantics captured within the ontology can be taken into consideration for the workflow synthesis. The annotated data is necessary to define the begin state of the workflow synthesis. Each data set in the begin state is specified in accordance with the semantics included in the ontology. This initial data state serves as the starting point for the synthesis engine to combine the annotated tools with the data. Each tool needs to be annotated with a semantic description of valid input data type and a semantic description of the generated output data type. If the annotated data matches with the input data type of a tool in the tool repository it can be implemented by the synthesis engine to generate a new data state. This new data state consists of the original data state, plus the output of the implemented tool. The synthesis engine will combine data and tools in this fashion until the goal specification is achieved, resulting in a functional GIS workflow synthesis prototype.

Currently, the field of GIS doesn't have a geo-analytical ontology that can cover the full spectrum of semantics necessary for completely functional workflow synthesis. However, there are several ontologies that address parts of the necessary knowledge that can be formalized into a

machine-readable ontology using an ontology editor such as protégé. For this research, the semantic data type ontology by Simon Scheider (Simon Scheider, 2019) was formalized. With this formalization, it was possible to annotate several GIS tools and GIS databases in an RDF format. The creation of both these datasets is necessary for the creation of a workflow synthesis prototype as there aren't any openly available repositories of annotated GIS tools and data that match the ontology.

In order to create a GIS workflow synthesis prototype the formalized ontology, the annotated tools, and annotated data were combined with the APE (Automatic Pipeline Engine) workflow synthesis engine. In order to import the tool description, the database needs to be translated from RDF to XML. This was done by a script that automatically translates tool descriptions from RDF to XML (appendix A). APE was developed for the research of Vedran Kasalica and Anna-Lena Lamprecht (Kasalica & Lamprecht, 2018) and allows for the creation of workflows using Semantic Linear-Time Logic (STL). This engine was selected as it was specifically designed for GIS workflow generation. By combining APE with the ontology and the annotated tools and data it is possible to create a functional workflow synthesis prototype.

The creation of a GIS workflow synthesis prototype doesn't mean that the prototype is also perfectly capable of synthesizing GIS workflows. If this type of system is ever to be a workflow management system, it requires reasonable reliability. As such the workflows that are created by the workflow synthesis engine require evaluation in order to determine the validity of the prototype and the underlying ontology. The search for a method to evaluate workflows led to the answer to our second main research question.

How can synthesized workflows be evaluated for their capacity to answer geo-analytical questions?

There is no readily available body of literature that focusses on the evaluation of GIS workflows. As such, this research used a qualitative approach for creating a workflow evaluation framework. By manually evaluating the workflows and identifying error types it is possible to construct a framework with the different error categories. This framework allows the reviewer of the workflow to categorize the encountered errors into specific error types (table 7.1).

Errors are categorized in accordance with their impact on the capacity of a workflow to answer the posed geo-analytical question. There are two distinct classes of workflow errors in terms of impact, the first class is the hard error and the second class is the soft error. A hard error renders the workflow completely meaningless for its intended purpose. When such an error occurs, the generated datasets are incapable of answering the posed geo-analytical question, or the proposed workflow cannot be executed. A soft error renders the workflow of lesser quality than the optimal workflow but doesn't prevent the workflow from answering the posed geo-analytical question. Workflows of lesser quality might be less capable of answering the question correctly or use unnecessary resources in doing so. If a workflow doesn't include any error, it can be considered a completely valid method of answering the posed geo-analytical question.

table 7. 1 Error types included in the evaluation framework.

Error class	Error type	Error description
Hard error	Synthesis error	An error that occurs due to the wrong interpretation of input and output data types by the synthesis engine.
	Data type error	An error that occurs due to the lack of expressive power within the ontology. Any workflow that achieves the goal specification but is still unable to answer the posed question can be considered to contain a data type error.
Soft error	Redundant tool error	An error caused by the inclusion of a redundant tool in the workflow. A tool is considered redundant when its output doesn't contribute to the goal specification.
	Loss of data quality.	An error caused by the unnecessary transformation of data between geometric data types and measurement levels. When data is transformed precision is lost, resulting in a workflow of reduced quality.

