Reasoning in agent-based network management

Vilho Räisänen
Nokia Bell Labs
Espoo, Finland
Email: first.last@nokia-bell-labs.com

Kasper Apajalahti
Department of Computer Science
Aalto University
Espoo, Finland
Email: first.last@aalto.fi

***Abstract*—An increasing complexity of mobile networks and use cases is reflected in requirements for network management. Advanced automation is required to address this challenge in an economically feasible manner. We describe a system based on agent mapping as a platform for automation for network management in 5G networks and beyond. We use classical and probabilistic reasoning for composing solutions to complex requests by means of relatively simple software agents. We describe different variants of the approach in terms of capabilities, ranging from triple store only to system with semantic and probabilistic reasoning functionalities. This approach provides flexibility for network functionality evolution and facilitates software reuse and is compatible with the use of task-specific machine learning algorithms in network management agents. We describe test system used for evaluating the concept, as well as use case evaluation obtained with it.**

I. Introduction

Goals for improvement of performance in 5G networks over previous systems have been laid out in [1]. The implementation of the requirements lead to architectural choices in network architecture. On the one hand, the network architecture is cloud-based, with 3GPP applications executed as virtualized applications [2]. On the other hand, the 5G network deployment is expected to increase technological complexity, highlighting the need for advanced automation. Furthermore, simultaneous support for 5G traffic types — massive machine type communications (mMTC), critical machine type communication (cMTC), and extreme broadband (xMBB) — requires new technologies for 5G access, which in most cases needs to co-exist with legacy access technologies. The role of network management needs to be re-assessed in the new architecture.

It is expected that new technologies in 5G such as network slices [3], [4] serve as a platform for new services. Taken together with the increasing complexity of radio access, flexibility is needed for network management since future needs cannot be fully predicted. The paradigms for network management have evolved from manual and template-based management towards agent-based management in Long-Term Evolution (LTE) Self-Organizing Networks (SON) paradigm [5], [6], [8]. Agents are a realization of autonomic computing concept for mobile network management [7]. The limitations of rule-based SON have become apparent in terms of feasibility of tailoring to varying network contexts.

In 5G, cloud-based execution is an integral part of the architecture, and configuration targets for NM are partly virtualized [3]. As a consequence of virtual execution environments in 5G, there is an interplay between orchestration of virtualized resources and NM. For example, orchestration affects the set of virtual functionalities managed by Network Management (NM). In [9], a framework was proposed for virtualized NM which lends itself to an analysis of interactions of agent-based network management for such a case.

In this article, we describe an architecture based on agent composition for NM. The composition in our approach makes use of classical reasoning based on semantic models. We argue that this approach facilitates agile development of network management capabilities while supporting efficiency in terms of software reuse.

In what follows, we discuss the role of agents in operability, followed by an account of the use of reasoning in the same area. We then proceed to describe our approach and a demonstrator environment and use case evaluation. We conclude this article with a summary.

We shall describe relevant prior references within the following technologies Sections in the interests of compactness of presentation and understandability.

II. Reasoning in network management

Automated reasoning based on First Order Logic (FOL) can be used to partially replace traditional programming, with the advantage that the consistency of the logical models employed are automatically checked. This lessens the need for testing in validation of proper functioning of a module. The use of logical models such as Description Logic -derived web ontology language (OWL) allows expressing computations with constructs akin to domain models, more understandable for humans than software implementations.

The scaling properties of reasoning algorithms are well understood, and pose limits for application in NM area. It is not feasible to represent the entire state of a mobile network in a single logical model for management purposes. For focused uses such as semantic modelling of configuration management [13], [14], mapping between concepts across domains [12], or analysis of agent coordination [10], classical reasoning is a valid choice. An architecture for knowledge delivery for the purposes of semantic interoperability of autonomic agents has been proposed in [15].

Earlier we noted that rule-based agents are not feasible for tailoring of NM to individual cell contexts. The use of a suitable variant of Case-Based Reasoning [16] allows for interpolation between cases, but is still limited by the ”space”

spanned by the case base. In view of the increasing complexity of networking technology, one should allow for the choice of a machine learning (ML) algorithm that is appropriate for the data at hand. Several ML algorithms may thus be used concurrently in agents.

