Artificial Intelligence 256 (2018) 68–104

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Reactive multi-context systems: Heterogeneous reasoning in dynamic environments

Gerhard Brewka [a], Stefan Ellmauthaler [a], Ricardo Gonçalves [b], Matthias Knorr [b,*], João Leite [b], Jörg Pührer [a,*]

[a] Institute of Computer Science, Leipzig University, Germany
[b] NOVA LINCS & Departamento de Informática, Universidade NOVA de Lisboa, Portugal

ARTICLE INFO

Article history:
Received 9 December 2016
Received in revised form 23 November 2017
Accepted 24 November 2017
Available online 5 December 2017

Keywords:
Heterogeneous knowledge
Stream reasoning
Knowledge integration
Reactive systems
Dynamic systems

ABSTRACT

Managed multi-context systems (mMCSs) allow for the integration of heterogeneous knowledge sources in a modular and very general way. They were, however, mainly designed for static scenarios and are therefore not well-suited for dynamic environments in which continuous reasoning over such heterogeneous knowledge with constantly arriving streams of data is necessary. In this paper, we introduce reactive multi-context systems (rMCSs), a framework for reactive reasoning in the presence of heterogeneous knowledge sources and data streams. We show that rMCSs are indeed well-suited for this purpose by illustrating how several typical problems arising in the context of stream reasoning can be handled using them, by showing how inconsistencies possibly occurring in the integration of multiple knowledge sources can be handled, and by arguing that the potential non-determinism of rMCSs can be avoided if needed using an alternative, more skeptical well-founded semantics instead with beneficial computational properties. We also investigate the computational complexity of various reasoning problems related to rMCSs. Finally, we discuss related work, and show that rMCSs do not only generalize mMCSs to dynamic settings, but also capture/extend relevant approaches w.r.t. dynamics in knowledge representation and stream reasoning.

1. Introduction

Fueled by initiatives such as the Semantic Web, Linked Open Data, and the Internet of Things, among others, the wide and increasing availability of machine-processable data and knowledge has prepared the ground and called for a new class of dynamic, rich, knowledge-intensive applications. Such new applications require automated reasoning based on the integration of several heterogeneous knowledge bases – possibly overlapping, independently developed, and written in distinct languages with different semantic assumptions – together with data/event streams produced by sensors and detectors, to be able to support automation and problem-solving, to enforce traceable and correct decisions, and to facilitate the internalization of relevant dynamic data and knowledge into such heterogeneous knowledge bases.

Consider a scenario where Dave, an elderly person suffering from dementia, lives alone in an apartment equipped with various sensors, e.g., smoke detectors, cameras, and body sensors measuring relevant body functions (e.g., pulse, blood pres-

* Corresponding authors.
E-mail addresses: brewka@informatik.uni-leipzig.de (G. Brewka), ellmauthaler@informatik.uni-leipzig.de (S. Ellmauthaler), rjrg@fct.unl.pt (R. Gonçalves), mkn@fct.unl.pt (M. Knorr), jleite@fct.unl.pt (J. Leite), puehrer@informatik.uni-leipzig.de (J. Pührer).

https://doi.org/10.1016/j.artint.2017.11.007
0004-3702/

sure, etc.). An assisted living application in such a scenario could leverage the information continuously received from the sensors, together with Dave's medical records stored in a relational database, a biomedical health ontology with information about diseases, their symptoms and treatments, represented in some description logic, some action policy rules represented as a non-monotonic logic program, to name only a few, and use it to detect relevant events, suggest appropriate action, and even raise alarms, while keeping a history of relevant events and Dave's medical records up to date, thus allowing him to live on his own despite his condition. After detecting that Dave left the room while preparing a meal, the system could alert him in case he does not return soon, or even turn the stove off in case it detects that Dave fell asleep, not wanting to wake him up because his current treatment/health status values rest over immediate nutrition. Naturally, if Dave is not gone long enough, and no sensor shows any potential problems (smoke, gas, fire, etc.), then the system should seamlessly take no action.