With the workflow evaluation framework, it becomes possible to categorize workflow errors and by doing so evaluate the synthesized workflow in a structured manner. The workflow synthesis prototype is based on the data type ontology. As a prototype, it is important to evaluate its capacity to answer geo-analytical questions through synthesis. In order to have a reference for the capacity, a second workflow synthesis prototype was created based on the flattened ontology that only included semantics commonly found in GIS software. Both of these versions were matched with the problem of generating a workflow for 5 geo-analytic competency questions. By comparing the rate of errors in the first 20 workflows of each question it becomes possible to compare the capacity for answering geo-analytical questions. Doing so allows us to answer the final main research question.

> *What is the capacity of the semantic data type ontology implemented in a workflow synthesis prototype to answer geo-analytical questions?*

Both versions of the prototype are barely capable of creating workflows that are completely error-free. The full ontology version only achieves zero errors in 7% of the workflows while the flattened version performs even worse, achieving a rate of 5%. However, there is a big difference in the type of errors that are generated in each respective version. The full ontology version only produces a hard error in 32% of the cases, making it capable of answering a geo-analytical question in 68% of the cases. Comparatively the flattened version only manages to produce workflows capable of answering the posed question in 18% of the cases. As such it can be concluded that the full ontology has a significantly higher capacity for answering geo-analytical questions. This demonstrates that the geometric properties, measurement levels and core concepts included in the semantic data type ontology drastically improve the capacity of workflow synthesis to answer geo-analytical questions. This is because the ontology provides some very relevant semantics for determining the validity of tool usage. By reducing meaningless tool usage, the occurrence of hard workflow errors drops drastically in comparison to the flattened ontology. This difference could have been even higher as the flattened ontology includes 0% synthesis errors and the full ontology contains 6% synthesis errors of which 5% are the exclusive reason for the failure of the workflow. This difference in synthesis error can be explained by the difference in the complexity of the ontology. The inclusion of multiple semantics and the creation of a more complex leave taxonomy for the synthesis engine allows for more synthesis errors.

The main cause of the hard workflow errors in the full semantic data type ontology lies in the fact that the ontology was only designed with the goal of meaningful workflow usage in

mind. As such it specifies the data types that can be used in a meaningful way. However, the ontology wasn't designed to codify the goal specifications. Because of that, it underspecifies certain geographic measures such as distance, density, majority, median by only being able to specify the geometric properties, measurement level and core concepts of such measures. This broad specification allows for multiple tools to provide the goal specification while only a singular tool might be able to provide the correct measure. Secondly, the ontology is incapable of defining the subject of the dataset, if two point-datasets with different subjects are placed into the begin state of the synthesis, the synthesis prototype can not decipher which point dataset is the subject for the goal specification.

When the performances of the synthesis prototypes are solely based on soft workflow errors the performance of the full ontology drastically drops. The full version has an occurrence of a soft workflow error in 82% of the cases, while the flattened ontology only has a 42% soft error rate. As discussed in the results this difference of soft workflow error occurrence can mainly be attributed to the difference in workflow length. The full prototype has an average workflow length of 3.04 tools and the flattened version has an average workflow length of 1.85 tools. Longer workflows allow for the occurrence of more soft errors, by providing more room to include redundant tools and more room for tools to arbitrarily transform data. However due to the low impact of soft workflow errors the high occurrence rate isn't problematic for the capacity of the full prototype to answer geo-analytical questions. The occurrence of soft workflow errors can mainly be assigned to the fact that the synthesis prototype is unable to determine the presence of redundant tools and cannot identify the unnecessary loss of data quality.

Even though the prototype currently suffers from underspecification in terms of goal specifications and needs a method to reduce soft workflow errors. The fact that the full ontology prototype achieved a capacity of 68% for answering the geo-analytical questions is a very promising start for this type of workflow support system. Further development of the ontology and synthesis engine could provide drastic improvements in an already impressive capacity.