The use of a set of machine learning algorithms in agents gives rise to semantic challenge in integrating the output of the learning agents to NM. Another challenge is related to knowledge of network state. For example, what-if analyses might be useful for human users given a set of information about network which may be incomplete. It can be argued that a combination of semantic reasoning and probabilistic reasoning addresses both of these problems [11]. Semantic modelling lends itself well to mapping between sets of concepts. Probabilistic reasoning complements semantic reasoning with ability to resolve conflicting information. Combining the two technologies, a mapping within an ontology can be achieved for concepts specific to machine learning algorithms.

Probabilistic reasoning is employed to establish the most likely explanation for the set of input information at hand — possibly contradictory – in the context of a domain model [11]. The most likely explanation can then be used in classical reasoning. The consistency of the output of probabilistic reasoning with classical reasoning can be ensured by using a method such a Markov Logic Network (MLN, [17]) which supports both a combination of certain and uncertain rules (latter ones associated with weights).

III. Reasoning in agent composition

In this section, we describe our reasoning system that dynamically composes agents to perform NM tasks (requests). We use triple store (graph database) with SPARQL query language, Description Logic (DL) reasoner, and probabilistic reasoning providing additional capabilities. This kind of system allows for inference over historical data which is represented as triples, with additional capabilities providing further functionality which is described later on. Such inference can be used in composing a solution to a request by means of agents. The triple store reasoning step itself replaces some traditional programming as we shall see later on.

Processing based on data included in requests is enabled by a reasoner which uses an ontology to infer further triples based on request data. As we shall see later on, probabilistic reasoning can be employed *en route* to accommodate hypothetical information such as output of machine learning algorithms.

The triple store can be viewed as a realization of an ontology, which in the "triple store only" case does not exist as a separate entity. When reasoning is used, it employs a formal ontology describing the knowledge model used for reasoning and infers implies triples. In our case, the main ingredient of ontology is domain model which ensures consistency of reasoning in NM.

A. Description of the approach

The software architecture is shown in Figure 1. Requests are received by a service interface, processed by a composition

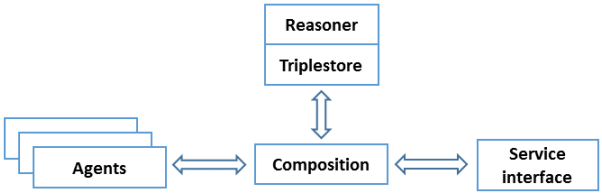

Fig. 1. Mapping architecture overview. Composition receives requests from service interface and uses query to triple store in agent mapping. The contents of triple store may be updated with triples induced by reasoner.

entity, and mapped to execution in $[0, N]$ agents. The "zero agents" case corresponds to performing all related reasoning in a combination composition entity and triple store query. The mapping could be performed by means of a rule base, but we argue that semantic modelling and reasoning provides more flexibility compared to static rule bases.

The core concept in our ontological approach is mapping of incoming requests to agents via operations and effects. Figure 2 describes the simplified ontology used for this mapping. The idea is adapted from a simple semantic web service model, WSMO-lite [20]. An agent — analogous to a service in WSMO-lite — has $[1, N]$ operations that monitor or change the status of the target. Operations have effects that represents the desired impact in the target. Furthermore, operations have metadata; for example, operation area (the part of the network where the agent is operating) and temporal range.

Fig. 2. Ontology constructs for agent (top), request (middle), and network measurements (bottom).

Inference over historical data only does not require the use of reasoner, and SPARQL query is sufficient. This correponds to a case where the query does not include new data instances, only parameters via which to query the existing data. Such parameters can be e.g. network scope (set of cells) or temporal range. In this approach, reasoner is not involved in processing the query, but may have been used when relevant data have arrived for inferring new triples.

From the viewpoint of knowledge model, a request including parameters corresponds to one or more instances that activate the OWL reasoner when added to the ontology. The Figure 2 shows a case in which the request is associated with effects in the ontology, reflecting concrete objectives that need to be met with the operations. The request instance is associated with a network scope (relevant cells) and point in time. The request may also have pre- and post conditions that need to be met with, such as a cell loading level. An example of such a request in LTE could be improvement of service quality in a set of cells, which would be associated with the

effect of changes in CQI distribution.

In earlier research, Web service execution environment for creation and execution of semantic web services based on ontology [18]. In another article, Distributed Management Task Force (DMTF) Common Information Model (CIM) is "lifted" to OWL and used for service composition with OWL-S [19]. Compared to these approaches, our system is not based on WS-* web services suite and takes a minimalist approach to ontology. As we shall see, the construction of knowledge model is driven by relevant network information and use cases.

