

Contextual Reasoning for Scene Generation

Technical Report

Loris Bozzato¹, Thomas Eiter², Rafael Kiesel², and Daria Stepanova³

¹ Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy

² Institute of Logic and Computation, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria

³ Bosch Center for Artificial Intelligence, Renningen, Germany

Abstract. We present a continuation to our previous work, in which we developed the MR-CKR framework to reason with knowledge overriding across contexts organized in multi-relational hierarchies. Reasoning is realized via ASP with algebraic measures, allowing for flexible definitions of preferences. In this paper, we show how to apply our theoretical work to real autonomous-vehicle scene data. Goal of this work is to apply MR-CKR to the problem of generating challenging scenes for autonomous vehicle learning. In practice, most of the scene data for AV learning models common situations, thus it might be difficult to capture cases where a particular situation occurs (e.g. partial occlusions of a crossing pedestrian). The MR-CKR model allows for data organization exploiting the multi-dimensionality of such data (e.g., temporal and spatial). Reasoning over multiple contexts enables the verification and configuration of scenes, using the combination of different scene ontologies. We describe a framework for semantically guided data generation, based on a combination of MR-CKR and Algebraic Measures. The framework is implemented in a proof-of-concept prototype exemplifying some cases of scene generation.

1 Introduction and motivation

Testing and evaluation are important steps in the development and deployment of Automated Vehicles (AVs). To comprehensively evaluate the performance of AVs, it is crucial to test the AVs' perception systems in safety-critical scenarios, which rarely happen in naturalistic driving environment, but still possible in practice. Therefore, the targeted and systematic generation of such corner cases becomes an important problem. Most existing studies focus on generating adversarial examples for perception systems of AVs which are concerned with very simple perturbations in the input (e.g., changing the color or position of a vehicle), whereas limited efforts have been put on the generation of ontology-based and context-specific complex scenes (e.g., child walking a dog in the evening in a rainy weather). This is exactly the problem we want to consider in this micro-project.

Specifically, we define our task of interest as follows: given an existing scene (represented by a scene graph) from a known dataset, we want to generate a new set of scenes that are variations of the current scene and are:

1. **Realistic:** that is, consistent with the ontologies describing objects in the scene (e.g., traffic signs usually do not move);
2. **Interesting:** that is, they satisfy a semantic restriction, which tells us that the scene is for example “dangerous” or challenging for our prediction model (e.g., seeing a cat in the middle of the street requires special action);
3. **Similar:** that is, changing the original scene to the generated scenes requires only small variations.

We propose to use symbolic methods to generate valid and challenging scenes on the base of existing scene graphs and semantic definitions of scenes. In particular, MR-CKR [3,2] is a useful formalism for this. Here, ontological knowledge is contextualized such that in different contexts it may have different interpretations (possibly with non-monotonic effects). This means that MR-CKR can help us in generating *realistic* scenes, since it is capable of handling the background ontologies describing the AC domain. Additionally, we may have different contextualized notions of *interestingness*. MR-CKR also allows us to express this by associating different independent semantic restrictions on scenes within different contexts.

Another benefit of MR-CKR is that it comes with a translation to Answer Set Programming (ASP), which is a declarative programming language that can be used to easily express and efficiently solve hard logical problems.

Apart from realism and interest, we care about similarity. Thus, we need a way to measure how similar the generated scenes are to the original scene that we started from. So-called Algebraic Measures [9] are of great use here. They are a general framework from the field of ASP that allows us to measure quantities associated with solutions. As such, they are also capable of expressing a similarity measure of scenes, which we need.

Our main contributions are as follows:

- We provide a novel framework for semantically guided data generation, which can be adversarial or training data.
- By basing our framework on a combination of MR-CKRs and Algebraic Measures we obtain a highly flexible approach with efficient solving options by employing translations to ASP.
- MR-CKRs allow us (i) to incorporate ontological background knowledge ensuring realism of the generated data and (ii) to contextualize the notion of what makes a generated input interesting.
- Algebraic Measures enable the maximization of similarity between original and generated data.
- Our prototype for scene generation in the domain of AV is intentionally kept minimal but shows promise.