Given the requirements posed by novel applications such as the one just described, the availability of a vast number of knowledge bases – written using many different formalisms – and the relevance of streams of data/events produced by sensors/detectors, modern research in knowledge representation and reasoning faces two fundamental problems: dealing with the **integration** of heterogeneous data and knowledge, and dealing with the **dynamics** of such novel knowledge-based systems.

Integration. The first problem stems from the availability of knowledge bases written in many different languages and formats developed over the last decades, from the rather basic ones, such as relational databases or the more recent triplestores, to the more expressive ones, such as ontology languages (e.g., description logics), temporal and modal logics, non-monotonic logics, or logic programs under answer set semantics, to name just a few. Each of these formalisms was developed for different purposes and with different design goals in mind. Whereas some of these formalisms could be combined to form a new, more expressive formalism, with features from its constituents – such as dl-programs [16] and Hybrid MKNF [35] which, to different extent, combine description logics and logic programs under answer set semantics – in general this is simply not feasible, either due to the mismatch between certain assumptions underlying their semantics, or because of the high price to pay, often in terms of complexity, sometimes even in terms of decidability. It is nowadays widely accepted that there simply is no such thing as a single universal, general purpose knowledge representation language.

What seems to be needed is a principled way of integrating knowledge expressed in different formalisms. Multi-context systems (MCSs) provide a general framework for this kind of integration. The basic idea underlying MCSs is to leave the diverse formalisms and knowledge bases untouched, and to use so-called bridge rules to model the flow of information among different parts of the system. An MCS consists of reasoning units – called contexts for historical reasons [27] – where each unit is equipped with a collection of bridge rules. In a nutshell, the bridge rules allow contexts to "listen" to other contexts, that is to take into account beliefs held in other contexts.

Bridge rules are similar to logic programming rules (including default negation), with an important difference: they provide means to access other contexts in their bodies. Bridge rules not only allow for a fully declarative specification of the information flow, but they also allow information to be modified instead of being just passed along as is. Using bridge rules we may translate a piece of information into the language/format of another context, pass on an abstraction of the original information, leaving out unnecessary details, select or hide information, add conclusions to a context based on the absence of information in another one, and even use simple encodings of preferences among parent contexts.

MCSs went through several development steps until they reached their present form. Advancing work in [26,34] aiming to integrate different inference systems, monotonic heterogeneous multi-context systems were defined in [27], with reasoning within as well as across monotonic contexts. The first, still limited attempts to include non-monotonic reasoning were done in [39,12], where default negation in the rules is used to allow for reasoning based on the *absence* of information from a context.

The non-monotonic MCSs of [9] substantially generalize previous approaches, by accommodating *heterogeneous* and both *monotonic* and *non-monotonic* contexts. Hence, they are capable of integrating, among many others, "typical" monotonic logics like description logics or temporal logics, and non-monotonic formalisms like Reiter's default logic, logic programs under answer set semantics, circumscription, defeasible logic, or theories in autoepistemic logic. The semantics of nonmonotonic MCSs is defined in terms of equilibria: a belief set for each context that is acceptable for its knowledge base augmented by the heads of its applicable bridge rules.

More recently, the so-called managed MCSs (mMCSs) [10] addressed a limitation of MCSs in the way they integrate knowledge between contexts. Instead of simply *adding* the head of an applicable bridge rule to the context's knowledge base, which could cause some inconsistency, mMCSs allow for operations other than addition, such as, for instance, *revision* and *deletion*, hence dealing with the problem of consistency management within contexts.

Dynamics. The second problem stems from the shift from static knowledge-based systems that assume a one-shot computation, usually triggered by a user query, to open and dynamic scenarios where there is a need to react and evolve in the presence of incoming information.

Indeed, traditional knowledge-based systems – including the different variants of MCSs mentioned above – focus entirely on static situations, which is the right thing for applications such as for instance expert systems, configuration or planning problems, where the available background knowledge changes rather slowly, if at all, and where all that is needed is the