Comparison of City Performances Through Statistical Linked Data Exploration

Claudia Diamantini, Domenico Potena, and Emanuele Storti(✉)

Dipartimento di Ingegneria dell'Informazione, Universita Politecnica delle Marche, via Brecce Bianche, 60131 Ancona, Italy
{c.diamantini,d.potena,e.storti}@univpm.it

Abstract. The capability to perform comparisons of city performances can be an important guide for stakeholders to detect strengths and weaknesses and to set up strategies for future urban development. Today, the rise of the Open Data culture in public administrations is leading to a larger availability of statistical datasets in machine-readable formats, e.g. the RDF Data Cube. Although these allow easier data access and consumption, appropriate evaluation mechanisms are still needed to perform proper comparisons, together with an explicit representation of how statistical indicators are calculated. In this work, we discuss an approach for analysis and comparison of statistical Linked Data which is based on the formal and mathematical representation of performance indicators. Relying on this knowledge model, a set of logic-based services are able to support novel typologies of comparison of different resources.

Keywords: Statistical datasets · Performance indicators · Logic reasoning · Smart cities

1 Introduction

Performance monitoring is becoming a more and more important tool in planning and assessing efficiency and effectiveness of services and infrastructures in urban contexts. This increasing attention is witnessed also by projects (e.g., CITYKeys[1]), standards (e.g., ISO 37120:2014, ISO/TS 37151:2015) and initiatives at international level (e.g., Green Digital Charter[2], European Smart City Index) which push forward the definition of shared frameworks for performance measurement at city level. Statistical data are capable of more effectively guiding municipal administrations in the decision making process and foster civic participation. They can also impact on the capability to attract private investments, which may be stimulated by opportunities that are made explicit by quantitative evidences and comparisons between different municipalities. Also thanks to the rise of the Open Data culture in public administrations, today statistical datasets are more frequently available and accessible in machine-readable formats. This

[1] http://citykeys-project.eu/.
[2] http://www.greendigitalcharter.eu/.

A. Longo et al. (Eds.): IISSC 2017/CN4IoT 2017, LNICST 189, pp. 3–12, 2018.
https://doi.org/10.1007/978-3-319-67636-4_1

enables the possibility to adapt to cities methods and solutions exploited for decades in enterprise contexts to assess the achievement of business objectives.

A recent trend in this respect is to publish statistical data according to the RDF Data Cube vocabulary[3], a W3C standard for the representation of statistical datasets in the web. This format follows the Linked Data approach and conceptually resorts to the multidimensional model [1] adopted in enterprise contexts for Data Warehouses, as observed values (e.g., level of CO2) are organized along a group of dimensions (e.g., time and place, as the measure is taken daily and each value refers to a specific monitoring station in the city), together with associated metadata. The publication of performance datasets according to the Linked Data approach allows to reduce heterogeneity, as measures from different datasets may be aligned with the same definition of indicator. However, besides it is a concrete step towards an easier access and interoperability among different datasets, appropriate mechanisms to evaluate and compare performances are yet to come. One of the main reasons is related to the lack of a shared, explicit and unambiguous way to define indicators. Indeed, no meaningful comparisons of performance can be made without the awareness of how indicators are calculated. To make an example, if we were interested in comparing the ratios of delayed trips in two public transportation systems, we would require to understand how such ratios are actually computed, e.g. if the first summed up trips made by trams and bus, while the second considered only the latter, the risk would be to derive wrong consequences and take uneffective decisions.

With the purpose to address the above mentioned issues, in this paper we propose a logic-based approach to enable the comparison of datasets published by different municipalities as Linked Open Data. The approach is based on the formal, ontological representation of indicators together with their calculation formulas. Measures are then declaratively mapped to these definitions in order to express their semantics. In this way, the ontology serves as a reference library of indicators that can be incrementally extended. Finally, a set of services, built on the top of the model and exploiting reasoning functions, offers functionalities to determine if two datasets are comparable, and to what extent. The rest of this work is organised as follows: next Section briefly presents a case study that will be used throughout the paper. In Sect. 3 we discuss an ontology to formally represent statistical indicators with their calculation formulas, and we introduce the representation of statistical data according to the RDF Data Cube vocabulary. These models and languages are exploited in Sect. 4 to provide a set of services aimed to support analysis and comparisons of Linked datasets. Finally, in Sect. 5 we provide conclusions and outline future work.