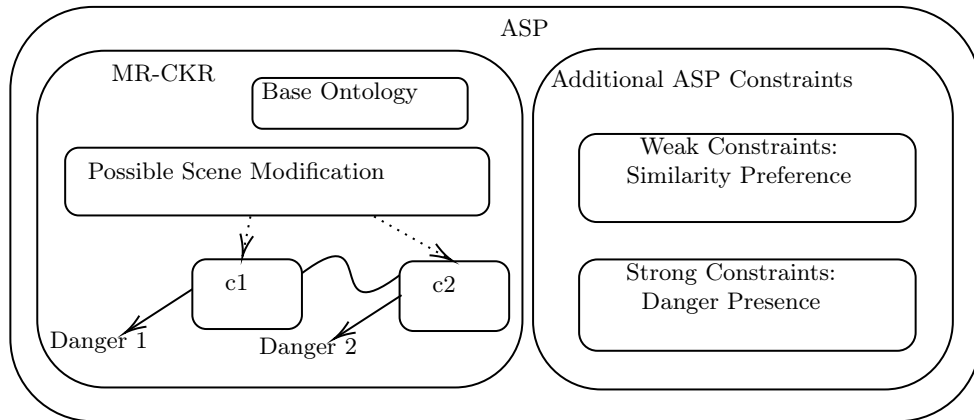


Fig. 1. The general framework for generating (similar) dangerous scenes with ASP according to (possibly) related types of dangers defined by an MR-CKR.

Related Work. The generation of adversarial or challenging examples for neural models is an important problem that gained interest both in industry⁴ and research [13,16,6,10]. Also in these works the generation of inputs that are similar to the original ones and realistic is of importance. However, instead of using symbolic methods to generate new inputs and to verify that the generated inputs are realistic, numerical methods are used here. E.g., [13] uses small numerical perturbations of images, [16] uses an optimization that minimizes the numerical change of the input data such that it leads to a different prediction of the network. The closest work that we found to ours is [10], which generates adversarial text for natural language processing by performing minimal replacements of characters. However, while this optimization for the minimal replacement can be seen as a symbolic approach, no verification of how realistic the newly generated text is, was performed.

2 Framework Overview

Before we go into the technical details of how we generate descriptions of new challenging training scenes, we provide a general structural overview of our framework.

We consider for this, the schema described in Figure 1. Here, we see that we use an MR-CKR to define, on the one hand, the possible scene modifications and on the other hand different contexts, here $c1$ and $c2$, that specify possibilities for a scene to be dangerous/interesting. Optimally, these different types of danger correspond to diagnoses of a neural network engineer for poor performance of the current neural network. For example, in the AV context, we might observe that a car does not stop in the correct location when there is not only a stop

⁴ <https://www.efemarai.com/>

sign but also a stop line marking that specifies where the car should stop. Here, we would therefore want to modify scenes in such a manner that they have both a stop sign and a stop line marking.

Generally, the goal is to generate more scenes that we suspect the network also performs badly on, such that we have adversarial examples that we can use to train the neural network in the hopes of improving its performance on these situations that are hard for it, due to a lack of training data. Given the definitions of danger in different contexts (i.e., based on different diagnosis) that may be related via specialization or otherwise, we can then obtain an equivalent encoding in ASP to obtain models, i.e., generated scenes that are realistic according to the base ontology included in the MR-CKR.

Additionally, we add further ASP constraints to make sure the modifications of the scene are such that the resulting scene is (a) dangerous (using the strong constraints) and (b) as similar as possible to a given starting scene (using the weak constraints that express the algebraic measure).

Putting both things together, we can thus obtain realistic, dangerous scenes that are as similar to the starting scene as possible. On top of that, the different contexts allow us to specify different types of target dangers resulting in one generated scene that includes it per context.

In the following, we substantiate our abstract idea by formalizing how we generate scenes with MR-CKR and measure their similarity with Algebraic Measures.

3 Formalization of scene generation problem in MR-CKR

We begin by introducing formally the MR-CKR framework and we provide a solution making use of MR-CKR in scene generation.

3.1 MR-CKR definition

We assume the customary definitions for description logics (see, e.g., [1] for an introduction). We summarize in the following the main definitions of MR-CKR (as introduced in [2]).

We consider a generic description language \mathcal{L}_Σ based on a DL signature Σ , which is composed of a set of concept names NC, role names NR and individual names NI.

Consider a nonempty set $\mathbf{N} \subseteq \text{NI}$ of *context names*. A *contextual relation* is any strict order $\prec_i \subseteq \mathbf{N} \times \mathbf{N}$ over contexts. A way to define contextual relations is to use *contextual dimensions* [4,14], that is a set of contextual “coordinates” associated to each of the contexts: in the case of scene descriptions, for example, these can represent the time of the day, location type or situation occurring in a scene. The contextual structure, then, is defined from the product of order of features (dimensions) associated to the contexts, corresponding to a contextual relation.

In a MR-CKR, axioms inside contexts can be specified as defeasible (i.e. they can be overridden in case of exceptions) with respect to one of the contextual relations composing the contextual structure.