The OWL reasoner generates triples by using class model and instances contained within request. By using semantic reasoning together with probabilistic reasoning, we can extend the capabilities of the system to inconsistent and hypothetical information, useful for linking to machine learning algorithms. This requires structured approach to interaction of semantic and probabilistic reasoning.

In our approach, the input model of probabilistic reasoning supports directly statements from the domain model. This is supported by the syntax of MLN, where ontological facts can be represented using logical syntax together with statement the likelihood of which is to be evaluated. The output of probabilistic reasoning is thus guaranteed to be compatible with domain model, and can be added to ontology. A test system combining semantic reasoner and MLN is described in Section IV.

B. Effect modelling and operation-request mapping

We shall next describe an example of how ontology constructs defined above (Figure 2) are used [12]. The approach allows linking effects of operations and requests in the ontology to each other by means of human-defined Key Performance Indicator (KPI) mappings and network measurement effects (Figure 2). The mappings provide dependencies (correlations) between KPI effects, a crucial part of the ontology for mapping. For example, effect dependencies might be used to infer that two functions, that are monitoring thresholds of different KPIs, are both associated with the same goal, since the KPI effects are positively correlated (increasing or decreasing simultaneously). Thus, both of the functions would be valid solutions to a request having similar effect as an objective. More details about the effect modelling is defined in [12].

As the Figure 2 depicts, the reasoner uses effects of agent operations, incoming requests, and network measurements. Using these as input, reasoner first performs inference to find relationships between effects [12]. Next, reasoner maps operations to requested effects with Equation 1. The rule is a Horn clause like rule written in Semantic Web Rule Language (SWRL) and is supported by the OWL reasoner [21]. The rule indicates that if the effect of an operation ($?oe$) is dependent on the effect of a request ($?re$), the operation satisfies (is able to produce) $?re$.

$$\begin{aligned} &Operation(?op) \sqcap Request(?req) \sqcap \\ &hasEffect(?op, ?oe) \sqcap hasEffect(?req, ?re) \sqcap \\ &hasDependency(?re, ?oe) \Rightarrow satisfies(?op, ?re) \end{aligned} \quad (1)$$

Eventually, the rule binds relevant operations into effects of a request. An effect in the request may be bound with multiple operations that produce the effect. Analogously, a particular operation may satisfy several effects in the request. Let us consider an example in which user request is mapped to effects for increasing a KPI value (effect re_1), monitoring the result (re_2), and rollback of the operation (re_3) in case the re_1 is not achieved. The ontology may contain operations that fulfil one of the effects or complex operations that fulfil multiple requested effects, such as a function that executes a configuration and outputs a boolean value whether the objective is achieved. Table II illustrates possible operations mapped to the the described request. As it can be seen, five operations are mapped to one or several effects of the request.

Operation	Satisfies effect
op_1	re_1
op_2	re_2
op_3	re_1, re_2
op_4	re_3
op_5	re_1, re_2, re_3

TABLE I
OPERATIONS MAPPED TO ONE OR SEVERAL EFFECTS OF A REQUEST.

The operation-request mappings may now be processed to obtain combinations of operations that fulfil the whole request. With SPARQL queries, we generate a list of responses that satisfy the request. In the given example, we get three sets of composed operations: $\{\{op_1, op_2, op_4\}\{op_3, op_4\}\{op_5\}\}$.

With multiple responses for a request, a ranking method is needed to select the best response. We address this issue by allowing classification of effects as primary or secondary effects. A primary effect is analysed and scored, whereas a secondary effect only needs to be executable. In the earlier example, an obvious primary effect would be re_1 as it is the actual objective of the request. Operations that are mapped to primary effects may be analysed by using historical data of similar operations in order to define the success ratios of the corresponding operations. The success ratios can then be used to determine the best response for a request.

Based on the operation analysis, the final step is executing the adequate operations of agents with specific sets of parameters.

C. Agents in network management

We shall next discuss our approach from the viewpoint of using agents in network management.