2 Case Study: Bike Sharing Services

Alternative, more sustainable and energy-efficient forms of urban mobility are among the major goals of many smart cities initiatives, both at national and

[3] https://www.w3.org/TR/vocab-data-cube/.

international level. Several cities have already started to share data about transport services with a larger audience as open data. In the following, we introduce a case study focusing on bike sharing services provided by two municipalities, City_A and City_B. The example is a simplified version of actual datasets published by a set of US municipalities including New York[4], Chattanooga[5] and many others. In details, let us suppose that each municipality provides a library of datasets, as follows:

- City_A measures the *total distance* (in miles) of bike rides, aggregated with respect to user type (residents/tourists) and time, and the *population* through dimension time.
- City_B measures the *total distance* of bike rides for residents and the *total distance* of rides for tourists aggregated with respect to time; it also measures the *population* with respect to time.

3 Data and Knowledge Layer

In this Section we discuss the models and languages that are used in this work to represent performance indicators (Subsect. 3.1) and datasets (Subsect. 3.2) according to the Linked Data approach.

3.1 Modeling of Performance Indicators

Reference libraries of indicators, e.g. VRM or SCOR [2], have been used as a reference for a long time, especially for performance management in the enterprise domain. More recently, the interest in the systematisation and organisation of the huge amount of existing PIs is witnessed by many collections of indicators proposed by public bodies or specific projects (e.g., [3] in the context of smart cities). Most of them, however, are not machine-readable and lack formal semantics. Several work in the Literature tried to fill this gap, proposing ontologies for declarative definition of indicators (e.g., [4,5]), even though in most cases they do not include an explicit representation of formulas capable to describe how to calculate composite indicators from others. On the other hand, the representation of mathematical expressions in computer systems has been investigated for a variety of tasks like information sharing and automatic calculation. The most notable and recent examples are MathML and OpenMath [6], mainly targeted to represent formulas in the web.

In the context of this work, indicators and their formulas are formally represented in KPIOnto, an ontology conceptually relying on the multidimensional model and originally conceived as a knowledge base for a performance monitoring framework for highly distributed enterprise environments [7]. As reported in Fig. 1, within the classes defined in KPIOnto[6] for the purpose of this work we focus on the following:

[4] https://www.citibikenyc.com/system-data.
[5] https://data.chattlibrary.org/.
[6] Full ontology specification is available online at http://w3id.org/kpionto.

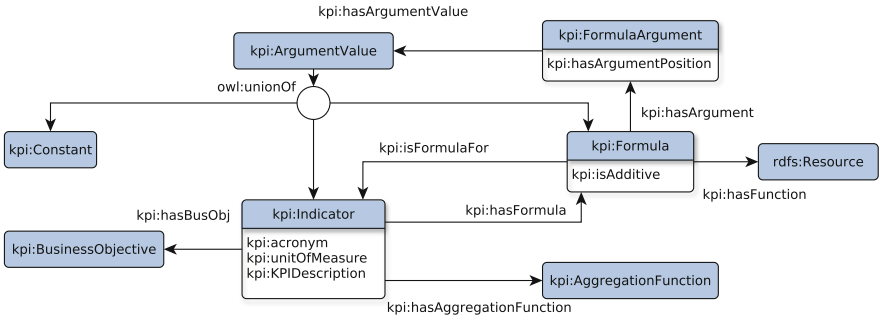

Fig. 1. KPIOnto: main classes and properties.

- `Indicator`, that represents a quantitative metric (or measure) together with a set of properties, e.g. one or more compatible dimensions, a formula, a unit of measurement, a business objective and an aggregation function.
- `Formula`, that formally represents an indicator as a function of other indicators. An indicator can indeed be either atomic or compound, built by combining several other indicators through a mathematical expression. Operators are represented as defined by OpenMath [6], an extensible XML-based standard for representing the semantics of mathematical objects. On the other hand, operands can be defined as indicators, constants or, recursively, as other formulas.

