New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics Volume II: Applications

Editors

Krassimir T. Atanassov
Władysław Homenda
Olgierd Hryniewicz
Janusz Kacprzyk
Maciej Krawczak
Zbigniew Nahorski
Eulalia Szmidt
Sławomir Zadrożny

SRI PAS  **IBS PAN**

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics Volume II: Applications

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics
Volume II: Applications

Systems Research Institute
Polish Academy of Sciences

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics
Volume II: Applications

Editors

Krassimir T. Atanassov
Władysław Homenda
Olgierd Hryniewicz
Janusz Kacprzyk
Maciej Krawczak
Zbigniew Nahorski
Eulalia Szmidt
Sławomir Zadrożny

IBS PAN  **SRI PAS**

Systems Research Institute
Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland
www.ibspan.waw.pl

ISBN 83-894-7541-3

Dedicated to Professor Beloslav Riečan on his 75th anniversary

Matching methods for semantic annotation-based XML document transformations

Marcin Szymczak[1] **and Julius Köpke**[2]

[1]PhD Studies, Systems Research Institute, Polish Academy of Sciences,
Newelska 6, Warsaw, Poland
szymczak.m@gmail.com

[2]Institute for Informatics Systems, University of Klagenfurt,
Universitaetsstr. 65-67, Klagenfurt, Austria

Abstract

The widely adoption of XML as an exchange format provides syntactic interoperability between different communication partners. However, in a heterogenous communication environment it is common, that different partners use different schemas for the transferred messages. This requires transformations between the different partners. The creation of such transformations is typically manual work, which is an error-prone and tedious task. The main problem to achieve a mapping between heterogenous schemas is the matching of the schemas according to their semantics. Semantic annotations can be used to explicitly define the semantics and can therefore, support the creation of schema matchings. In this paper, we will present a schema matching and mapping solution that maps heterogenous schemas based on their semantic annotations and allows the automatic creation of transformation scripts.

Keywords: interoperability, XML, transformation, semantic annotation, XML-Schema matching.

1 Introduction

Semantic annotations of XML-Schema allow the semantic interpretation of the schema elements. This information can be used to create mappings between

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume II: Applications *(K.T. Atanassow, W. Homenda, O. Hryniewicz, J. Kacprzyk, M. Krawczak, Z. Nahorski, E. Szmidt, S. Zadrożny, Eds.), IBS PAN - SRI PAS, Warsaw, 2012.*

different annotated schemas. Such mappings can be used to generate transformation scripts (e.g. XSLT[12]) that actually transform instance data from the representation of the source schema to the representation of the target schema. The semantic annotation of XML-Schema is addressed in the W3C recommendation SAWSDL[11]. It adds two additional properties to XML-Schema elements: Model-References and Schema Mappings. Model References assign concepts of some semantic model to elements or types of a schema, while Schema Mappings are references to scripts that transform data from the instance documents to instances of some semantic model (lifting) or back to XML (lowering). While lifting and lowering mappings are already widely used the Model-References are used in a much lesser degree. They are limited to direct references to concepts, which can be problematic, when schemas should be annotated with a general reference ontology. To solve this issue an extended annotation method based on SAWSDL was presented in [6]. It describes the schema elements in form of annotation path expressions that consist of concepts and properties of a reference ontology. Those annotation path expressions can directly be added to the schema documents. In order to create semantic matchings they can automatically be transformed to OWL[5] concepts. Thus, the semantic matching can be realized with reasoning support and knowledge from the reference ontology. In this paper we will present a complete matching and mapping solution that is based on the proposed annotation method.