Definition 1 (r-defeasible axiom). *Given a set \mathcal{R} of contextual relations over \mathbf{N} and a description language \mathcal{L}_Σ , an r-defeasible axiom is any expression of the form $D_r(\alpha)$, where α is an axiom of \mathcal{L}_Σ and $\prec_r \in \mathcal{R}$.*

We allow for the use of r-defeasible axioms in the local language of contexts:

Definition 2 (contextual language). *Given a set of context names \mathbf{N} , for every description language \mathcal{L}_Σ we define $\mathcal{L}_{\Sigma, \mathbf{N}}$ as the extension of \mathcal{L}_Σ where: (i) $\mathcal{L}_{\Sigma, \mathbf{N}}$ contains the set of r-defeasible axioms in \mathcal{L}_Σ ; (ii) $\text{eval}(X, \mathbf{c})$ is a concept (resp. role) of $\mathcal{L}_{\Sigma, \mathbf{N}}$ if X is a concept (resp. role) of \mathcal{L}_Σ and $\mathbf{c} \in \mathbf{N}$.*

Multi-relational CKRs are then composed by a global structure of context based on the contextual relations in \mathcal{R} and a set of DL knowledge bases associated to each of the local contexts.

Definition 3 (multi-relational simple CKR). *A multi-relational simple CKR (sCKR) over Σ and \mathbf{N} is a structure $\mathfrak{K} = \langle \mathfrak{C}, \mathbf{K}_\mathbf{N} \rangle$ where:*

- \mathfrak{C} is a structure $(\mathbf{N}, \prec_1, \dots, \prec_m)$ where each \prec_i is a contextual relation over \mathbf{N} , and
- $\mathbf{K}_\mathbf{N} = \{\mathbf{K}_\mathbf{c}\}_{\mathbf{c} \in \mathbf{N}}$ for each context name $\mathbf{c} \in \mathbf{N}$, $\mathbf{K}_\mathbf{c}$ is a DL knowledge base over $\mathcal{L}_{\Sigma, \mathbf{N}}$.

Example 1. We provide a simple example of MR-CKR to better explain the intended use of defeasible axioms. Consider the sCKR $\mathfrak{K} = \langle \mathfrak{C}, \{\mathbf{K}_1, \mathbf{K}_2\} \rangle$ composed by the following elements:

$$\begin{aligned} \mathfrak{C} &= \{\mathbf{c}_2 \prec_c \mathbf{c}_1\} \\ \mathbf{K}_1 &= \{D_c(Dog \sqsubseteq \neg DangerousAnimal)\} \\ \mathbf{K}_2 &= \{Dog \sqsubseteq DangerousAnimal, Dog(d)\} \end{aligned}$$

Intuitively, we want to recognize that in the more specific context \mathbf{c}_2 , dogs are considered as dangerous animals, thus the more general defeasible axiom in \mathbf{c}_1 is not applied to the instance d of *Dog*.

Interpretations of MR-CKRs are family of DL interpretations associated to each of the contexts.

Definition 4 (sCKR interpretation). *An interpretation for $\mathcal{L}_{\Sigma, \mathbf{N}}$ is a family $\mathfrak{I} = \{\mathcal{I}(\mathbf{c})\}_{\mathbf{c} \in \mathbf{N}}$ of \mathcal{L}_Σ interpretations, such that $\Delta^{\mathcal{I}(\mathbf{c})} = \Delta^{\mathcal{I}(\mathbf{c}')}$ and $a^{\mathcal{I}(\mathbf{c})} = a^{\mathcal{I}(\mathbf{c}')}$, for every $a \in \text{NI}$ and $\mathbf{c}, \mathbf{c}' \in \mathbf{N}$.*

The interpretation of concepts and role expressions in $\mathcal{L}_{\Sigma, \mathbf{N}}$ is obtained by extending the standard interpretation to eval expressions: for every $\mathbf{c} \in \mathbf{N}$,

$\text{eval}(X, \mathbf{c}')^{\mathcal{I}(\mathbf{c})} = X^{\mathcal{I}(\mathbf{c}')}$. We consider the definition of axiom instantiation provided by [3]: given an axiom $\alpha \in \mathcal{L}_{\Sigma}$ with FO-translation $\forall \mathbf{x}. \phi_{\alpha}(\mathbf{x})$, the *instantiation* of α with a tuple \mathbf{e} of individuals in NI, written $\alpha(\mathbf{e})$, is the specialization of α to \mathbf{e} , i.e., $\phi_{\alpha}(\mathbf{e})$, depending on the type of α .

A *clashing assumption* for a context \mathbf{c} and contextual relation r is a pair $\langle \alpha, \mathbf{e} \rangle$ such that $\alpha(\mathbf{e})$ is an axiom instantiation of α , and $\mathbf{c}' \succeq_{-r} \mathbf{c}'' \succ_r \mathbf{c}$. A *clashing set* for $\langle \alpha, \mathbf{e} \rangle$ is a satisfiable set S of ABox assertions s.t. $S \cup \{\alpha(\mathbf{e})\}$ is unsatisfiable.