8 Discussion and future work

Due to the time limitations of this research, only a small set of all possible tools was annotated and formalized within the tool data set. This is relevant as the number of available tools influences the capacity of the workflow synthesis prototype. The prototype currently isn't capable of fully specifying the goal specification. Because of that, it underspecifies certain geographic measures such as distance, density, majority or median by only being able to specify the geometric properties, measurement level and core concepts of such measures. This broad specification allows for multiple tools to provide the goal specification while only a singular tool might be able to provide the correct measure. With an increase in tools, the number of tools that can achieve a specific data state will increase. As a result of this increase in tools, the probability that the correct tool will be selected decreases, because the ratio of correct tools to wrong tools decreases. In order to increase the realism of the capacity test, future work should consist of creating a larger tool repository.

A second limitation for the calculation of the capacity is the fact that the performance metrics were limited to the precision of the generated workflows. The APE synthesis engine only specifies workflows that successfully achieved the goal specification, any workflow that didn't manage to achieve said specification is simply deleted. As a result of this, the output of the synthesis engine doesn't provide negative results. Because of this, it was impossible to calculate other classification metrics such as recall, sensitivity, and specificity. For future work, it might be necessary to adapt the workflow synthesis prototype to also report negative workflows. This would allow the creation of a more diverse range of evaluation metrics.

A second limitation of the workflow synthesis engine is the fact that the leaf method of the APE synthesis engine limits the construction of valid workflows. This is because more specific leaves such as a FieldRasterRatio, can't be used as input for more broad leaves such as FieldRaster. This is because the leave system doesn't allow for the specification of subclasses, as such, it perceives the FieldRasterRatio and FieldRaster as two wholly unique data types that have no relation to one another. Because of this method of workflow synthesis, some meaningful workflows are not constructed, affecting the capacity of the workflow synthesis engine. If the ontology is expanded upon in the future by the addition of new semantics, the leaf method will also generate a larger number of possible data type permutations. This increase in permutations will results in a higher occurrence of mismatches, as such the leaf problem is a serious limitation for the development of this synthesis method. An important focus of future work for workflow synthesis should be the improvement of the workflow synthesis prototype so the leaves can be interpreted as sub and super-classes based on the ontology.

A third limitation for the method used for calculating the capacity of the prototype was the fact that the evaluation of workflows was limited to the first 20 workflows instead of all workflows up to a certain tool length. As such the shortest 20 possible workflows were the only workflows to be evaluated. In the case of the flattened synthesis version the fact that the flattened ontology allowed for a larger number of short workflows to be created compared to the full ontology version warped the respective occurrence of soft workflow errors. Said errors need a workflow length of at least two in order to occur. With an average workflow length of 1.85, many workflows couldn't even contain a soft workflow error. For future work, the workflow evaluation should evaluate all workflows up to a certain length for both the versions of the prototype.

A fourth limitation for the method used for calculating the capacity of the prototypes was the fact that the initial data states only specified the necessary elements for the posed question. A proper workflow synthesis engine should be capable of identifying the necessary data and only using said data for the workflow. If more data would have been added to the initial dataset it's likely that the prototype would have lower capacity as it wouldn't be able to differentiate between different subjects of datasets. An important improvement for future work would the capability to define the subject of a dataset for the synthesis prototype and to describe the goal specification as being based on a dataset describing a specific subject.

The capacity of the full ontology for answering geo-analytical questions is quite substantial with the current operationalization of the workflow synthesis prototype. However, in order to gain a more realistic view on the actual capacity for answering geo-analytical questions more tools need to be formalized. The evaluation of the workflows generated by the full ontology prototype demonstrated some clear weaknesses for workflow synthesis in the data type ontology. The fact that the ontology can not specify measures such as density, distance, median or majority limits the goal specification and allows the synthesis to come up with invalid solutions. A valuable addition in future work would be the creation of a measure ontology that links the measures to their respective data type semantics and allows for the specification of certain measures in both the goal specification, dataset repository, and the tool database. This addition could go a long way in reducing the number of data type errors. Secondly, the synthesis engine would be greatly benefitted by the addition of a method that allows the user of the synthesis engine to specify the subject of the final data state. Currently the begin data state only contains a single instance of each data type, but if the synthesis prototype would have been connected to a large data repository it's highly unlikely that each semantic data type would only be represented once. As such the prototype needs a method to specify the subject for the goal specification and a method to identify that subject throughout the workflow. With the structure of the workflow synthesis, such tracking of a subject would be a difficult feat, but the addition would help out greatly with future implementations of GIS workflow synthesis.