An important advantage of the proposed approach is flexibility. If there is a change either in the NM capabilities or telecommunications functionalities managed by NM, it can be reflected automatically in composition, provided that it has been described in the domain model. Similarly, up-to-date

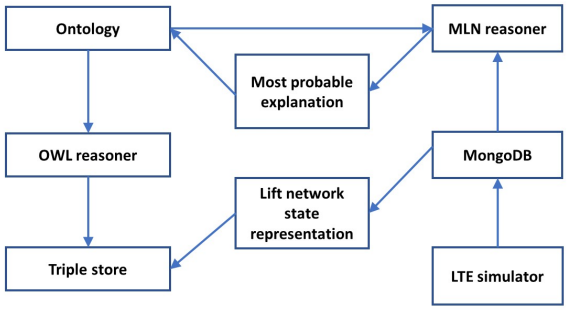

Fig. 3. Information flows in test system. For simplicity SON function and information bus have been omitted from the figure. As discussed in the text, lifting network state directly and importing it into ontology ABox via MLN are mechanisms which may be used depending on the use case.

network status can be used in composition with reasoning and not just in the behaviour of individual agents.

The agents used in composition can be traditional complex SON-style agents, microservices for composing "virtual" agents, or a mixture of the two types [9]. The ontology model for composition assumes metadata about agents [22]. It is not necessary to semantically model the functioning of the agents, only the operations they provide.

The stand-alone agents or software modules used in composition of virtual agents may employ a variety of machine learning algorithms according to the use case. The output of machine learning needs to be semantically integrable to domain model concepts. Consequently, the domain model is important for "sanity checks" of output of probabilistic reasoning. As discussed earlier, e.g. Markov Logic Networks allows for inclusion of "hard facts" in probabilistic reasoning, guaranteeing consistency.

Agent composition can be executed once as a response to incoming request, or run for a period of time (activated) . Requests can be high-level goals from human users, for example. Reasoner and domain models map these to technical parameters using agents as tools. Requests can also be anomalies. In this case, composition identifies anomaly detection and recovery planning functionalities [9] required for processing, composing a virtual agent.

IV. Test system

We are using the test system shown in Figure 3 to validate our approach.

Network is represented in the test system by an internal LTE simulator with configuration and Performance Management (PM) Application Programming Interfaces (APIs). The simulator supports research of agents monitoring LTE KPIs such as Channel Quality Indicator (CQI) and Radio Link Failure (RLF) statistics and performing configurations to transceiver (TRX) power and antenna tilt (RET) parameters, for example [12]. We used Capacity and Coverage Optimization (CCO) SON function based on fixed rule set in the test system. Configuration Management (CM) and PM data from simulator are streamed over publish/subscribe bus [23] to information cross-linking functionality [24] which stores resulting information in MongoDB.

In the first test, metadata representation of cross-linked CM and PM data was lifted directly from MongoDB to triple store (AllegroGraph). Lifting was performed by transforming MongoDB JSON data structure into triples. The source data consisted of 1453 data structures — each corresponding to a CM operation performed by the SON function — which resulted in 193,369 triples. This count is based on preprocessing of PM data associated with operations cases. We also tested a version where CM and PM data were lifted to triple store, which lead to quintupling of triple count in our case. With preprocessing, some of the computation needs in query phase can be eliminated.

In the second test, we used MLN reasoner, which can perform probabilistic reasoning on network data and domain model elements (semantic ontology). In essence, MLN performs analysis of likelihood of hypotheses on data, given a set of ground truth statements [11]. The statements with the highest likelihood are then used in semantic reasoning together with domain model. The results of reasoning on amended ontology are then inserted into the triple store for use with SPARQL queries.

The second test with MLN is a simplified version of learning agent architecture in the sense that machine learning capability was performed by the MLN reasoner rather than a learning agent (CCO function in our case). In a future system, the agents would execute machine learning algorithms, and their outputs would be combined with domain model with probabilistic reasoning. This would change some of the detailed information flows in Figure 3, but the end result — combining machine learning with domain model — would be the same.

A demonstration of substituting traditional programming with reasoning was already possible with our test system. The original goal was to implement the query phase or self-operation system [24] as a warm-up exercise, mapping it to agents. It was found that two SPARQL queries from the coordinator were sufficient to achieve the same end result than the previous Clojure software implementation for similarity search. Our preliminary results also indicate that the similarity search with SPARQL queries have the same level of query processing time as the earlier implementation. Since similarity query of self-operation only retrieves and aggregates historical data and query parameters related to metadata, OWL reasoner was not needed for this demonstration.

The test system described above is flexible, since it allows for routing of information in multiple ways. For the first test, we lifted network data from MongoDB directly to triple store. The second test illustrates more advanced reasoning on network state is facilitated by import into OWL ontology.

