Scalable Graph Query Evaluation and Benchmarking with Realistic Models

Gábor Szárnyas
Budapest University of Technology and Economics
Department of Measurement and Information Systems
MTA-BME Lendület Research Group on Cyber-Physical Systems
szarnyas@mit.bme.hu

ABSTRACT

Model queries are widely used in model-driven engineering toolchains: models are checked for errors with validation queries, model simulations and transformations require complex pattern matching, while injective mappings for views are defined with model queries. Efficient and scalable evaluation of complex queries on large models is a challenging task. To achieve scalable graph query evaluation, I identified key challenges such as the lack of credible benchmarks and difficulties of obtaining real models for performance testing. To address these challenges, my contributions target (1) distributed incremental graph queries, (2) a cross-technology benchmark for model validation, (3) characterization of realistic models, and (4) realistic models generation.

ACM Reference format:
Gábor Szárnyas. 2017. Scalable Graph Query Evaluation and Benchmarking with Realistic Models. In *Proceedings of Student Research Competition, San Francisco, June 2017 (ACM SRC),* 5 pages.
DOI: 10.475/123_4

1 PROBLEM AND MOTIVATION

Model-Driven Engineering (MDE) is a development methodology used in many application domains such as critical applications (automotive, avionics and railway systems [11, 25, 49]). To increase the efficiency of development, MDE facilitates the use of models in various modelling languages targeting different levels of abstraction. Models can be used not only for presenting the structure and behaviour of the system, but also for synthesizing various design artifacts (such as source code, configuration files, documentation). To catch design flaws early, *model validation* techniques check the *well-formedness* of models. Design rules and *well-formedness constraints* are often captured in the form of *graph patterns* [6] to highlight invalid model elements to systems engineers. MDE tools check these patterns by evaluating *graph queries*.[1]

[1]In this paper, I use the term *graph* as a synonym for *instance model*.

ACM SRC, San Francisco

DOI: 10.475/123_4

1.1 Scalable Graph Queries

As models are rapidly increasing in size and complexity, efficient execution of model validation operations is challenging for the currently available toolchains, like ARTOP [1], Capella [31] or Papyrus [16].

The last decade brought considerable improvements in distributed storage and query technologies, known as NoSQL systems. These systems provide quick evaluation of simple retrieval operations and they are able to answer complex queries in a scalable manner, albeit not instantly. Providing quick response times for evaluating such queries over large and evolving data sets is still a challenging task.

Graph queries capturing validation constraints are often complex, including many join, antijoin and filtering operations. However, most query technologies cannot efficiently evaluate such operations for models with 10 million model elements [42], while models of critical systems, software and geospatial models are often 1–2 orders of magnitude larger [36]. A possible solution for scalable graph queries is to use distributed query processing techniques [13, 50]. This brings us to the first research question I investigated.

> **RQ 1.** *How to incrementally evaluate graph queries over a distributed platform?*

1.2 Benchmarking

To assess the performance of a graph query engine, a benchmark framework is of high importance. According to the *Benchmark Handbook* [17], a useful benchmark is (1) relevant, (2) portable, (3) scalable, and (4) simple. To ensure *relevance*, the benchmark must use a *representative* workload and data sets similar to realistic ones. Providing relevant results, while also guaranteeing the other properties (portability, scalability and simplicity) is a major challenge.

For real-world industrial systems, both metamodels and instance models are protected by intellectual property rights (IPR). For example, AUTOSAR [11] is not an open standard, but only available to members of the consortium, therefore it is not suitable for an open performance benchmark. Similarly, engineering models in the avionics and railway domains are also not available to the public.

These challenges confirm the need for a benchmark framework, which provides a real-world-like workload scenario and evaluates realistic queries on realistic models. Therefore, the second research question is the following.