A second focus for future work should be the reduction of soft workflow errors. Currently the workflow synthesis prototype can not identify redundant tools, resulting in the creation of many workflows without a real new solution. These workflows have inferior quality and obfuscate alternative solutions. By creating a method that tracks the usage of data up until the final goal specification it should be possible to identify data sets that were unused for the goal specification and if created by a tool can be considered the product of a redundant tool. The synthesis prototype also currently can't interpret the reduction of data quality. In order to alleviate this problem, the concept of GIS data quality needs to be formalized and integrated into the synthesis prototype.

9 Bibiliography

Action Technologies Inc. (1993). Action Workflow System product literature.

Albrecht, J., Derman, B., & Ramasubramanian, L. (2008a). Geo-ontology Tools: The Missing Link. *Transactions in GIS*, *12*(4), 409–424. Retrieved from 8.01108.x

Albrecht, J., Derman, B., & Ramasubramanian, L. (2008b). Geo-ontology Tools: The Missing Link. *Transactions in GIS*, *12*(4), 409–424.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., & Mock, S. (n.d.). Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.* (pp. 423–424). IEEE.

Athanasis, N., Kalabokidis, K., Vaitis, M., & Soulakellis, N. (2009). Towards a semantics-based approach in the development of geographic portals. *Computers & Geosciences*, *35*(2), 301–308.

Barga, R., & Gannon, D. (2007). Scientific versus Business Workflows. In *Workflows for e-Science* (pp. 9–16). London: Springer London.

Bechhofer, S., Buchan, I., De Roure, D., Missier, P., Ainsworth, J., Bhagat, J., … Goble, C. (2013). Why linked data is not enough for scientists. *Future Generation Computer Systems*, *29*(2), 599–611.

Berners-Lee, T., & Hendler, J. (2001). The Semantic Web. *Scientific American*, *284*(5), 34–43.

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, *5*(3), 1–22.

Bjørner, D. (2007). *Software engineering 1: Abstraction and modeling*. Berlin: Springer Science & Business Media.

Chen, L., Shadbolt, N. R., Goble, C., Tao, F., Cox, S. J., Puleston, C., & Smart, P. R. (2003). Towards a Knowledge-Based Approach to Semantic Service Composition (pp. 319–334). Springer, Berlin, Heidelberg.

Chun, S. A., Atluri, V., & Adam, N. R. (2002). Domain Knowledge-Based Automatic Workflow Generation (pp. 81–93). Springer, Berlin, Heidelberg.

Clarke, K. C. (1986). Advances in Geographic Information Systems. *Computers, Environment and Urban Systems*, *10*(3–4), 175–184.

Clempner, J. B. (2017). Classical workflow nets and workflow nets with reset arcs: using Lyapunov stability for soundness verification. *Journal of Experimental & Theoretical Artificial Intelligence*, *29*(1), 43–57.

Deelman, E., Gannon, D., Shields, M., & Taylor, I. (2009). Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, *25*(5), 528–540.

Deelman, E., Mehta, G., Kim, J., Gil, Y., & Ratnakar, V. (2007). Wings for Pegasus: creating large-scale scientific applications using semantic representations of computational workflows. In *Proceedings of the 19th national conference on Innovative applications of artificial intelligence - Volume 2* (pp. 1767–1774). Vancouver: AAAI Press.

Devillers, R., & Jeansoulin, R. (2006). *Fundamentals of Spatial Data Quality*. *Fundamentals of spatial data quality* (1st ed.). London: ISTE.