LTE simulator was run on a laptop, and the rest of the system functionalities on a Linux server. The system specifications for server used in SPARQL substitute for similarity search: four-core i7 with 32 GB of RAM and RAID SCSI

hard discs.

V. Case study for agent mapping

In this Section, we evaluate simple agent mapping with selected use cases with an LTE simulator. The results of such evaluations could be used in the system described in SectionIV by evaluating the results of respective agent executions with MLN, and transferring the top ranking agents to be used in triple store mappings.

A. Scenario description

The simulator environment comprises 20 LTE base stations with 32 LTE macro cells covering an area with a radius of about 5 km. The simulator creates Performance Management (PM) data reports that contain cell level KPIs. The cell level KPIs are aggregations of the measurements made by the user equipments (UEs) that constantly report the experienced signal status to cell they are attached to. The PM data of the cells are reported periodically in 15 minute intervals in simulation time, amounting to 5-6 hours of simulation time per scenario.

We use three network management scenarios for our experiments: coverage problem, local overload, and mobile overload. The scenarios reflect network issues with similar objectives but in different contexts. In all scenarios, users demand higher throughput, but the solutions differ from each other.

In the coverage problem scenario, the UEs are located uniformly in an area where the coverage is insufficient. The objective is to increase the throughput by increasing the TXP of the cells. The second scenario, local overload, has a few hundred UEs located in a small area near one base station hosting three cells. The throughput of these cells should be increased by adjusting the antenna TRX tilt angles (remote electrical tilt, RET) towards the group of UEs. In the third scenario, mobile overload, there are 1000 uniformly located background UEs and two groups of 200 UEs constantly moving in the simulated area causing abrupt load peaks in the cells. This issue should be addressed by balancing the load between the nearby cells.

B. Analysing the SON agents

For each scenario, we have deployed an agent that executes a desired action. A naïve TXP agent increases the TXP of all target cells by 5 dB, with the target of improving coverage. A naïve RET agent reduces the angles of target cells by two to three degrees (downtilt) aiming to improve the capacity near the antenna. The TXP and RET adjustments are part of a Cell Coverage Optimization (CCO) agent but we have separated these operations for our analysis. A Mobility Load Balancing (MLB) agent constantly adjusts the handover parameters of the cells while it is active. The MLB redirects UE connections from an overloaded cell to neighbor cells. In order to get diverse and comparable results, each scenario is tested with all of three operations, even though enhancements are not expected in some cases.

Figure 4 shows the relative changes of the throughput with standard error margins before and after actions on every scenario. As assumed, the best solution for the coverage problem is the TXP agent (17 % increase), for the local overload the RET agent (13 % increase), and for the mobile overload the MLB agent (18 % increase).

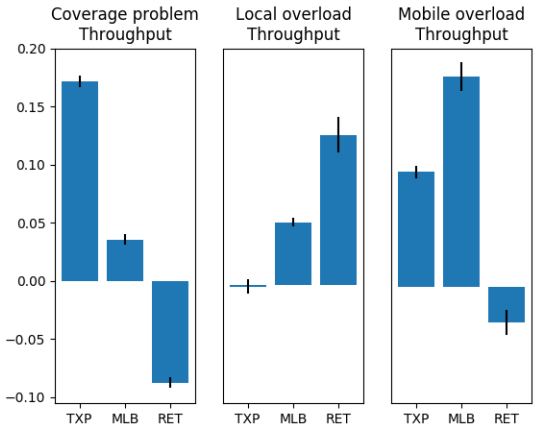

Fig. 4. Scenario-specific relative changes of the throughput values after actions were made.

For further use, we shall use these context-specific results to suggest suitable agents for upcoming network issues in similar contexts.

C. Evaluation of the mapping task

To evaluate the context-specific agent mapping, we create new simulations corresponding to the three contexts, but having diverse parameters. For example, we create coverage problem scenario with more users (2500 instead of 1000), but reduce slightly the coverage holes (more transmission power in antennas). In the same manner, the setups of the other two contexts differ from the earlier simulations by the number of the UEs and by the size, shape and movement direction of the UE groups.

All the new scenarios are simulated multiple times for each agent in order to evaluate extensively the agent mapping. To give an overview of the agent performances in the new scenarios, Figure 5 shows the relative changes they produced in the throughput values.