1.1 Annotation Method

The annotations that are used for this approach are based on the annotation method described in [6]. An annotation is a path expression that consists of a sequence of steps. Each step may refer to a concept or a property in the reference ontology. The first step must refer to a concept, the last step may refer to a concept or a data-type property. Two concept steps may only be connected by an object-property-step. An example for such an annotation path p is */Order/hasBillingAddress/Address/hasZipCode/ZipCode*. This path could be used to annotate a zip-code element of an XML-Schema for order documents. Such an annotation path expression can directly be transformed to an OWL concept ($p.concept$). The corresponding concept needs only to exist at runtime of the matching engine and has the string representation of the annotation path as its URI. An example annotation concept for $p.concept$ is shown in listing 1.

```
Class: /Order/hasBillingAddress/Address/hasZipCode/ZipCode
EquivalentClasses(
      ConceptAnnotation and ZipCode and inv
        (hasZipCode) some
           (Address and inv (hasBillingAddress) some (Order)
        ))
```

Listing 1: Representation of a Concept Annotation path in OWL

1.2 Schema-Matching

Schema matching approaches that are not based on semantic annotations [9] try to find correspondences between the elements of the schema by exploiting different dimensions of the schema. Such dimensions can for example be attribute names, structural similarities or constraints over data-types. Typical schema-level match solutions such as [1] use multiple matchers for the different dimensions in order to achieve good results. Each matcher returns confidence values that represent the relatedness of the elements. This means each matcher produces a $n \times m$ matrix that relates each element of the source schema to each element of the target schema with confidence values. Finally the correspondence values of the different matchers are combined and elements with the highest overall scores are considered to be the best matches. Thus the result is a $n \times m$ matrix.
Such an approach implies that local cardinalities other than 1:1 are typically not addressed. For example a source schema might contain two fields $firstname$ and $lastname$, while the target schema only contains one element $name$. This requires a matching expression of the form *(<firstname,' ',lastname>, name, concat)*. Thus, a $n : 1$ expression with an arbitrary function (concat). Such an expression cannot be derived without additional knowledge. In addition the search space explodes since any subset of elements from the source schema can be related to any subset of elements from the target schema with any arbitrary function. In this paper we will show how a practical matching system can be implemented that is capable to efficiently match annotated source and target schemas including complex (non *1:1*) matches. The presented solution generates an output format that can directly be used to automatically create transformation scripts.

1.3 The matching approach

The matching approach was implemented in form of a proof of concept for the annotation method. The implementation details can be found in [10]. On the one hand any composite matching approach that allows the definition of custom

matchers can also be extended to match schemas that are annotated with the used annotation method. On the other hand preliminary tests have shown, that a matching system that is solely based on our annotation method can already produce good results and we needed a system that supports complex matches where local cardinalities of the matches are not restricted to *1:1*. We therefore decided to match the schemas based on the semantic annotations in first place. Limited structural matching is only used if ambiguities of the annotations exist.

2 Semantic level matching

The matching approach is based on the observation that a schema does typically not contain more than one element with the same semantics. We have therefore optimized the matching process by first matching the semantic annotations. We call this phase semantic level matching. The first step is to extract the semantic annotations from the source and target schemas. We assume that in an annotated schema each node is extended with a *sawsdl:model-reference* attribute that contains the semantic annotation. The extraction step also includes annotation path concatenation[6] which is required, when named types or elements are reused. The result of this step are two sets of annotation path expressions: A_s with all annotations from the source schema and A_t with all annotations from the target schema. Afterwards the corresponding annotation concepts for A_s and A_t are created and added to the ontology. Each created concept has the original annotation string as its URI. Both sets are distinguished with a different prefix (S and T). Finally the ontology is classified by a standard reasoner.
The goal of the semantic matching phase is to find the best possible annotation from the target schema for each annotation from the source schema. This is realized with different matchers: *equalString, equivalentConcept, superConcept, subConcept, equalSuperConcept*. The output of each matcher is a matching which is defined as a tuple (a_s, a_t, m_{exp}, C_l) where a_s is an annotation from the source schema, a_t is an annotation from the target schema, m_{exp} is a matching expression and C_l is a confidence level. The matching expression is a directional relation from the source to the target schemas between the matched annotations (not all matchings allow non directional relations). The confidence level C_l of the interval [0,1] expresses the certainty of a match which depends on a statical value of the used matcher (a confidence value C_v) and a dynamic value which depends on the match situation. Higher value indicates higher level of certainty.