> **RQ 2.** *How to assess query technologies for a continuous model validation scenario?*

1.3 Characterization of Realistic Models

While existing generators may produce large models in increasing sizes, these models are usually simple and synthetic, which hinders their credibility for industrial and research benchmarking purposes. Up to my best knowledge, there are no existing techniques to characterize models used in MDE practice. To develop such a technique, first I had to address questions about model metrics, such as:

- Which metrics can be used for characterizing models?
- Is is possible to distinguish models of different domains, purely based on their metrics?

To answer these questions, I conducted a literature review in other disciplines, e.g. network theory and social network analysis. The high-level goal of the research is to answer the following question.

RQ 3. *What makes a model realistic?*

1.4 Generating Realistic Models

Custom generators of graph-based models are used in MDE for many purposes such as functional testing and performance benchmarking of modeling environments to ensure the correctness and scalability of tools. However, none is capable of generating realistic models scalable in size:

- Logic-based synthesis (like Alloy [21]) generate well-formed models but lack scalability.
- Rule-based approaches [42] are capable of generating large models by using transformation rules or random mutations to add new elements. However, they provide no guarantees that the resulting model is realistic. Some approaches do not even guarantee well-formedness, which is a prerequisite for realistic models.

It is an open research question if it is possible to ensure these properties.

RQ 4. *How to generate scalable and realistic models?*

2 PRELIMINARIES

This section introduces an example used throughout the paper and presents the concept of *incremental queries*.

2.1 Running Example: Railway Network

As a running example, I use a small railway network, defined on the metamodel of the Train Benchmark [42], a model validation benchmark (the benchmark and my related contributions are discussed in Section 4.2).[2] Figure 1 shows a schematic representation of the network, with routes (1–3), switches and segments. As the first switch is set to a straight position and the second switch is set to a diverging position, a train passing through this track would follow route #3, hence that route *active*.

Modeling tools often represent their models as graphs. Figure 2 shows the example network as a labelled, attributed graph, along with the metamodel of the graph. Routes follow a set of *switch*

[2]To guarantee that the example is concise and easy to understand, the example only uses a fraction of the Train Benchmark metamodel. The benchmark uses models that are significantly more complex: they contain more metamodel elements (types) and consist of more elements (objects).

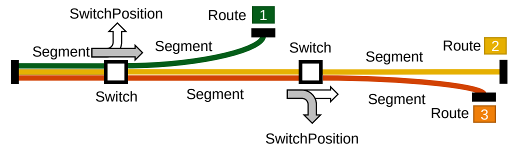

Figure 1: Railway example model. The positions of the switches designate route #3 as active.

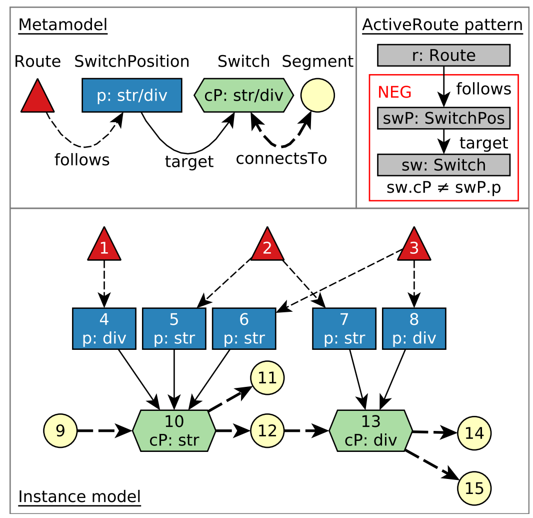

Figure 2: Representing the railway example of Figure 1 as a graph. The top left corner shows the metamodel is in the, the top right corner presents the graph pattern for finding the active routes. The instance model is shown in the bottom.

positions that contain the prescribed *position* (straight or diverging) of the switch. The railway track consists of switches and segments.