Based on the relative changes in the throughput values, the table II describes the classification and mapping performance of the agents. The case base data (Figure 4) defines the ”predicted” classification of the agents and the new simulations the ”true” classes. The first row depicts the results, when agents are classified with thresholds to four groups. For example, the agent performance is classified as neutral, if the relative change of the throughput is between 0 % and +5 %. The second row depicts an agent mapping task in which +5 % is set as the threshold for mapping results (hits). Last row shows an agent mapping task, when only the best agents are considered (those improving the throughput more than +15 %).

As the classification and retrieval metrics indicate, the accuracy of the classification improves when the threshold for

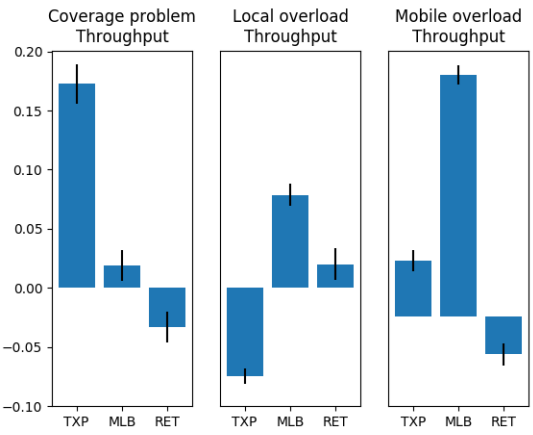

Fig. 5. Scenario-specific relative changes of the throughput for the new simulation scenarios.

"good" performances is higher. The recall values for the two agent mapping tasks indicate high sensitivity in identifying suitable agents, as all possible solutions are present in the search results. Precision values show that some false positive cases are also retrieved. With respect to the Figures 4 and 5 the most probable explanation for the false positive hits are the RET agent in the local overload scenario and the TXP agent in the mobile overload scenario. Clearly, the new local overload scenario is not solved with the downtilt, because the user group is located in a wider area around the base station. Finally, the F_1 score, which is the combination of precision and recall, indicates that the overall performance of the classification is good in this demonstrated use case.

Classes	Thresholds	Precision	Recall	Acc.	F_1
{bad,neutral, ok,good}	0 %, +5 %, +15 %	0.74	0.74	0.74	0.74
{reject,hit}	+5 %	0.73	1.0	0.85	0.85
{reject,hit}	+15 %	0.83	1.0	0.96	0.91

TABLE II
STATISTICS OF THE AGENT MAPPING TASK

All in all, we may conclude that the context-specific agent mapping works in this experimental case study. Moreover, the local overload scenarios demonstrated the challenges in defining contexts; scenarios sharing the same context might actually differ from each other. For this purpose, the contexts should be enriched with semantic contextual metadata that explains the scenarios in a more detailed level. This is an important task that will be addressed in the future research.

VI. Network Management Perspective

We shall make some notes regarding the use of our proposed system in network management.

If network information is directly lifted to triple store without an ontology, its consistency depends on the information model that was lifted. A domain model encapsulated in OWL ontology brings benefits by providing a central coordinating role for information used in the system. Furthermore, the OWL ontology allows advanced reasoning. The domain model needs to be kept up to date with information models of network state and learning algorithms (where applicable).

Ontology constructs result in induced triples (e.g. ABox instance `isA` relations to classes). In principle, ontology could be generated by lifting a suitably formal information model (e.g. SID [25]). Previous work has shown that not all information models are sufficiently consistent or formal for this. We have approached ontology construction from minimal viewpoint, focusing on its roles: facilitation of agent mapping, ensuring consistency of output of machine learning, and providing use case specific reasoning to replace traditional programming.

Human input to ontology can be validated by MLN in the same way as machine learning to avoid inconsistencies in the ontology.

The architecture for inserting the output of machine learning algorithms to probabilistic reasoning, such as MLN is not within the scope of this article and will be accounted for in other publications. Similarly, we have not considered the interfaces between MLN and semantic reasoner.

VII. Summary

We presented an approach where multiple agents or software modules are composed to provide a solution to a request such as a high-level goal. Composition supports software reuse through participation of agents to multiple solutions. Knowledge-based reasoning in composition can by itself substitute software implementations.

First level of functionality is achieved with SPARQL queries to a triple store, sufficient for reasoning over historical information about the network. With semantic reasoner triggered by the query, advanced inferences —- including parameters of the request — are possible. Combined with suitable method for probabilistic reasoning such as Markov Logic Networks, the ontology used by semantic reasoner can be used for semantic integration of machine learning output from diverse algorithms chosen to match data. Probabilistic reasoning is important since the outputs of different algorithms may be conflicting. Provided that the domain model part of the ontology is used in probabilistic reasoning, its output can be introduced to ontology.