Esri. (n.d.-a). ArcGIS modelbuilder.

Esri. (n.d.-b). ArcGIS Pro.

Fitzner, D., Hoffmann, J., & Klien, E. (2011). Functional description of geoprocessing services as conjunctive datalog queries. *GeoInformatica*, *15*(1), 191–221.

Freire, J., & Silva, C. T. (2012). Making Computations and Publications Reproducible with VisTrails. *Computing in Science & Engineering*, *14*(4), 18–25.

Freitag, B., Steffen, B., Margaria, T., & Zukowski, U. (1995). An Approach to Intelligent Software Library Management. In *Proceedings of the 4th internatnional conference on database systems for advaced applications* (pp. 10--13).

Frye, C. (1994). Move to workflow provokes business process scrutiny. *Software Magazine*, *14*(4), 77–83.

Gangemi, A., Catenacci, C., Ciaramita, M., & Lehmann, J. (2005). A theoretical framework for ontology evaluation and validation. *SWAP*, *166*, 16.

Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., & Goble, C. (2014). Common motifs in scientific workflows: An empirical analysis. *Future Generation Computer Systems*, *36*, 338–351.

Garijo, D., Garijo, D., De Informática, F., & Gil, Y. (2012). Towards Open Publication of Reusable Scientific Workflows: Abstractions, Standards and Linked Data. Los Angeles: Faculty of

computer science at the University of Southern California.
Gensesereth, M. R., & Nilsson, N. R. (1987). *Logical foundation of artificial intelligence* (2nd ed.). Los Altos: Morgan and Kaufmann.
Georgakellos, D. A., & Macris, A. M. (2009). Application of the Semantic Learning Approach in the Feasibility Studies Preparation Training Process. *Information Systems Management*, *26*(3), 231–240.
Georgakopoulos, D., Hornick, M., & Sheth, A. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, *3*(2), 119–153.
Gil, Y. (2007). Workflow Composition: Semantic Representations for Flexible Automation. In *Workflows for e-Science* (pp. 244–257). London: Springer London.
Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., … Myers, J. (2007). Examining the Challenges of Scientific Workflows. *Computer*, *40*(12), 24–32.
Goderis, A., Sattler, U., Lord, P., & Goble, C. (2005). Seven Bottlenecks to Workflow Reuse and Repurposing. In *Internal Seamantic Web Conference* (pp. 323–337). Galway: Springer, Berlin, Heidelberg.
Goodchild, M. F. (1992). Geographical information science. *International Journal of Geographical Information Systems*, *6*(1), 31–45.
Goodchild, M. F. (2010). Twenty years of progress: GIScience in 2010. *Journal of Spatial Information Science*, *2010*(1), 3–20.
Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, *43*(5–6), 907–928.
Guarino, N. (1997). Understanding, building and using ontologies. *International Journal of Human-Computer Studies*, *46*(2–3), 293–310.
Guarino, N. (1998). Formal ontology in information systems : proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy. In *International Conference on Principles of Knowledge Representation and Reasoning* (pp. 3–18). Amsterdam: IOS Press.
Hofer, B., Mäs, S., Brauner, J., & Bernard, L. (2017). Towards a knowledge base to support geoprocessing workflow development. *International Journal of Geographical Information Science*, *31*(4), 694–716.
Ison, J., Kalas, M., Jonassen, I., Bolser, D., Uludag, M., McWilliam, H., … Rice, P. (2013). EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, *29*(10), 1325–1332.
Kasalica, V., & Lamprecht, A.-L. (2018). Automated composition of scientific workflows: A case study on geographic data manipulation. In *2018 IEEE 14th International Conference on e-Science* (pp. 362–363). IEEE.
Kasalica, V., & Lamprecht, A.-L. (2019). Workflow Discovery Through Semantic Constraints: A Geovisualization Case Study. In *International conference on computational science and its applications* (pp. 473–488). Cham: Springer. https://doi.org/10.1007/978-3-030-24302-9_34
Kona, S., Bansal, A., Blake, M. B., & Gupta, G. (2008). Generalized Semantics-Based Service Composition. In *2008 IEEE International Conference on Web Services* (pp. 219–227). IEEE.
Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, *26*(12), 2267–2276.
Lamprecht, A.-L., Margaria, T., & Steffen, B. (2009). Bio-jETI: a framework for semantics-based service composition. *BMC Bioinformatics*, *10*(Suppl 10), S8.
Lamprecht, A.-L., Naujokat, S., Margaria, T., & Steffen, B. (2010). Synthesis-Based Loose Programming. In *2010 Seventh International Conference on the Quality of Information and Communications Technology* (pp. 262–267). IEEE.
Lamprecht, A.-L., Naujokat, S., Steffen, B., & Margaria, T. (2010). Constraint-Guided Workflow Composition Based on the EDAM Ontology.
Lehmann, J., Athanasiou, S., Both, A., Buhmann, L., Garcia Rojas, A., Giannopoulos, G., … Wetphal, P. (2015). *The geoknow handbook* (1st ed.). Leipzig: Institute of Applied Informatics.
Lemmens, R., Wytzisk, A., By, R. d., Granell, C., Gould, M., & van Oosterom, P. (2006). Integrating Semantic and Syntactic Descriptions to Chain Geographic Services. *IEEE Internet Computing*, *10*(5), 42–52.
Lin, C., Lu, S., Lai, Z., Chebotko, A., Fei, X., Hua, J., & Fotouhi, F. (2008). Service-Oriented Architecture for VIEW: A Visual Scientific Workflow Management System. In *2008 IEEE International Conference on Services Computing* (pp. 335–342). IEEE.
Ludäscher, B., Weske, M., McPhillips, T., & Bowers, S. (2009). Scientific Workflows: Business as Usual? (1st ed., pp. 31–47). Berlin: Springer.