Definition 5 (CAS-interpretation). A CAS-interpretation is a structure $\mathcal{J}_{CAS} = \langle \mathcal{J}, \bar{\chi} \rangle$ where \mathcal{J} is an interpretation and $\bar{\chi} = \{\chi_1, \dots, \chi_m\}$ such that each χ_i , for $i \in \{1, \dots, m\}$, maps every $\mathbf{c} \in \mathbf{N}$ to a set $\chi_i(\mathbf{c})$ of clashing assumptions for context \mathbf{c} and context relation \prec_i .

Definition 6 (CAS-model). Given a multi-relation sCKR \mathfrak{R} , a CAS-interpretation $\mathcal{J}_{CAS} = \langle \mathcal{J}, \bar{\chi} \rangle$ is a CAS-model for \mathfrak{R} (denoted $\mathcal{J}_{CAS} \models \mathfrak{R}$), if the following holds:

- (i) for every $\alpha \in K_c$ (strict axiom), and $\mathbf{c}' \preceq_* \mathbf{c}$, $\mathcal{I}(\mathbf{c}') \models \alpha$;
- (ii) for every $D_i(\alpha) \in K_c$ and $\mathbf{c}' \preceq_{-i} \mathbf{c}$, $\mathcal{I}(\mathbf{c}') \models \alpha$;
- (iii) for every $D_i(\alpha) \in K_c$ and $\mathbf{c}'' \prec_i \mathbf{c}' \preceq_{-i} \mathbf{c}$, if $\langle \alpha, \mathbf{d} \rangle \notin \chi_i(\mathbf{c}'')$, then $\mathcal{I}(\mathbf{c}'') \models \phi_{\alpha}(\mathbf{d})$.

We provide a *local preference* on clashing assumption sets for each of the relations:

- (LP). $\chi_i^1(\mathbf{c}) > \chi_i^2(\mathbf{c})$, if for every $\langle \alpha_1, \mathbf{e} \rangle \in \chi_i^1(\mathbf{c}) \setminus \chi_i^2(\mathbf{c})$ with $D_i(\alpha_1)$ at a context $\mathbf{c}_1 \succeq_{-i} \mathbf{c}_{1b} \succ_i \mathbf{c}$, some $\langle \alpha_2, \mathbf{f} \rangle \in \chi_i^2(\mathbf{c}) \setminus \chi_i^1(\mathbf{c})$ exists with $D_i(\alpha_2)$ at context $\mathbf{c}_2 \succeq_{-i} \mathbf{c}_{2b} \succ_i \mathbf{c}$ s.t. $\mathbf{c}_{1b} \succ_i \mathbf{c}_{2b}$.

Intuitively, $\chi_i^1(\mathbf{c})$ is preferred to $\chi_i^2(\mathbf{c})$ if $\chi_i^1(\mathbf{c})$ exchanges the “more costly” exceptions of $\chi_i^2(\mathbf{c})$ at more specialized contexts with “cheaper” ones at more general contexts.

Two DL interpretations \mathcal{I}_1 and \mathcal{I}_2 are NI-congruent, if $c^{\mathcal{I}_1} = c^{\mathcal{I}_2}$ holds for every $c \in \text{NI}$. This extends to CAS interpretations $\mathcal{J}_{CAS} = \langle \mathcal{J}, \bar{\chi} \rangle$ by considering all context interpretations $\mathcal{I}(\mathbf{c}) \in \mathcal{J}$.

Definition 7 (justification). We say that $\langle \alpha, \mathbf{e} \rangle \in \chi_i(\mathbf{c})$ is justified for a CAS model \mathcal{J}_{CAS} , if some clashing set $S_{\langle \alpha, \mathbf{e} \rangle, \mathbf{c}}$ exists such that, for every $\mathcal{J}'_{CAS} = \langle \mathcal{J}', \bar{\chi} \rangle$ of \mathfrak{R} that is NI-congruent with \mathcal{J}_{CAS} , it holds that $\mathcal{I}'(\mathbf{c}) \models S_{\langle \alpha, \mathbf{e} \rangle, \mathbf{c}}$. A CAS model \mathcal{J}_{CAS} of a sCKR \mathfrak{R} is justified, if every $\langle \alpha, \mathbf{e} \rangle \in \bar{\chi}$ is justified in \mathfrak{R} .

We define a *model preference* by combining the preferences of the relations: it is a global lexicographical ordering on models where each \prec_i defines the ordering at the i -th position.