The matching methods are divided into two group: lossy and lossless. Lossy matching creates $quasi - matching$, because some information may be lost[8] or that matchings may be incorrect. On the other hand, lossless matching guar-

antees the high quality of the matching based on strong semantic relations of the annotation concepts in the ontology.

equivalentConcept - creates a matching candidate based on reasoning if and only if annotation concept $p.concept_s$ is equivalent to annotation concept $p.concept_t$. Therefore, the semantics of source and target are equal. This matcher returns matchings with a very high confidence level C_l which depends on high static confidence value C_v and no dynamic aspect.

equalString - Is more or less an optimization of the first matcher. It creates a matching candidate if and only if the annotation path from the source schema p_s is syntactically equal to the annotation path from the target schema p_t. That matching method also returns high static values, because syntactically equivalent path must also be semantically equivalent. This matcher is always used first. The other matchers are only applied, if no matching could be established with this matcher.

subConcept - creates a matching candidate based on reasoning if and only if $p.concept_s$ is a sub-concept of $p.concept_t$. The confidence level C_l of matchings, which are returned by this matcher, depends on the quality of the sub-class relation, i.e. a relation son-parent is considered to be more semantics preserving than son-grandparent. The best candidate is thus, selected based on the $semantic\ similarity$ between the concepts in the ontology. The semantic similarity $SemSim$ is presented in [4]. It is a value from interval [0,1] and expresses a level of similarity between concepts. Value 1 indicates a strong relation, 0 in the opposite. A parameter of the semantic similarity is the $semantic\ distance$ which is a hierarchical distance between concepts in the ontology with computed weights. It is contrary to semantic similarity, longer distance between concepts (longer path in the ontological hierarchy tree) indicates lower similarity between them. Because of that the confidence level for matched concepts depends on semantic similarity and is computed by equation 1.

$$C_l = C_v + SemSim(p.concept_s, p.concept_t) \tag{1}$$

superConcept - creates a matching candidate based on reasoning if and only if $p.concept_s$ is a super-concept of $p.concept_t$ (lossy matching). That method is contrary to the $subConcept$ matcher and matches a generalization of concept with a specialization. In contrast to the previous approaches this is not a strong semantical relation. Nevertheless there are situations, where this

matcher can find suitable matchings. This matcher thus, returns matchings with a very low confidence level and the resulting match needs additional review by the user. Moreover, C_l is calculated also by equation 1.

equalSuperConcept - creates matching candidate based on reasoning if and only if semantic annotation concepts $p.concept_s$ and $p.concept_t$ have the same super-concept (lossy matching). This matcher is also not based on strict semantics and thus additional review by the user is required.

Algorithm 1 is used to generate the matchings between the semantic annotations. It takes as input the set of semantic annotations from the source schema A_s, the set of annotations from the target schema A_t and creates a set of matchings M_A. For each annotation path from the target algorithm searches the best possible matching with annotation from the source. Algorithm selects the best matching candidate based on the confidence level.

Algorithm 1 MATCHING($\Downarrow$ Annotations A_s,$\Downarrow$ Annotations A_t, $\Uparrow$ Matchings M_A)

```
1: for all Matcher matcher ∈ MatchingMethods do
2:     for all a_t ∈ A_t do
3:         for all a_s ∈ A_s do
4:             matcher.getMatch(a_s,a_t,M_A);
5:         end for
6:     end for
7: end for
```

3 Complex matchings

In this section we extend the matchings from the previous one by complex matchings with multiple input annotations and a $transformation\ function$ as the matching expression. That $transformation\ function$ expresses a relationship between source annotation and target annotation. For instance the system can create a matching for the concepts $FirstName$, $LastName$ from the source with $FullName$ from the target (n:1 matching). The $Transformation\ function$ for that example is a concatenation of two inputs into one output with a predefined separator.

