NAKED: N-Ary Graphs from Knowledge Bases Expressed in Datalog±
Bruno Yun, Madalina Croitoru, Srdjan Vesic, Pierre Bisquert

To cite this version:
Bruno Yun, Madalina Croitoru, Srdjan Vesic, Pierre Bisquert. NAKED: N-Ary Graphs from Knowledge Bases Expressed in Datalog±. AAMAS 2019, May 2019, Montréal, Canada. lirmm-02180395

HAL Id: lirmm-02180395
https://hal-lirmm.ccsd.cnrs.fr/lirmm-02180395
Submitted on 11 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
NAKED: N-Ary Graphs from Knowledge Bases Expressed in Datalog±

Bruno Yun*¹, Madalina Croitoru†¹, Srdjan Vesic‡², and Pierre Bisquert§²

¹INRIA GraphIK, Université de Montpellier, France
²Department of Mechanical Engineering, L′ÉX University
³IATE, INRA, INRIA GraphIK, France

Abstract

In this demonstration paper, we introduce NAKED: a new generator for n-ary logic-based argumentation frameworks instantiated from inconsistent knowledge bases expressed using Datalog±. The tool allows to import a knowledge base in DLGP format, generate, visualise and export the corresponding argumentation hypergraph. We show its application on a use-case from the NoAW project.

Keywords: Logic-based Argumentation; Datalog±; Agent Reasoning

1 The NAKED’s Timeliness

This demonstration paper will present NAKED, a hypergraph generator [22] that uses knowledge bases (KB) expressed in Datalog± [12]. We place ourselves in the context of multi-agent KB expressed in Datalog± [12]. We place ourselves in the context of multi-agent argumentation systems [13], and, more precisely, logic-based argumentation systems (i.e. argumentation systems that employ arguments built over a logical KB). The application setting we will consider is issued from the NoAW H2020 project that aims for approaches to turn agricultural waste into ecological and economic assets.

In the setting where data are gathered from multiple sources or captors, the resulting KB is often inconsistent, i.e. conflicts may appear between the several pieces of information. Since the classical logical inference mechanism does not work in presence of inconsistencies, many inconsistent-tolerant reasoning techniques and inferences were developed to handle inconsistent KBs. Argumentation is such a reasoning method that is based on building arguments

*yun@lirmm.fr
†croitoru@lirmm.fr
‡vesic@cril.fr
§pierre.bisquert@inra.fr
and attacks such that the attacks model the intrinsic conflicts of the KB. This method allows to entail meaningful information from the conclusions of particular sets of arguments. The set of arguments and the corresponding set of attacks is referred to as an argumentation framework (AF). The AFs are usually represented as directed graphs where nodes are arguments and edges between nodes are attacks. However, instantiating such AFs from logic formalisms have been shown to have limitations such as the exponential number of arguments w.r.t. the size of the KB. In order to fix this limitation, several solutions were proposed, which consist in either rewriting the KB prior to the instantiation or filtering the arguments and using sets of attacking arguments by using an n-ary attack relation between arguments. In the NAKED tool, we adopt a novel approach and instantiate the framework of Nielsen and Parsons which allows us to avoid the explosion of the number of arguments.

Classically, reasoning with argumentation graphs consists of either finding extensions (the maximal sets of arguments that do not attack each other and defend themselves as a group from all incoming attacks) or a ranking arguments from the most to the least acceptable. As a result, most of the past work has been focused, amongst others, on optimising the efficiency of the extension finding procedures, on the investigation of various extension and ranking-based notions and on the investigation of desirable properties of logic based instantiations.

There are few practical tools that allow to generate an AF from a given KB. Furthermore, the few available tools for reasoning using argumentation over inconsistent logical KBs either do not allow further tool interoperability (allowing their output argumentation graph to be loaded in other tools) or do not scale up for a practical use.

Our workflow will enable any data engineer to (1) input a KB in the well-known DLGP format for Datalog±, (2) generate an argumentation hypergraph that instantiate the framework of Nielsen and Parsons, (3) interact with the graph representation by allowing arguments re-positioning, (4) observe a specific argument by highlighting the corresponding argument and its attackers in different colours and (5) export the generated argumentation hypergraph in the DOT format for a better tool interoperability. All of these functions could be useful for a non computer science expert who wants to reason over an inconsistent KB in a particular domain using argumentation. It could also be useful for investigating the theoretical properties of the graph based representation of the generated AF. A presentation video explaining all of the features of NAKED is available online at: https://youtu.be/q54iNWBZ9dY.
2 Using the NAKED tool

NAKED is a tool that assists domain experts and argumentation developers in the specification, visualisation and export of logic-based AFs built over the Datalog± language.