Luscher, P., Burghard, D., & Weibel, R. (2007). Ontology-driven enrichtment of spatial databases. In *10th ICA Workshop on Generalisation and Multiple Representations* (p. 13).
Lutz, M. (2007). Ontology-Based Descriptions for Semantic Discovery and Composition of Geoprocessing Services. *GeoInformatica*, *11*(1), 1–36.
Maffini, G. (1987). Raster versus vector data encoding and handling: a commentary. *Photogrammetric Engineering and Remote Sensing*, *53*(10), 1397–1398.
Maliene, V., Grigonis, V., Palevičius, V., & Griffiths, S. (2011). Geographic information system: Old principles with new capabilities. *URBAN DESIGN International*, *16*(1), 1–6.
Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., … Sycara, K. (2005). Bringing Semantics to Web Services: The OWL-S Approach (pp. 26–42). Springer, Berlin, Heidelberg.
McCready, S. (1992). There is more than one kind of workflow software. *Computerworld*, *2*, 86–90.
Medina-Mora, R., Winograd, T., Flores, R., & Flores, F. (1993). The action workflow approach to workflow management technology. *The Information Society*, *9*(4), 391–404.
Munir, K., & Sheraz Anjum, M. (2018). The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, *14*(2), 116–126.
Oliveira, W., de Oliveira, D., & Braganholo, V. (2015). Experiencing PROV-Wf for Provenance Interoperability in SWfMSs. Springer, Cham.
OSGEO. (n.d.). wxGUI Graphical Modeler.
Rusinkiewicz, M., & Sheth, A. (1995). Modern database systems : Specification and execution of transactional workflows. In *Modern database systems* (1st ed., Vol. 1st, pp. 592–620). New York: ACM Press.
Scheider, S, & Tomko, M. (2016). Knowing Whether Spatio-Temporal Analysis Procedures Are Applicable to Datasets. *Frontiers in Artificial Intelligence and Applications*, *283*, 67–80.
Scheider, Simon. (2019). Semanitc data type signatures for representing spatial core concepts in GIS operation on spatial layers. Utrecht: Utrecht University.
Son, J. H., Sun Kim, J., & Ho Kim, M. (2005). Extracting the workflow critical path from the extended well-formed workflow schema. *Journal of Computer and System Sciences*, *70*(1), 86–106.
Sonntag, M., Karastoyanova, D., & Deelman, E. (2010). Bridging the Gap between Business and Scientific Workflows: Humans in the Loop of Scientific Workflows. In *2010 IEEE Sixth International Conference on e-Science* (pp. 206–213). IEEE.
Stasch, C., Scheider, S., Pebesma, E., & Kuhn, W. (2014). Meaningful spatial prediction and aggregation. *Environmental Modelling & Software*, *51*, 149–165.
Steffen, B., Margaria, T., & Freitag, B. (1993). *Module Configuration by Minimal Model Construction*. Passau.
Stevens, S. S. (1946). On the Theory of Scales of Measurement. *Science (New York, N.Y.)*, *103*(2684), 677–680.
Tomlinson, R. F. (1969). A Geographic Information System for Regional Planning. *Journal of Geography (Chigaku Zasshi)*, *78*(1), 45–48.
Ubels, S. (2018). *Understanding abstract geo-information workflows and converting them to executable workflows using Semantic Web Technologies*. University of Twente.
van der Aalst, W. M. P. (1998). The application of petri nets to worklfow management. *Journal of Circuits, Systems and Computers*, *08*(01), 21–66.
van der Aalst, W. M. P., Hirnschall, A., & Verbeek, H. M. W. (2002). An Alternative Way to Analyze Workflow Graphs. In *International conference on advanced information systems engineering* (pp. 535–552). berlin: Springer.
VertiGIS. (n.d.). Geocortex Workflow.
Vouk, M. A., & Singh, M. P. (1996). Quality of Service and Scientific Workflows. North Carolina State University at Raleigh.
Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., … Goble, C. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, *41*(W1), W557–W561.
Workflow Management Coalition. (1999). Gloassary, Terminology and Glossary.
Yue, P., Di, L., Yang, W., Yu, G., & Zhao, P. (2007). Semantics-based automatic composition of geospatial Web service chains. *Computers & Geosciences*, *33*(5), 649–665.
Zhang, J., Tan, W., Alexander, J., Foster, I., & Madduri, R. (2011). Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse. In *2011 IEEE International Conference on Services Computing* (pp. 48–55). IEEE.
Zion Market Research. (2019). *Workflow Management System Market by Deployment (On-Premises and Cloud), by Component (Software and Services), and by End-Use Industry (Retail, BFSI, IT*