As regards the case study, we define indicators *Distance* and *TotalPopulation* for City$_A$, *Distance_Tourists* and *Distance_Citizens* for City$_B$.

3.2 Representation of Statistical Datasets

Several standards for representation of statistical data on the web have been adopted in the past with the purpose to improve their interpretation and interoperability, e.g. SDMX (Statistical data and metadata exchange) [8] and DDI (Data Documentation Initiative)[7] just to mention the most notable examples. In the last years, in order to rely on more flexible and general solutions for publishing statistical datasets on the web, several RDF vocabularies have been proposed in the Literature. To address the limits of early approaches (e.g., the capability to properly represent dimensions, attributes and measures or to group together data values sharing the same structure), the Data Cube vocabulary (QB) [9], was proposed by W3C to publish statistical data on the web as RDF following the Linked Data principles. According to the multidimensional model, the QB language defines the schema of a cube as a set of dimensions, attributes and measures through the corresponding classes qb:DimensionProperty, qb:AttributeProperty and qb:MeasureProperty. Data instances are represented in QB as a set of `qb:Observation`s, that can be optionally grouped in subsets named Slices.

[7] http://www.ddialliance.org/.

To make an example about the case study of Sect. 2, the data structure of the first dataset for City$_A$ includes the following components:

- `cityA:Distance`, a qb:MeasureProperty for the total distance;
- `sdmx-dimension:timePeriod`, a qb:DimensionProperty for the time of the observation;
- `cityA:userType`, a qb:DimensionProperty for the user type.

Please note that the prefix *"qb:"* stands for the specification of the Data Cube vocabulary[8], *"sdmx-dimension:"* points to the SDMX vocabulary for standard dimensions[9], while *"cityA:"* is a custom namespace for describing measures, dimensions and members of the dataset for City$_A$. In order to make datasets comparable, the approach we take in this work is to rely on KPIOnto as reference vocabulary to define indicators. As such, instances of MeasureProperty as defined in Data Cube datasets have to be semantically aligned with instances of `kpi:Indicator`, through a RDF property as follows: `cityA:Distance rdfs:isDefinedBy kpi:TotalDistance.` In this way, the semantics of the measure `Distance`, as used by City$_A$, will be provided by the corresponding concept of `TotalDistance` in KPIOnto.

For what concerns observations, i.e. data values, we report an example about the measure Distance for City$_A$, for time *December, 5th 2016* (time dimension), and user type *citizen*:

```
cityA:obs001 a qb:Observation;
          sdmx-dimension:timePeriod"2016-12-05"^^xsd:date;
          cityA:userType cityA:resident;
          cityA:Distance 80214;
          qb:dataSet cityA:dataset1.
```

4 Services for Analysis and Comparison of Datasets

In this Section we discuss a set of services that are aimed to support analysis and comparisons of statistical datasets. As depicted in Fig. 2, services are built on top of the Data/knowledge layer, while access to datasets is performed through SPARQL queries over corresponding endpoints. A single endpoint may serve a library of datasets belonging to the same municipality. In the first subsection, we introduce the reasoning framework, which comprises basic logical functions for formula manipulation, on which the others rely, while in Subsect. 4.2 we focus on services for dataset analysis and comparison. Further services are available in the framework and devised to support indicator management, which enable the definition of new indicators and exploration of indicator structures. For lack of space, we refer the interested reader to a previous work of ours discussing in detail these services [7].

[8] https://www.w3.org/TR/vocab-data-cube/.
[9] http://purl.org/linked-data/sdmx/2009/dimension.