Moreover our implementation is currently limited to *1:1* or *n:1* matchings with a transformation function as matching expression. The inputs can be any set of annotations from the source annotated XML schema S_s. The output can be any

annotation from the target schema S_t. However, most *1:n* or *n:m* matchings can be replaced by a set of *1:1* or *n:1* matchings. For instance the *1:n* matching which split a concept $FullName$ into $FirstName$ and $LastName$ can be replaced by two *1:1* matchings: One that matches $Fullname$ to $FirstName$ and one for $FullName$ to $LastName$.

These transformations are defined and annotated by users and stored in a $transformation\ library$. The annotations of the inputs and output of the transformations in the library allows the automatic reuse by matching A_s with the annotations of the inputs and A_t with the annotations of the output. We have implemented the transformation library with a specific ontology where each transformation is an instance of that ontology. Additionally, there are generic functions with a set of ontological concepts as inputs, an ontological concept as an output and a transformation expression which transforms the inputs into the output. Because of equivalence, sub- and super-concept relations between semantic annotations concepts $p.concepts$ and inputs/output from the transformation the system is able to find an appropriate (explicit) transformation and create a matching based on the transformation function. In case of many possible sub-concepts, the concept with the highest semantic similarity to the required concept is used. Moreover, the algorithm prefers not yet matched annotations, because annotations should be used only once. It ensures that the system matches as many annotations as possible. Transformations can resolve many matching problems, for instance: elements with different granularity (concatenation/split), currency convert (i.e. from Euro to Dollar), date format (i.e. from 2010-09-24 to 24th Sep 10), value map (i.e. renames of values), aggregation (i.e. sum or count), schema instance problems.

If the $transformation\ library$ does not contain an explicit transformation for non-matched annotations, the system tries to automatically create an implicit transformation. Such a transformation is a combination of explicit transformations. It is represented as a tree, where the root is the output and leaf-nodes are the matched annotations. Intermediate nodes are explicit transformations, edges connect inputs and outputs of the explicit transformations. Such implicit transformations are generated by an iterative deeping depth-first[7] search (IDDFS) algorithm. The algorithm matches the inputs with the outputs of the transformations based on their annotations. The usage of an IDDFS algorithm has the advantage that the shortest possible implicit transformations are found.

For instance Figure 1 presents an example of a matching with an implicit transformation. The elements first name with title (annotated by /user/hasFirstNameWithTitle/NameWithTitle) and last name (annotated by /user/hasLastName/-lastName) are matched with the element initials (annotated by /user/hasInitials/-

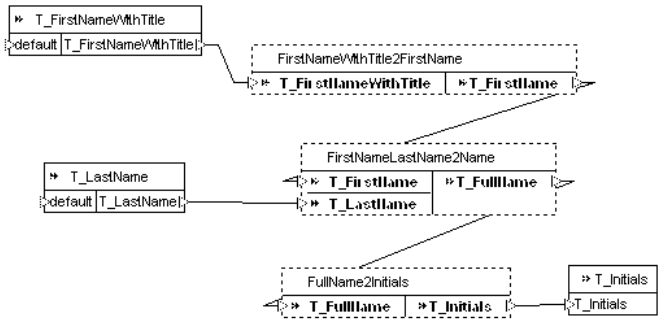

Figure 1: Details of implicit transformation (Altova®MapForce®).

initsials). Because the $Transformation\ library$ does not contain this transformation, three other existing transformations are composed to achieve the result. First of all it transforms first name with title to first name using transformation $FirstNameWithTitle2FirstName$. Afterwards, the result of this transformation and the second input (last name) are inputs of transformation $FirstName-LastName2Name$ which transforms first name and last name to full name. Finally, the full name is transformed by transformation $FullName2Initials$ to initials.

4 Schema level matching

In the previous phases matchings between annotations were generated based on different matching methods. The matching expression between annotations are either direct mappings or transformation functions. The next step, schema level matching, applies the annotation's matchings to the corresponding XML schemas and creates a set of mappings between XML-nodes which is later used to generate a transformation script.