All in all, the combination of technologies provides a flexible platform for future network management use cases. The reduction of traditional programming was demonstrated with an example where Clojure implementation of similarity query [24] was replaced with SPARQL calls. Agent composition, in particular together with microservices paradigm, supports efficient use case driven network management.

An important advantage is provided by a support for centralized processing of domain model, query parameters, and network state, avoiding the need to replicate this in individual agents. This is crucial for using composition to reduce the need for software implementations in agents.

REFERENCES

[1] NGMN Alliance and M Iwamura, *NMGN view on 5G architecture*, in Proc. VTC Spring 2015.

[2] V. Ziegler, *et al.*, *Architecture vision for 5G era*, in Proc. IEEE ICC, 2016.

[3] 5GPPP architecture working group, *View on 5G architecture*, URI: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-For-public-consultation.pdf (accessed March 2017).

[4] NGMN, *NGMN 5G white paper*, v1.0, February 2015.

[5] S. Hämäläinen *et al.* (eds.), *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*, John Wiley & Sons, Chichester, England, 2011.

[6] D. Goldszmidt and Y. Yemini, *Delegated Agents for Network Management*, IEEE Communications magazine **36**, p. 66 ff., 1998.

[7] J. O. Kephart and D. M. Chess, *The vision of autonomic computing*, Computer **36**, p. 41 ff., 2003.

[8] NGMN alliance, *NGMN Recommendation on SON and O&M Requirements*, 2008.

[9] V. Räisänen, Agent composition in 5G management and orchestration, in Proc. CNSM 2017.

[10] V. Räisänen and H. Tang, *Knowledge Modeling for Conflict Detection in Self-organized Networks*, in Proc. MONAMI 2011.

[11] K. Apajalahti *et al.*, *Combining ontological modelling and probabilistic reasoning for network management*, J. Ambient Intelligence **9**, p. 63 ff., 2017.

[12] K. Apajalahti *et al.*, *Sharing performance measurement events across domains*, to appear in Proc. IFIP/IEEE IM, 2017.

[13] A. K. Y. Wong, *et al.*, *Ontology Mapping for the Interoperability Problem in Network Management*, J. selected areas in communications **23**, p. 2058 ff., 2005.

[14] J. E. Lopez de Vergara *et al.*, *Ontologies: Giving Semantics to Network Management Models*, IEEE Network, p. 15 ff., 2003.

[15] J. Keeney *et al.*, *Runtime Semantic Interoperability for Gathering Ontology-based Network Context*, in Proc. NOMS, p. 56 ff., 2006.

[16] P. Szilagyi, and S. Novaczki, *An Automatic Detection and Diagnosis Framework for Mobile Communication Systems*, IEEE Transactions on Network and Service Management **9**, p. 184 ff., 2012.

[17] M. Richardson and P. Domingos, *Markov Logic Networks*, Machine learning **62**, p. 107 ff., 2006.

[18] A. Haller, *et al.*, *WSMX — a semantic service-oriented architecture*, in Proc. International conference on Web services (ICWS), 2005.

[19] J. Keeney *et al.*, *Ontology-based Semantics for Composable Autonomic Elements*, in Proc. Workshop of AI in Autonomic Communications at the 19th Intnl Joint Conf. on AI, Edinburgh (2005)

[20] *WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web*, W3C Recommendation, World Wide Web Consortium, Tech. Rep., Aug. 2010, Available: https://www.w3.org/Submission/WSMO-Lite/ (accessed March 2017).

[21] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Recommendation, World Wide Web Consortium, Tech. Rep., May 2004. URI: https://www.w3.org/Submission/SWRL/ (accessed March 2017).

[22] K. Apajalahti, *Multivariate Time Series-Based Annotation Process for Semantic IoT Services*, in preparation.

[23] V. Kojola *et al.*, *Distributed Computing of Management Data in a Telecommunications Network*, in Proc. MONAMI 2016.

[24] H. Tang, *et al.*, *Automatic Definition and Application of Similarity Measures for Self-Operation of Network*, in Proc. MONAMI 2016.

[25] TMForum, *GB922, Information framework (SID) R16.5.1*, URI: http://https://www.tmforum.org/information-framework-sid/ (accessed March 2017).