- (MP). $\mathcal{J}_{CAS}^1 = \langle \mathcal{J}^1, \chi_1^1, \dots, \chi_m^1 \rangle$ is preferred to $\mathcal{J}_{CAS}^2 = \langle \mathcal{J}^2, \chi_1^2, \dots, \chi_m^2 \rangle$ if

- (i) there exists $i \in \{1, \dots, m\}$ and some $\mathbf{c} \in \mathbf{N}$ s.t. $\chi_i^1(\mathbf{c}) > \chi_i^2(\mathbf{c})$ and not $\chi_i^2(\mathbf{c}) > \chi_i^1(\mathbf{c})$, and for no context $\mathbf{c}' \neq \mathbf{c} \in \mathbf{N}$ it holds that $\chi_i^1(\mathbf{c}') < \chi_i^2(\mathbf{c}')$ and not $\chi_i^2(\mathbf{c}') < \chi_i^1(\mathbf{c}')$.

- (ii) for every $j < i \in \{1, \dots, m\}$, it holds $\chi_j^1 \approx \chi_j^2$ (i.e. (i) or its converse do not hold for \prec_j).

Definition 8 (CKR model). *An interpretation \mathcal{I} is a CKR model of a sCKR \mathfrak{K} (in symbols, $\mathcal{I} \models \mathfrak{K}$) if: (i) \mathfrak{K} has some justified CAS model $\mathcal{I}_{CAS} = \langle \mathcal{I}, \bar{\chi} \rangle$; (ii) there exists no justified $\mathcal{I}'_{CAS} = \langle \mathcal{I}', \bar{\chi}' \rangle$ that is preferred to \mathcal{I}_{CAS} .*

Example 2. Using the semantics mechanism shown above, we can show how to interpret the sCKR in previous example. In particular, we can consider the CAS-interpretation $\mathcal{I}_{CAS} = \langle \mathcal{I}, \bar{\chi} \rangle$ where $\chi(c_2) = \{\langle Dog \sqsubseteq \neg DangerousAnimal, d \rangle\}$. This implies that \mathcal{I}_{CAS} is a CAS-model if the defeasible axiom of c_1 is not applied to the only *Dog* in c_2 , as expected. Note that such CAS-model is also justified, since the clashing assumption admits the clashing set $\{Dog(d), DangerousAnimal(d)\}$: thus, considering that no other alternative CAS-model that is minimal with respect to the preference can be defined, the considered interpretation is also a CKR-model. The preference defined above is useful to prefer defeasible axioms in the most specific contexts: for example, if $Dog \sqsubseteq DangerousAnimal$ in c_2 was defined as defeasible (w.r.t. the same contextual relation of the above defeasible axiom), the context below c_2 would have preferred the more specific axiom and thus the interpretations where exceptions are made on the more generic axiom of the upper context c_1 .

As a method to implement reasoning on MR-CKR, in [2] we provided a translation for MR-CKRs to ASP logic programs: in particular, we showed that such translation can be used to reason on instance checking and query answering in a given context.

3.2 Scene generation in MR-CKR

Following the intuitive structure of Figure 1, the role of MR-CKR in our architecture is to define the logical constraints of the scenes we want to generate, on the base of a common scene ontology.

Given its multi-contextual structure, the MR-CKR is useful to provide a complex representation (a contextualization) of the contents of the base scene and its modifications towards the different diagnoses of interest.

With respect to the second aspect, the basic organization of contexts can be defined as in Figure 2.

The contexts of this structure are related by a contextual relation \succ_{sim} , denoting the relation of similarity: the upper context *Exchange* contains, in form of defeasible axioms, the axioms that can be modified in the diagnosis scenes. In the *Base* context, we assume to have the description of the base scene and the base axioms of the scene description ontology. The contexts *Diagnosis-1*, \dots , *Diagnosis-N*, then, provide the different modifications to the base scene we are interested to model. The kind of axioms that are needed to model the different modifications depend on the kind of changes (additions, deletions, etc.) that we want to admit in scene modifications: more detail on such axioms will be provided in the following sections, where we consider specific modifications.

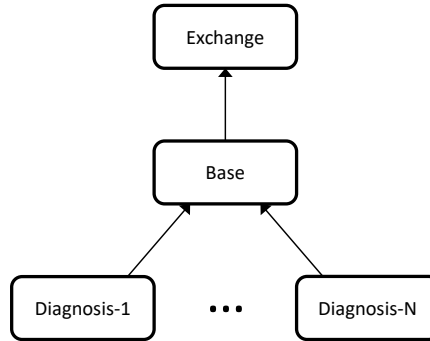


Fig. 2. General structure of contexts for scene modification

With respect to the scene contextualization, we can make use of the multi-relational nature of MR-CKR to further define the context in which the scene take place.