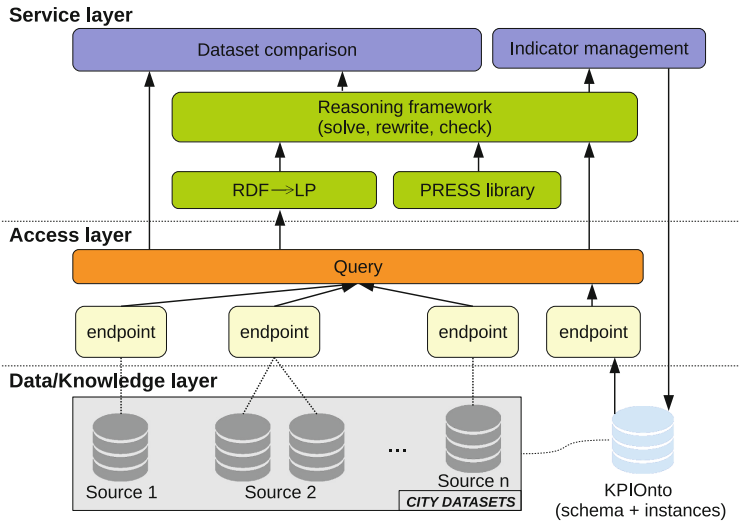

Fig. 2. Architecture of the framework.

In the following, we will refer to the example introduced in Sect. 2. After the definition of the indicators, we assume these mappings have been defined between datasets' measures and KPIOnto indicators:

cityA:Distance rdfs:isDefinedBy kpi:Distance.
cityA:Population rdfs:isDefinedBy kpi:TotalPopulation.
cityB:Distance_Residents rdfs:isDefinedBy kpi:Distance_Citizens.
cityB:Distance_Tourists rdfs:isDefinedBy kpi:Distance_Tourists.
cityB:Population rdfs:isDefinedBy kpi:TotalPopulation.

Let us suppose also that the formula *kpi:Distance=kpi:Distance_Citizens + kpi:Distance_Tourists* is defined by the user to state that the indicator can be calculated as the summation of the two types of distances. Moreover, let us suppose that the user is interested to better understand the inclination of the local population in using bike sharing services. For this reason, the user will define a further indicator *AvgDistancePerCitizen*, with formula $\frac{kpi:Distance\_Population}{kpi:TotalPopulation}$, which measures the distance covered on average by residents. As for dimensions, for simplicity we assume that the time dimension is defined as *sdmx-dimension:timePeriod* in all datasets[10].

4.1 Reasoning on Indicator Formulas

A set of logic-based functionalities are defined to enable an easy and transparent management of the indicator formulas defined according to KPIOnto. We refer in particular to Prolog as logic language for its versatility, capability of symbolic manipulation as well as for the wide availability of well-documented reasoners

[10] Please note that *owl:sameAs* links can be defined between different definitions of the same dimension for interoperability purposes.

and tools. Indicators formulas are thus translated to Prolog facts, and a set of custom reasoning functions is defined to support common formula manipulations exploited by services discussed in the next subsections, among which:

- `solve_equation(eq,indicator)`, which is capable to solve the equation *eq* with respect to variable *indicator*;
- `get_formulas(ind)`, which returns all possible rewritings of the formula for a given indicator; the predicate is capable to manipulate the whole set of formulas and find alternative rewritings by applying mathematical axioms (e.g., commutativity, associativity, distributivity and properties of equality). This also allows to derive a formula for an atomic indicator, e.g. $Distance\_Citizens{=}AvgDistancePerCitizen * TotalPopulation$ is inferred by solving the *AvgDistancePerCitizen* formula w.r.t. the variable *Distance_Citizens*.
- `derive_all_indicators(measures)`, which returns a list of all the indicators that can be calculated starting from those provided in input. The function exploits `get_formulas` to decompose all the available indicators in any possible way, and each of these rewriting is checked against the list in input. If there is a match, the solution is returned in output, e.g. if in input we have $\{Distance\_Citizens, TotalPopulation\}$, the function returns the list $\{Distance\_Citizens, TotalPopulation, AvgDistancePerCitizen\}$, as the last indicator can be calculated from the others through the formula $\frac{Distance\_Citizens}{TotalPopulation}$.

Such functionalities are built upon PRESS (PRolog Equation Solving System) [37], a library of predicates formalizing algebra in Logic Programming, which are capable to manipulate formulas according to mathematical axioms. We refer interested readers to previous work specifically focused on this reasoning framework [7,10], which includes also computational analyses on efficiency of these logic functions.