2.1 Agent Techniques: Logic Argumentation

Let us first make a note about the logical language used for instantiating the KBs. Existential rules (whose computationally decidable subclasses are usually referred to as Datalog±) have been recently investigated on the Semantic Web for their generalisation w.r.t. Description Logic fragments [26]. It has been shown [14] that using argumentation techniques over inconsistent existential rules KBs yields extensions logically equivalent to the maximally consistent subsets of the KB, called repairs [19]. Using argumentation over existential rules has been shown to be of practical interest over existing repair based approaches [17]. Argumentation for handling inconsistency tolerant semantics enhance the human interaction [4], are used in food science applications [11, 3] or allow for alternative computation methods [27]. Such techniques have been shown to have implications w.r.t. human reasoning and bias detection [10].

An existential rule KB = (F, R, N) is composed of a finite set of facts F (such as \{packaging(a)\} representing the fact that the object a is a packaging), a set of rules R (such as \{∀X(packaging(X) ∧ has(X, plasticFilm) → pollute(X))\} representing the implication that a packaging that has a plastic film is polluting the environment) and a set of negative constraints N (such as \{∀X(pollute(X) ∧ protectEnv(X) → ⊥)\} representing that a certain packaging cannot both protect the environment and pollute it at the same time). The constraints are used to express negative knowledge about the world. In the considered setting, rules and constraints act as an ontology used to “access” different data sources. Therefore, we suppose that all of the inconsistencies come from the facts and that the set of rules is compatible with the set of negative constraints, i.e. the union of those two sets is satisfiable [19].

Example [Datalog± KB] In this KB, a packaging a with a plastic film is said to protect the environment. However, since the possession of a plastic film leads to pollution, this KB is thus inconsistent. Formally, KB = (F, R, N) is such that:

- \(F = \{packaging(a), has(a, plasticFilm), protectEnv(a)\}\)
- \(R = \{∀X(packaging(X) ∧ has(X, plasticFilm) → pollute(X))\}\)
- \(N = \{∀X(pollute(X) ∧ protectEnv(X) → ⊥)\}\)

Starting from an inconsistent existential rule KB, we generate the arguments and the attacks corresponding to the KB. An argument in Datalog± is either a fact or built upon other facts. The Skolem chase coupled with the use of decidable classes of Datalog± ensures the finiteness of the AF proposed (following
from [7]. The attack considered is a particular undermining: a set of arguments $S$ attacks $a$ if and only if the union of the conclusions of all arguments in $S$ and an element of the support of $a$ entails a negative constraint. Note that the attack relation is not symmetric.

Example[Cont’d Example 2.1] We have six attacks on the following four arguments (represented in Figure 1):

- $A_0 : has(a, plasticFilm)$
- $A_1 : protectEnv(a)$
- $A_2 : [A_3, A_1] \rightarrow pollute(a)$
- $A_3 : packaging(a)$

An example of attack is $(\{A_1, A_3\}, A_0)$.

The AF above outputs a set of preferred extensions [22] equivalent to the repairs [19] of the KB (i.e. the maximal with respect to inclusion consistent sets of facts).

2.2 Usability Scenarios

We consider two usability scenarios of NAKED. All of these scenarios are easily employed using NAKED.

**Scenario 1** We consider the task of a specialist inputting an inconsistent KB of his or her expertise and wanting to find the maximally consistent point of views. Please note that tools for assisting non domain experts in building such KBs without computer expertise exists [13]. Finding maximally con-
sistent point of views (or repairs) consists in computing all maximal subsets of facts that do not trigger any negative constraints of $\mathcal{KB}$. There are three repairs: $\{\text{packaging}(a), \text{has}(a, \text{plasticFilm})\}$, $\{\text{packaging}(a), \text{protectEnv}(a)\}$ and $\{\text{has}(a, \text{plasticFilm}), \text{protectEnv}(a)\}$.

**Scenario 2**  We consider the task of an argumentation expert that wants to generate argumentation hypergraphs for benchmarking purposing. Although efficient algorithms that compute extensions exists for argumentation hypergraphs [21], there is a lack of such graphs. Our tool provides a DOT format output which enables interoperability with many graph tools.

**Acknowledgement**

We were supported by the H2020 NoAW project (ID 688338).

**References**