We use Algorithm 2 for the mapping of schema nodes. It takes the following inputs: the source annotated XML schema S_s, the target annotated XML schema S_t and the set of matchings M_A. The algorithm searches for the best possible mapping between nodes based on the matched annotations and returns the set of mappings M_N. M_N contains mappings m_N between nodes from the source schema n_s and the target schema n_t with optional transformation functions.

For each node from the target schema the algorithm executes two times the method $getMatchingNodes$. The method returns nodes which path annotations are matched based on the set of matchings M_A. The first execution returns a set

of nodes from the target schema (line 2), the second from the source schema (line 3).

In most cases cardinality of sets with nodes from the source and target schema are equal 1. In that case the system creates a *1:1* mapping between the selected nodes (line 5). However, there may occur that the cardinality is bigger than 1, for instance when input schemas contain more than one node with the same annotation. Because of that there are three conditions: *1:n*, *n:1* and *n:m* mappings. In the first case, when the set of nodes from the target schema contains more than one element (line 7), simply each selected node from the target schema is mapped with the corresponding node from the source schema.

In other cases, when the set of nodes from the source schema or/and the set of nodes from the target schema contains more than one element (line 10/13), the system has to choose which node from the source has to be mapped with the node from the target.

The used heuristic solution is based on the $relative\ distance$ ($RelDist$). It is computed between a conflicting node n and previously matched node n_{ref} (sibling or ancestor of n) from the source schema by equation 2:

$$RelDist(n, n_{ref}) = |AbsPos(n) - AbsPos(n_{ref})|, \tag{2}$$

where $AbsPos(n)$ is an absolute position of the node n on flattened structure of schema.

The system chooses the node n with the lowest value of the relative distance. The assumption is that typically semantically related nodes are grouped together in both schemas. Of course this is only a heuristics where the quality of the matching depends on the input data and the nature of the conflict scenario. Moreover, the schema level matching is a complex process. There could exist additional conflicts between matched elements because of the schema restrictions, i.e. cardinality, data-type range constraints, enumeration constraints or pattern constraints, which are currently not considered by the prototype. Most of these problems can be avoided by well-annotated XML schemas.

5 Transformation generation

The last step of our implementation is to generate a transformation script which is able to transform instance documents of the source schema to instance documents of the target schema. The transformation script is generated based on the set of mappings from the previous section 4. There are many applications which generate transformation script based on mappings between schema elements. One of

Algorithm 2 NODEMAPPING(⇓ Schema S_s, S_t, ⇓ Matchings M_A, ⇑ Mappings M_N)

1: **for all** $n_t \in S_t.N$ **do**
2: N_T := getMatchingNodes(S_t, $n_t.annotation$, M_A);
3: N_S := getMatchingNodes(S_s, $n_t.annotation$, M_A);
4: **if** $|N_S|$==1 and $|N_T|$ == 1 **then**
5: m_N := createOneToOneMapping(N_S[0],n_t);
6: $M_N \uplus m_N$;
7: **else if** $|N_S|$ == 1 and $|N_T| > 1$ **then**
8: $M_{OneToMany}$:= createOneToManyMapping(N_S[0],N_T);
9: $M_N \uplus M_{OneToMany}$;
10: **else if** $|N_S| > 1$ and $|N_T|$==1 **then**
11: $M_{ManyToOne}$:= createManyToOneMapping(N_S,n_t);
12: $M_N \uplus M_{ManyToOne}$;
13: **else if** $|N_S| > 1$ and $|N_T| > 1$ **then**
14: $M_{ManyToMany}$:= createManyToManyMapping(N_S,N_T);
15: $M_N \uplus M_{ManyToMany}$;
16: **end if**
17: **end for**

them is Altova®MapForce®which is applied in our implementation. Our existing prototype stores the set of schema mappings as $MapForce\ Design\ Files$ (MFD). Such a design file is basically an XML-representation of the schema mapping. The file can be opened in MapForce®. Therefore, additional review and refinement by the user is possible. Finally MapForce®can be used to create XSLT[12] or XQuery[13] transformation scripts automatically.