4.2 Dataset Comparison and Evaluation

In order to enable performance analyses across multiple datasets, belonging to the same or different libraries (i.e. to different municipalities), a preliminary evaluation must be performed in order to verify whether they are comparable and to what extent. The services discussed in this subsection are hence devised to assess comparability taking into account both measures and dimensions. In detail, we define two datasets comparable at *schema level* if their schemas (i.e. the DataStructure in the Data Cube model) have a non-empty intersection in terms of measures and dimensions. Hereafter, we consider two different cases, namely how to determine the comparable measures of two given datasets and, in turn, how to determine which datasets are comparable with a given indicator.

Evaluation of comparable measures and dimensions. Given two libraries of datasets and their endpoints, the service `get_common_indicators` retrieves available and derivable indicators from each dataset and compares them.

```
1  get_common_indicators(endpoint1,endpoint2):
2       I1= get_all_indicators (endpoint1)
3       I2= get_all_indicators (endpoint2)
4       return I1∩I2
5
6   get_all_indicators (endpoint):
7      measures←get_measures(endpoint)
8      ∀ m ∈ measures:
9         indicators←get_ind_from_mea(m,endpoint)
10     availableIndicators ←derive_all_indicators(indicators)
11     return  availableIndicators
```

In detail, the service `get_all_indicators` firstly retrieves all the MeasureProperties from each library of datasets by executing this SPARQL query to the corresponding endpoint (line 7):

```
SELECT ?m ?dataset
WHERE {?dataset qb:structure ?s.
       ?s qb:component ?c.
       ?c qb:measure ?m.}
```

Then, for each measure m the service gets the corresponding KPIOnto indicator (see line 9) through the query:

```
SELECT ?ind
WHERE {<m> rdfs:isDefinedBy ?ind.}
```

Finally, the service calls the logic function `derive_all_indicators` (line 5), which is capable to derive all indicators that can be calculated from the available measures through mathematical manipulation. Once compatible measures are found, a similar check is made with respect to dimensions, i.e. firstly the dimensions related to each compatible measure are retrieved, and finally such sets are compared in order to find the common subset.

Let us consider the comparison of libraries City_A and City_B. Indicators from the former are I_A ={*kpi:Distance*, *kpi:TotalPopulation*}. On the other hand, City_B includes indicators {*kpi:Distance_Citizens*, *kpi:Distance_Tourists*, *kpi:TotalPopulation*}. By using the logical predicate `derive_all_indicators` on this last set, the reasoner infers that I_B={*kpi:Distance_Citizens*, *kpi:Distance_Tourists*, *kpi:TotalPopulation*, *kpi:Distance*, *kpi:AvgDistancePerCitizen*}. Indeed, the last two indicators can be calculated from *kpi:Distance*=*kpi:Distance_Citizens* + *kpi:Distance_Tourists* and *kpi:AvgDistancePerCitizen*=$\frac{kpi:Distance\_Citizens}{kpi:TotalPopulation}$. As a conclusion, the two libraries share the indicator set $I_A \cap I_B$ = {*kpi:Distance*, *kpi:TotalPopulation*}. Please also note that without the explicit representation of formulas and logic reasoning on their structure, only TotalPopulation would have been obtained. Both indicators are comparable only through dimension *sdmx-dimension:timePeriod*. In particular, *kpi:Distance* is measured by City_A also along the user type dimension. This means that some manipulation (i.e. aggregation) must be performed on City_A values before the indicator can be actually used for comparisons.

Search for datasets measuring a given indicator. Given an indicator, a list of dataset libraries and the corresponding endpoints, the service returns those datasets in which the indicator at hand is available or from which it can be calculated. The approach relies on the exploitation of KPIOnto definitions of indicator formulas, and Logic Programming functions capable to manipulate them. Firstly, for each library the following query is performed to determine if the indicator is explicitly provided by some dataset:

```
SELECT ?m ?dataset
WHERE {?dataset qb:structure ?s.
       ?s qb:component ?c.
       ?c qb:measure ?m.
       ?m rdfs:isDefinedBy <indicator>.}
```