The *active route* can be determined by evaluating a *graph query* (by *graph pattern matching*). A route is *active* if all its switches are in the position prescribed by the switch positions of the route. In other words, a route is active if none of its switches are set to a different position as the prescribed position. This pattern is shown in the upper right corner of Figure 2. In the example, the graph query selects route #3 as the active one, as both its switch positions (6 and 8) are satisfied by the corresponding switches (10 and 13).

2.2 Incremental Query Evaluation

In many use cases, queries are continuously evaluated, while changes affect only a restricted part of data. The queries and transformations for simulation and well-formedness validation in MDE are typical examples of such a workload. The goal of *incremental query evaluation* is to speed up such queries, utilizing the (partial) results obtained during the previous executions of the query to compute the latest set of changes. For example, if the *current position* of the second switch in Figure 1 changes from diverging to straight, the change only affects a small part of the graph (node 13 in Figure 2). This allows an incremental query engine to quickly reevaluate the query: in this case, the *active route* is changed from #3 to #2.

Incremental query evaluation algorithms use additional data structures for caching interim results, hence they consume more memory than search-based, non-incremental algorithms. In other words, they trade memory consumption for execution speed. While incremental query engines provide quick response times for various use cases [6, 42], their excessive memory consumption limits their scalability.

3 RELATED WORK

To appropriately address all the research questions in the context of MDE, a wide range of multidisciplinary topics needs to be covered.

Distributed incremental graph queries. The Rete algorithm was originally created by Charles Forgy for rule-based expert systems [15]. Bunke et al. [10] were the first to propose the Rete algorithm in the context of graph transformations. Bergmann et al. adapted the algorithm for the Eclipse Modeling Framework in the EMF-IncQuery project [6], now part of the Viatra project [46].

Query languages and execution engines have been developed to support *incremental graph queries* on a single-machine environment. Drools [22] is an incremental business rule engine for Java-based systems. INSTANS [33] provides incremental queries over RDF [48].

Cross-technology benchmark for continuous validation. Numerous benchmarks have been proposed to compare the performance of query and transformation engines, but no openly available cross-technology benchmarks exist for continuous model validation.

The first transformation benchmark was proposed in [47], which gave an overview of typical application scenarios of graph transformations together with their characteristic features. Many transformation challenges have been proposed as cases for graph and model transformation contests. However, only [18, 51] focus on query performance, while others measure the usability of tools, the conciseness/readability of query languages and test various advanced features, including reflection, traceability, etc.

There are numerous benchmarks from the area of semantic databases. *SP^{2}Bench* [37] features a synthetic DBLP-like dataset, the *Berlin SPARQL Benchmark (BSBM)* [7] simulates an e-commerce application, while the *DBpedia SPARQL benchmark* [29] features a real data set with queries based on real-world user queries. The Linked Data Benchmark Council (LDBC) recently developed the Social Network Benchmark [14], a cross-technology benchmark, which provides an interactive workload and focuses on navigational pattern matching (i.e. traversal operations). While some of these benchmarks feature update operations and hence measure incremental query performance, they provide workloads that significantly differ from MDE use cases.

Characterization of realistic models. Revealing essential structural similarities and differentiations among networks from different fields is a fundamental objective in network theory with a wide range of applications. The authors of [12] list 22 areas using network theory, including social network analysis, transportation, biomolecular networks and chemistry. Network theory is also studied in physics, e.g. in the context of statistical mechanics [3]. However, most of these applications use untyped (one-dimensional) networks. So far, existing multidimensional studies only focused on models of a single application domain, such as neighbourhood and centrality analysis of a social network [8], relevance and correlation analysis of different dimensions in Flickr [23], community detection in the network of YouTube [44]. Multidimensional metrics are also defined in [5] where the authors study the expressiveness of their metrics on real-life networks.

Metrics are also used for understanding the main characteristics of domain-specific metamodels, for studying model transformations with respect to the corresponding metamodels, and search correlations between them via analytical measures [34].