Example 3. An example of such contextualization is shown in Figure 3.

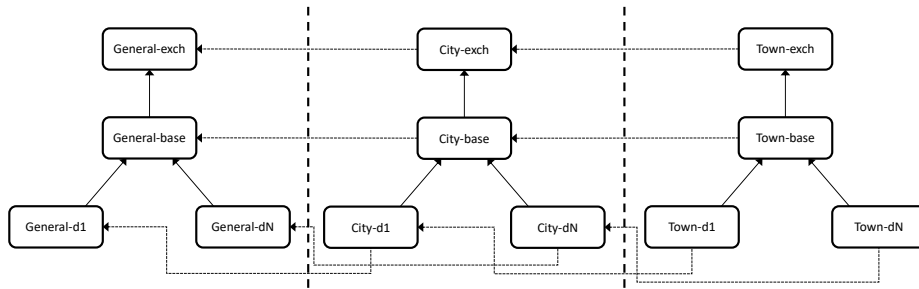


Fig. 3. Example of general structure of MR-CKR for scene representation

In this contextual structure, the relation represented by the horizontal arrows represent the specialization of scenes with respect to the specificity of the location: starting from axioms that are verified for *general* scenes, we can add further logical constraints that are true for *city* scenes and then *town* scenes. Note that such direction is orthogonal to the base contextual structure described above.

After modelling scenes by such framework, we want to use the translation of MR-CKR to ASP in order to generate the possible models of the diagnoses contexts: these models then correspond to alternative generated scenes. However, we now need a method to provide a measure the *similarity* of the generated scenes with respect to the scenes of interest: as we detail in the following sections, this can be easily obtained by means of algebraic measures.

4 Formalization of Similarity using Algebraic Measures

If we want to generate a new scene based on a starting scene, we want to optimize a measure of similarity. For measuring similarity, we can make use of algebraic measures. The intuitive idea behind algebraic measures is that they allow us to measure a quantity associated with an interpretation or a model. In order to allow measuring different quantities in a uniform framework, we use the algebraic structure of *semirings*, which allow for many different forms of computation.

4.1 Preliminaries

We introduce algebraic measures and their necessary preliminaries.

Definition 9 (Monoid). A monoid $\mathcal{M} = (M, \otimes, e_\otimes)$ consists of an associative binary operation \otimes on a set M with neutral element e_\otimes , also called identity element. Here, a binary operation on M is a function $\otimes : M \times M \rightarrow M$ that maps pairs of values from M to a value in M . We write the application of such a binary operation \otimes to a pair (m_1, m_2) of values $m_1, m_2 \in M$ in infix notation $m_1 \otimes m_2$.

A value $e_\otimes \in M$ is a neutral element for a binary operation \otimes on M if for all values $m \in M$ it holds that

$$e_\otimes \otimes m = m = m \otimes e_\otimes.$$

Additionally, a binary operation \otimes on M is associative, if for all $m, m', m'' \in M$ it holds that

$$m \otimes (m' \otimes m'') = (m \otimes m') \otimes m''.$$

Some examples of monoids are

- $\text{STRINGS} = (\{0, 1\}^*, \odot, \varepsilon)$, the set of binary strings with concatenation \odot and empty string ε is a non-commutative, non-idempotent, and non-invertible monoid,
- $\mathcal{P}(A) = (2^A, \cup, \emptyset)$, the set of subsets for a set A with union \cup is a commutative, idempotent and non-invertible monoid,
- $\mathcal{P}(A) = (2^A, \cap, A)$, the set of subsets for a set A with intersection \cap is a commutative, idempotent and non-invertible monoid,
- $\mathbb{Z} = (\mathbb{Z}, +, 0)$, the integers with addition $+$ is a commutative, non-idempotent and invertible monoid.

Based on monoids, we introduce semirings.

Definition 10 (Semiring). A semiring $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ is a nonempty set R equipped with two binary operations \oplus and \otimes , called addition and multiplication, such that

- (R, \oplus) is a commutative monoid with identity element e_\oplus ,
- (R, \otimes) is a monoid with identity element e_\otimes ,

- multiplication left and right distributes over addition, i.e., for all $r, r', r'' \in R$ it holds that

$$\begin{aligned} r \otimes (r' \oplus r'') &= r \otimes r' \oplus r \otimes r'' \\ (r' \oplus r'') \otimes r &= r' \otimes r \oplus r'' \otimes r \end{aligned}$$

- and multiplication by e_{\oplus} annihilates R , i.e., for all $r \in R$ it holds that

$$r \otimes e_{\oplus} = e_{\oplus} = e_{\oplus} \otimes r.$$

Some examples of semirings are

- $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$, the Boolean semiring, with disjunction and conjunction as addition and multiplication,
- $\mathbb{F} = (\mathbb{F}, +, \cdot, 0, 1)$, for $\mathbb{F} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ the semiring of the numbers in \mathbb{F} with addition and multiplication,
- $\mathcal{P}(A) = (2^A, \cup, \cap, \emptyset, A)$, the semiring over the powerset of A with union and intersection, and
- $\mathcal{R}_{\min,+} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$, the min-plus semiring.