In case the response is negative, the service (1) derives all possible alternative ways to calculate the indicator through the logic function `get_formulas`. Then (2) it searches into the libraries for combinations of datasets including those measures. Let us suppose to search for the indicator *kpi:AvgDistancePerCitizen* in datasets of City_A and City_B. Given that such an indicator is not directly available in any City, the service calls the `get_formulas` predicate, which returns two solutions, i.e. $s_1 = \frac{kpi:Distance\_Citizens}{kpi:TotalPopulation}$ and $s_2 = \frac{(kpi:Distance - kpi:Distance\_Tourists)}{kpi:TotalPopulation}$. Please note that this last is a rewriting of s_1, obtained by solving the formula *kpi:Distance=kpi:Distance_Citizens+kpi:Distance_Tourists*, with respect to the variable *kpi:Distance_Citizens*. At step 2, each solution is tested against the libraries. Checking a solution means to verify, through queries like the one above, that every operand of the solution is measured by a dataset in the library at hand. As for City_B, solution s_1 can be used, as it includes both measure `cityB:Distance_Residents` (that corresponds to kpi:Distance_Citizens) and `cityB:Population` (corresponding to kpi:TotalPopulation). As for City_A, instead, no solution is valid, as it lacks both `kpi:Distance_Citizens` (needed by s_1) and `kpi:Distance_Tourists` (required by s_2):

Library	Formula	Measures
City_A	$\frac{kpi:Distance\_Citizens}{kpi:TotalPopulation}$	× kpi:Distance_Citizens
		✓ kpi:TotalPopulation←cityA:Population
City_A	$\frac{kpi:Distance - kpi:Distance\_Tourists}{kpi:TotalPopulation}$	✓ kpi:Distance←cityA:Distance
		× kpi:Distance_Tourists
		✓ kpi:TotalPopulation←cityA:Population
City_B	$\frac{kpi:Distance\_Citizens}{kpi:TotalPopulation}$	✓ kpi:Distance_Citizens←CityB:Distance_Residents
		✓ kpi:TotalPopulation←CityB:Population

The service reports as output, for each solution, the used formula and the available measures, specifying the corresponding mappings between the KPIOnto indicators and the specific MeasureProperty names, according to rdfs:isDefinedBy properties. Output includes also partial solutions (like the first two), in order to make users be aware of which specific measures are missing.

5 Discussion and Future Work

In this work, we discussed a knowledge-based approach to the representation and the comparisons of city performances referring to different urban settings, published as Linked Data and monitored through specific indicators. So far, KPIOnto has been used in a variety of applications, ranging from performance monitoring in the context of collaborative organizations, to serving as a knowledge model to support ontology-based data exploration of indicators [10].

As for RDF Data Cube, we note that some limitations make it not perfectly suited to a variety of real applications, mainly for its lack of proper support for the representation of dimension hierarchies. Some possible extensions have been already proposed in the Literature to overcome such limits (e.g., QB4OLAP [11]), that will be considered in future work. Furthermore, we are investigating to provide a more fine-grained comparison between datasets by means of a more comprehensive notion of comparability taking into account both schema and instance levels of datasets.

References

1. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd edn. Wiley, New York (2002)
2. Supply Chain Council: Supply chain operations reference model. SCC (2008)
3. Bosch, P., Jongeneel, S., Rovers, V., Neumann, H.M., Airaksinen, M., Huovila, A.: Deliverable 1.4. smart city kpis and related methodology. Technical report, CITYKeys (2016)
4. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. Softw. Syst. Model. **13**(3), 1015–1041 (2014)
5. del Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the definition and design-time analysis of process performance indicators. Inf. Syst. **38**(4), 470–490 (2013)
6. Buswell, S., Caprotti, O., Carlisle, D.P., Dewar, M.C., Gaetano, M., Kohlhase, M.: The open math standard. Technical report, version 2.0, The Open Math Society, 2004 (2004). http://www.openmath.org/standard/om20
7. Diamantini, C., Potena, D., Storti, E.: SemPI: a semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators. Future Gener. Comput. Syst. **54**, 352–365 (2015)
8. SDMX: SDMX technical specification. Technical report (2013)
9. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF data cube vocabulary. Technical report, World Wide Web Consortium (2014)
10. Diamantini, C., Potena, D., Storti, E.: Extended drill-down operator: digging into the structure of performance indicators. Concurr. Comput. Pract. Exper. **28**(15), 3948–3968 (2016)
11. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 45–56. Springer, Cham (2014). doi:10.1007/978-3-319-10160-6_5