Realistic model generation. The *SP^{2}Bench* [37] benchmark uses a generator based on the statistics of the DBLP library. The authors of [30] use *Boltzmann samplers* to ensure efficient generation of uniform models. OMOGEN [9] is a tool for automatically generating models for testing model transformations. The tool combines a set of model fragments to build larger instances. gMark [4] is a domain-independent framework for synthesizing large graphs, allowing the user to specify parameters for the graphs to be generated.

4 APPROACH AND CONTRIBUTIONS

This section presents my approach along with achieved and proposed contributions.

4.1 Distributed incremental graph queries

To achieve scalable incremental query evaluation, I adapted the Rete algorithm for distributed systems. I demonstrate the Rete algorithm works on the ActiveRoute query (Figure 2). As described in Section 2.1, the query collects Routes, where all Switches along the route are in the position prescribed by the corresponding SwitchPosition. In other words, without using the universal quantifier (∀), it searches for routes that do *not* have a SwitchPosition which prescribes a position *different* from the current position of its target Switch [32].

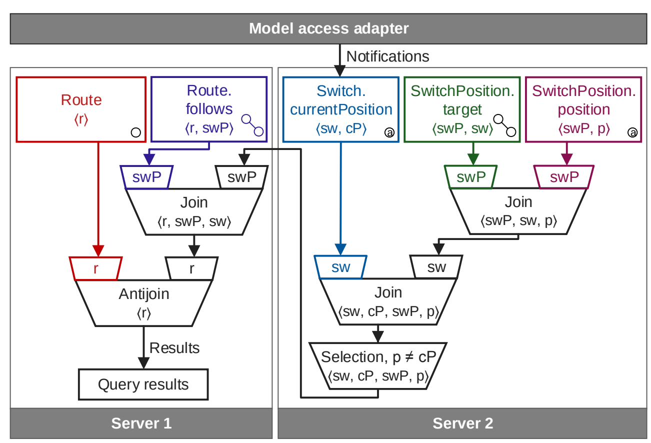

Figure 3: Rete network for the ActiveRoute pattern.

Figure 3 shows a distributed Rete network implementing this relational algebra expression. The network is allocated to two machines, Server 1 and Server 2. This allows the query engine to scale for larger graphs, for which the Rete network would not fit in the memory of a single workstation. However, this approach still has a bottleneck limiting scalability: if a Rete node cannot fit to the memory of a single workstation, it will run out of memory. Using these techniques and algorithms, I made following contributions.

Combine distributed actor model with Rete-based query evaluation network. I designed a distributed architecture and prototyped IncQuery-D, a Rete-based query engine using actors for distributed scalability. I presented a detailed performance evaluation in the context of incremental well-formedness validation. The results showed nearly instantaneous complex query reevaluation beyond 50M+ model elements [39]. To further extend the scalability of the system, I proposed *sharding* individual Rete nodes in [26].

Distributed termination protocol for asynchronous Rete. As Rete is an asynchronous algorithm, determining if the network is in a consistent state w.r.t. the latest change set requires a distributed *termination protocol.* The protocol was presented in [39] and [26].

Experimental evaluation over distributed NoSQL databases. The proposed architecture and algorithms are *representation-agnostic.* They have been integrated with graph databases, such as Neo4j [19] and 4store, a semantic database [39].

Evaluation of Rete network optimization and allocation strategies. Allocating the Rete nodes in the cloud is a complex optimization problem, where the goal is to minimize the cost of communication between the nodes. I presented a solver-based approach for allocating Rete nodes in [27]. I also proposed optimization techniques used in relational query optimization for enhancing the performance of graph queries [43].

Uniqueness. Up to my best knowledge, existing technologies are either distributed [24, 38] or incremental [46], but there is no system that provides scalable, distributed incremental graph queries.

4.2 Cross-technology benchmark for continuous validation

In Section 2.1, I used a running example from the Train Benchmark framework. The Train Benchmark is an incremental model validation benchmark, continuously developed by the Fault-Tolerant Systems Research Group since 2010. I have significantly extended the Train Benchmark, both conceptually and implementation-wise.