Another list of semirings, which is annotated with applications, can be found in [12].

In order to connect the quantitative aspects of semirings and the qualitative ones of logics we use weighted logics. They were initially introduced by [8] in the second order setting. Here, we only introduce the restricted version for propositional logic.

Definition 11 (Syntax). *Let \mathcal{V} be a set of propositional variables and let $\mathcal{R} = (R, \oplus, \otimes, e_{\oplus}, e_{\otimes})$ be a semiring. A weighted (propositional) formula over \mathcal{R} is of the form α given by the grammar*

$$\alpha ::= k \mid v \mid \neg v \mid \alpha + \alpha \mid \alpha * \alpha$$

where $k \in R$ and $v \in \mathcal{V}$.

We can evaluate weighted formulas with respect to an interpretation to obtain a value from the semiring.

Definition 12 (Semantics). *Given a weighted propositional formula α over a semiring $\mathcal{R} = (R, \oplus, \otimes, e_{\oplus}, e_{\otimes})$ and propositional variables from \mathcal{V} as well as an interpretation \mathcal{I} , i.e., a subset of \mathcal{V} , the semantics $\llbracket \alpha \rrbracket_{\mathcal{R}}(\mathcal{I})$ of α over \mathcal{R} w.r.t. \mathcal{I} is defined as follows:*

$$\begin{aligned} \llbracket k \rrbracket_{\mathcal{R}}(\mathcal{I}) &= k \\ \llbracket v \rrbracket_{\mathcal{R}}(\mathcal{I}) &= \begin{cases} e_{\otimes} & v \in \mathcal{I} \\ e_{\oplus} & \text{otherwise.} \end{cases} \quad (v \in \mathcal{V}) \\ \llbracket \neg v \rrbracket_{\mathcal{R}}(\mathcal{I}) &= \begin{cases} e_{\oplus} & v \in \mathcal{I} \\ e_{\otimes} & \text{otherwise.} \end{cases} \quad (v \in \mathcal{V}) \\ \llbracket \alpha_1 + \alpha_2 \rrbracket_{\mathcal{R}}(\mathcal{I}) &= \llbracket \alpha_1 \rrbracket_{\mathcal{R}}(\mathcal{I}) \oplus \llbracket \alpha_2 \rrbracket_{\mathcal{R}}(\mathcal{I}) \\ \llbracket \alpha_1 * \alpha_2 \rrbracket_{\mathcal{R}}(\mathcal{I}) &= \llbracket \alpha_1 \rrbracket_{\mathcal{R}}(\mathcal{I}) \otimes \llbracket \alpha_2 \rrbracket_{\mathcal{R}}(\mathcal{I}). \end{aligned}$$

We define algebraic measures to combine the qualitative language of answer set programs with the quantitative one of weighted logic.

Definition 13 (Algebraic Measure). *An algebraic measure $\mu = \langle \Pi, \alpha, \mathcal{R} \rangle$ consists of an answer set program Π , a weighted formula α , and a semiring \mathcal{R} . Then, the weight of an answer set $\mathcal{I} \in \mathcal{AS}(\Pi)$ under μ is defined by*

$$\mu(\mathcal{I}) = \llbracket \alpha \rrbracket_{\mathcal{R}}(\mathcal{I}).$$

Additionally, the result of an (atomic) query for an atom a from Π is given by

$$\mu(a) = \bigoplus_{\mathcal{I} \in \mathcal{AS}(\Pi), a \in \mathcal{I}} \mu(\mathcal{I}),$$

and the result of the overall weight query of Π is

$$\mu(\Pi) = \bigoplus_{\mathcal{I} \in \mathcal{AS}(\Pi)} \mu(\mathcal{I}).$$

Intuitively, the idea here is that for an algebraic measure $\mu = \langle \Pi, \alpha, \mathcal{R} \rangle$ the semiring \mathcal{R} determines the mode of quantitative computation, Π states the logical background theory that determines which interpretations are solutions and α assigns each answer set \mathcal{I} a weight over the semiring, by performing a calculation over the semiring that depends on the satisfaction of atomic formulas in the interpretation \mathcal{I} .

4.2 Similarity of Scenes

Broadly speaking, we can modify a scene in three ways:

- (i) Object Addition,
- (ii) Object Deletion, or
- (iii) Object Modification.