& Telecom, Healthcare, Government Sector, Transportation, Manufacturing, and Others): Global Indus. New York.
Zúñiga, G. L. (2001). Ontology: its transformation from philosophy to information systems. In *Proceedings of the international conference on Formal Ontology in Information Systems* (Vol. 2001, pp. 187–197). New York, New York, USA: ACM Press.

10 Appendix A

Entire software package + results and operationalizations + ontologies + tool descriptions

https://drive.google.com/open?id=1sYtxgiSWpi_Dl19w4NvMVdV8GJ_zldgm

Flat APE prototype

https://drive.google.com/open?id=18VwCh3mbnzKfVahjW0nuLcruhyBzpbmL

Full APE prototype

https://drive.google.com/open?id=1SrJ4WNQ5YbjR9UqtxCUglZjYY5ofh2Ou

Results and operationalization for each question

https://drive.google.com/open?id=1CLGopcTFp09PfQ1jPI5Ii0zcNLbe393e

Script for flattening tool description and ontology into a flattened ontology and tool description

https://drive.google.com/open?id=1qfoRziJxBbLdh__YRVUaJ5G72JDSsYTG

Semantic data types ontology + Flattened semantic data types ontology

https://drive.google.com/open?id=1x-EonoDWHhavT75BStBIiHb3MyvTeCMu

All tool descriptions

https://drive.google.com/open?id=16XGk4w4s11vDig-SK_TyzWYBeVA9OumQ