The Train Benchmark is a macro benchmark that aims to measure the performance of continuous model validation with graph-based models and constraints captured as queries. The benchmark is *cross-technology,* i.e. it is implemented on a range technologies, including Eclipse-based model-driven engineering toolchains (EMF), graph databases [35], relational databases (SQL) and semantic technologies (RDF [48]). The framework is extensible which allows users of the benchmark to incorporate new technologies.

Earlier versions of the benchmark have been continuously used for performance measurements since 2012 [39, 45]. The benchmark is also part of the benchmark suite used by the MONDO EU FP7 [28] project and was selected as a case for the 2015 Transformation Tool Contest [40] as well. The benchmark framework is available as an open-source project.[3]

[3] https://github.com/FTSRG/trainbenchmark

Scalable technology-agnostic model generator. While the original benchmark framework included a model generator, its scalability was limited. I redesigned the model generator focusing on two aspects: (1) ensuring scalability for large models, and (2) allowing the framework users to easily adapt new representations.

Propose novel query and transformation mixes for benchmark. The workload profile of the benchmark simulates *real-world model validation scenarios* of users loading, validating and transforming their models. The transformations capture user edits and quick-fix like automated refactoring operations. Some queries in the benchmark are structurally similar to AUTOSAR [11] validation queries (presented in [6]), while other aim to test various features of graph query engines (such as efficient filtering and evaluation of negative conditions).

Automated visualization and reporting. The framework features end-to-end automation [20] to (1) set up configurations of benchmark runs, (2) generate large model instances (3) execute benchmark measurements, (4) synthesize diagrams for measurements.

Cross-technology evaluation of incremental query execution time and memory consumption. This *cross-technology benchmark* can be adapted to different model representation formats and query technologies. This is demonstrated by 12+ reference implementations over four different technological spaces (EMF, graph databases, RDF and SQL) presented in [42].

Uniqueness. Compared to other benchmarks, the Train Benchmark has the following set of distinguishing features:

- The workload profile follows a real-world model validation scenario by updating the model with changes introduces by simulated user edits or transformations.
- The benchmark measures the performance of both initial validation and incremental revalidation.
- This benchmark was designed with cross-technology adaptations in mind. It can be implemented with different model representation formats and query technologies.

4.3 Characterization of realistic models

In [41], I presented multidisciplinary graph metrics and evaluated them on instance models from different domains. As a result, I proposed some metrics which turned out to be useful for characterizing the structure of models.

Adapt multidisciplinary metrics for engineering models. I performed a literature review and identified several graph metrics from other disciplines. For evaluating these metrics, I gathered instance models from six software and systems engineering domains.

Statistical characterization of different domains and models. I used both exploratory and confirmatory data analysis techniques in order to determine the "usefulness" of metrics. I considered a metric *useful* if it separates models of different domains from each other, while provides similar values for models within the same domain. I also investigated whether some of these metrics can *distinguish real models from auto-generated synthetic ones.*

Exploratory analysis relied on data visualization, while confirmatory analysis used statistical methods (such as performing

Kolmogorov–Smirnov tests on the derived metrics distributions). My initial finding is that different versions of clustering coefficients (i.e. how tightly connected the model elements are) were particularly useful for such classifications. But, unsurprisingly, no single metric was able to sufficiently handle all the domains. The analysis also provides some insights that can be used in future model generators to synthesize realistic models.

Automated classification of domain models using machine learning. As a *future research objective*, I plan to use *machine learning* techniques for automated classification of domain models.

Uniqueness. Up to my best knowledge, this is the first investigation for using multidimensional graph metrics for both *characterizing the realism* of models and *distinguishing different domain models* from each other.