6 Conclusion and future work

In this paper we have presented a matching system that creates complex schema matchings and mappings between different annotated XML-Schemas. Those mappings can directly be used to generate transformation scripts that transform documents of the source schema to documents of the target schema. The presented heuristic matching approach is based on an expressive and declarative annotation method. It operates in multiple stages: Semantic matching, complex matching, schema-level matching and output generation. The approach has been implemented an its usefulness could be demonstrated.

Moreover, one major task in the proposed scenario is the addition of semantic annotations to source and target XML schemas. We plan to automate this task as much as possible in the future. The existing work in the area of soft-computing such as [2, 3] are expected to be a good foundation for a semi-automatic annotation system.

Acknowledgment

This contribution is supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme (2007-2013) and European Regional Development Fund.

References

[1] David Aumueller, Hong H. Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908, New York, NY, USA, 2005. ACM.

[2] Antoon Bronselaer and Guy De Tré. A possibilistic approach to string comparison. *Trans. Fuz Sys.*, 17:208–223, February 2009.

[3] Antoon Bronselaer, Axel Hallez, and Guy De Tré. A possibilistic view on set and multiset comparison. *Control and Cybernetics*, 38:341–366, 2009.

[4] Jike Ge and Yuhui Qiu. Concept similarity matching based on semantic distance. In *SKG '08: Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid*, pages 380–383, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Pascal Hitzler, Markus Krtzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer. W3C Recommendation, World Wide Web Consortium, October 2009.

[6] Julius Köpke and Johann Eder. Semantic annotation of xml-schema for document transformations. In Robert Meersman, Tharam Dillon, and Pilar Herrero, editors, *Proc. of OTM'10 Workshops*, volume 6428 of *LNCS*, pages 219–228. Springer, 2010.

[7] Richard E. Korf. Depth-first iterative-deepening: an optimal admissible tree search. *Artif. Intell.*, 27(1):97–109, 1985.

[8] Michele Missikoff and Francesco Taglino. Semantic mismatchffes hampering data exchange between heterogeneous web services. In *W3C Workshop on Frameworks for Semantics in Web Services*, 2005.

[9] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[10] Marcin Szymczak. Semantic annotation-based XML document transformation. Master's thesis, University of Klagenfurt, Universitatesstrasse 65-67, Klagenfurt, 2010.

[11] W3C. Semantic Annotations for WSDL Working Group, 2006-2007.

[12] W3C. XSL Transformations (XSLT) Version 1.0, September 2010.

[13] Norman Walsh, Mary Fernández, Ashok Malhotra, Marton Nagy, and Jonathan Marsh. XQuery 1.0 and XPath 2.0 data model (XDM). W3C recommendation, W3C, January 2007.

The papers presented in this Volume 2 constitute a collection of contributions, both of a foundational and applied type, by both well-known experts and young researchers in various fields of broadly perceived intelligent systems.
It may be viewed as a result of fruitful discussions held during the Tenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGN-2011) organized in Warsaw on September 30, 2011 by the Systems Research Institute, Polish Academy of Sciences, in Warsaw, Poland, Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences in Sofia, Bulgaria, and WIT - Warsaw School of Information Technology in Warsaw, Poland, and co-organized by: the Matej Bel University, Banska Bystrica, Slovakia, Universidad Publica de Navarra, Pamplona, Spain, Universidade de Tras-Os-Montes e Alto Douro, Vila Real, Portugal, and the University of Westminster, Harrow, UK:

Http://www.ibspan.waw.pl/ifs2011

The consecutive International Workshops on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGNs) have been meant to provide a forum for the presentation of new results and for scientific discussion on new developments in foundations and applications of intuitionistic fuzzy sets and generalized nets pioneered by Professor Krassimir T. Atanassov. Other topics related to broadly perceived representation and processing of uncertain and imprecise information and intelligent systems have also been included. The Tenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGN-2011) is a continuation of this undertaking, and provides many new ideas and results in the areas concerned.

We hope that a collection of main contributions presented at the Workshop, completed with many papers by leading experts who have not been able to participate, will provide a source of much needed information on recent trends in the topics considered.

ISBN-13 9788389475411