4.4 Realistic model generation

As a *proposed contribution*, I plan to design and develop a generator that is capable of producing realistic models scalable in size. While there are solutions for generating either scalable or realistic models, there are no known approaches for the combination of both, rendering this a high-risk research task. The long-term research objective of generating scalable and realistic models breaks down to the following steps:

(1) metrics-guided generation of realistic models,
(2) domain model generation by design space exploration [2],
(3) scalable rule-based generation of domain models.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dániel Varró for his guidance during my research. I would also like to express my gratitude to István Ráth, Gábor Bergmann along with numerous colleagues and co-authors for sharing their ideas.

REFERENCES

[1] *Artop: The AUTOSAR Tool Platform.* https://www.artop.org/.
[2] Hani Abdeen and others. 2014. Multi-objective optimization in rule-based design space exploration. In *ASE*. 289–300.
[3] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1.
[4] Guillaume Bagan and others. 2016. Generating Flexible Workloads for Graph Databases. *Proc. VLDB Endow.* 9, 13 (Sept. 2016), 1457–1460.
[5] F. Battiston and others. 2014. Structural measures for multiplex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 89, 3 (2014).
[6] Gábor Bergmann and others. 2010. Incremental Evaluation of Model Queries over EMF Models. In *MODELS*. 76–90.
[7] Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL Benchmark. *Int. J. Semantic Web Inf. Syst.* 5, 2 (2009), 1–24.
[8] Piotr Bródka and others. 2012. Analysis of Neighbourhoods in Multi-layered Dynamic Social Networks. *Int. J. Comp. Intell. Syst.* 5, 3 (2012), 582–596.
[9] Erwan Brottier and others. 2006. Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool. In *ISSRE*. 85–94.
[10] Horst Bunke and others. 1990. An Efficient Implementation of Graph Grammars Based on the RETE Matching Algorithm. In *Graph-Grammars and Their Application to Computer Science*. 174–189.
[11] AUTOSAR Consortium. *The AUTOSAR Standard.* http://www.autosar.org/.
[12] Luciano da Fontoura Costa and others. 2011. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics* 60, 3 (2011), 329–412.
[13] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*. 137–150.
[14] Orri Erling and others. 2015. The LDBC Social Network Benchmark: Interactive Workload. In *SIGMOD*. 619–630.
[15] Charles Forgy. 1982. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. *Artif. Intell.* 19, 1 (1982), 17–37.
[16] The Eclipse Foundation. 2015. *Papyrus.* https://eclipse.org/papyrus/.
[17] Jim Gray (Ed.). 1993. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition).* Morgan Kaufmann.
[18] Tassilo Horn, Christian Krause, and Matthias Tichy. 2014. The TTC 2014 Movie Database Case. In *TTC at STAF*. 93–97.
[19] Benedek Izsó, Gábor Szárnyas, István Ráth, and Dániel Varró. 2013. IncQuery-D: Incremental graph search in the cloud. In *BigMDE*.
[20] Benedek Izsó, Gábor Szárnyas, István Ráth, and Dániel Varró. 2014. MONDO-SAM: A Framework to Systematically Assess MDE Scalability. In *BigMDE at STAF*. 40–43.
[21] Daniel Jackson. 2002. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.* 11, 2 (2002), 256–290.
[22] JBoss. *Drools.* http://www.jboss.org/drools.
[23] Przemyslaw Kazienko, Katarzyna Musial, and Tomasz Kajdanowicz. 2011. Multidimensional Social Network in the Social Recommender System. *IEEE Trans. Systems, Man, and Cybernetics* 41, 4 (2011), 746–759.
[24] Christian Krause and others. 2014. Implementing Graph Transformations in the Bulk Synchronous Parallel Model. In *FASE*.
[25] Bjørnar Luteberget and others. 2016. Rule-Based Consistency Checking of Railway Infrastructure Designs. In *IFM*. 491–507.
[26] János Maginecz and Gábor Szárnyas. 2016. Sharded Joins for Scalable Incremental Graph Queries. In *23rd PhD Mini-Symposium*. Budapest University of Technology and Economics.
[27] József Makai and others. 2015. Optimization of Incremental Queries in the Cloud. In *CloudMDE at MODELS*.
[28] MONDO project. 2016. *Scalable Modeling and Model Management on the Cloud Project, 7th EU Framework Programme.*
[29] Mohamed Morsey and others. 2011. DBpedia SPARQL benchmark: Performance Assessment with Real Queries on Real Data. In *ISWC*. 16.
[30] Alix Mougenot and others. 2009. Uniform Random Generation of Huge Metamodel Instances. In *ECMDA-FA*. 130–145.
[31] PolarSys. *Capella.* https://www.polarsys.org/capella/.
[32] Arend Rensink. 2004. Representing First-Order Logic Using Graphs. In *ICGT*. 319–335.
[33] Mikko Rinne and others. 2012. INSTANS: High-Performance Event Processing with Standard RDF and SPARQL. In *ISWC*.
[34] Juri Di Rocco and others. 2015. Mining Correlations of ATL Model Transformation and Metamodel Metrics. In *MiSE*. 54–59.
[35] Marko A. Rodriguez and Peter Neubauer. 2010. Constructions from Dots and Lines. *Bulletin of American Society for Information Science & Technology* (2010).
[36] Markus Scheidgen and others. 2012. Automated and Transparent Model Fragmentation for Persisting Large Models. In *MODELS*. 102–118.
[37] Michael Schmidt and others. 2009. SP2Bench: A SPARQL Performance Benchmark. In *ICDE*. IEEE.
[38] Bin Shao, Haixun Wang, and Yatao Li. 2013. Trinity: a distributed graph engine on a memory cloud. In *SIGMOD*.
[39] Gábor Szárnyas and others. 2014. IncQuery-D: A Distributed Incremental Model Query Framework in the Cloud. In *MODELS*. 653–669.
[40] Gábor Szárnyas and others. 2015. The TTC 2015 Train Benchmark Case for Incremental Model Validation. In *TTC*. 129–141.
[41] Gábor Szárnyas and others. 2016. Towards the Characterization of Realistic Models: Evaluation of Multidisciplinary Graph Metrics. In *MODELS*. 87–94.
[42] Gábor Szárnyas, Benedek Izsó, István Ráth, and Dániel Varró. 2017. The Train Benchmark: Cross-Technology Performance Evaluation of Continuous Model Validation. *Softw Syst Model* (2017).
[43] Gábor Szárnyas, János Maginecz, and Dániel Varró. 2017. Evaluation of Optimization Strategies for Incremental Graph Queries. *Periodica Polytechnica, EECS* (2017).
[44] Lei Tang and others. 2012. Community detection via heterogeneous interaction analysis. *Data Min. Knowl. Discov.* 25, 1 (2012), 1–33.
[45] Zoltán Ujhelyi and others. 2015. EMF-IncQuery: An integrated development environment for live model queries. *Sci. Comput. Program.* 98 (2015), 80–99.
[46] Dániel Varró and others. 2016. Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework. *Softw Syst Model* 15, 3 (2016), 609–629.
[47] Gergely Varró, Andy Schürr, and Dániel Varró. 2005. Benchmarking for Graph Transformation. In *VL/HCC*. IEEE Press.
[48] W3C. *Resource Description Framework (RDF).* http://www.w3.org/standards/techs/rdf/.
[49] Jon Whittle and others. 2014. The State of Practice in Model-Driven Engineering. *IEEE Software* 31, 3 (2014), 79–85.
[50] Reynold S. Xin and others. 2013. GraphX: a resilient distributed graph system on Spark. In *GRADES at SIGMOD/PODS*. 2:1–2:6.
[51] Albert Zündorf. 2008. *AntWorld benchmark specification, GraBaTs.* http://is.tm.tue.nl/staff/pvgorp/events/grabats2009/cases/grabats2008performancecase.pdf