Our goal is to assign these actions a cost. Then we can compute how costly it is to obtain one scene from another by performing a sequence of actions and summing up their costs. The higher this cost is the lower is the similarity between the two scenes.

The effect of (i) and (ii) are clear and do not allow for many suboptions. The only possibility in this direction is to differentiate between the addition/deletion of objects of different complexities, assigning higher costs to more complex objects. This option can be explored more later if necessary. For now, we assume that an addition and deletion have fixed costs $\text{cost}(\text{Add})$ and $\text{cost}(\text{Del})$, respectively.

For modification, however, we have different options:

- Displacement (e.g., to force an overlap between two objects),
- Class Variation, and
- Property Variation (i.e., removing a property, adding a property, or modifying the value of a property).

For class variation it makes sense to add some restrictions, otherwise, we could perform an object modification to achieve a deletion and addition in the same step. For this, we assert that it is only possible to exchange class C by class C' if they share a reasonable superclass C_s , i.e., $C \sqsubseteq C_s$ and $C' \sqsubseteq C_s$ must hold for a superclass C_s that is not *owl:Thing* or something of the sort. Here, we choose a list of reasonable superclasses including *Vehicle*, *Animal*, and *StreetSign*.

The costs we assign for each of the modifications are as follows:

- **Displacement:** Either the distance between the former and the latter location multiplied by a constant factor or a constant cost $\text{cost}(Disp)$.
- **Class Variation:** When replacing class C by C' with lowest common superclass C_s , the cost is the minimal number of DL-axioms that need to be used to derive that $C \sqsubseteq C_s$ and $C' \sqsubseteq C_s$. For example, consider the following set of axioms:

$$\begin{aligned} Hedgehog &\sqsubseteq DangerousAnimal, \\ Tiger &\sqsubseteq DangerousAnimal \end{aligned}$$

Here, for $C = Hedgehog$, $C' = Tiger$, and $C_s = DangerousAnimal$, we need one DL-Axiom to derive $Hedgehog \sqsubseteq DangerousAnimal$ and one DL-Axiom to derive $Tiger \sqsubseteq DangerousAnimal$. Thus, replacing a tiger by a hedgehog would result in a cost of 2.

- **Property Variation:** We assign deletion, addition, and modification a constant value $\text{cost}(PDel)$, $\text{cost}(PAdd)$, and $\text{cost}(PMod)$, each. It makes sense to have $\text{cost}(PDel) = \text{cost}(PAdd) > \text{cost}(PMod)$.

Given these assumptions on how we measure the similarity, we can proceed with the modelling of its measurement.

4.3 Measuring Similarity with Algebraic Measures

Algebraic measures consist of three parts. The program, specifying the logical constraints, the weighted formula specifying how we measure the weight, and the semiring specifying what kind of weight we measure. For the semiring it makes sense to use $\mathcal{R}_{\min,+}$. Then, we can sum up different costs that are incurred by an interpretation and choose the minimum possible cost, if there are different options.

The logical constraints are mainly given by the MR-CKR. However, in order to specify a weighted formula, we need to be able to use some atomic formulas that tell us which modifications were performed to arrive at the scene in the model. Thus, we ensure that the signature of the program includes the following predicates:

- $\text{addition}(C, I)$, denoting additions of individual I to class C ;
- $\text{deletion}(C, I)$, denoting deletion of individual I from class C ;
- $\text{displacement}(I)$ (resp. $\text{displacement}(I, D)$), denoting displacement of individual I (resp. by distance D);

- $\text{classVar}(I, C, C')$, denoting that individual I was in class C in the original scene but is now in class C' ;
- $\text{propertyVar}(I, T)$, denoting that a property of individual I underwent a modification of type T .

Using the above mentioned predicates, we can easily specify the weighted formula α_{cost} that measures the cost of transforming the original scene into the modified one, as follows:

$$\begin{aligned}
& \prod_{\text{class } c, \text{individual } i} (\text{addition}(c, i) * \text{cost}(Add) + \neg\text{addition}(c, i)) \\
& * \prod_{\text{class } c, \text{individual } i} (\text{deletion}(c, i) * \text{cost}(Del) + \neg\text{deletion}(c, i)) \\
& * \prod_{\text{individual } i} (\text{displacement}(i) * \text{cost}(Disp) + \neg\text{displacement}(i)) \\
& * \prod_{\text{classes } c, c', \text{individual } i} (\text{classVar}(i, c, c') * \text{dist}(c, c') + \neg\text{classVar}(i, c, c')) \\
& * \prod_{\text{individual } i, t \in \{PDel, PAdd, PMod\}} (\text{propertyVar}(i, t) * \text{cost}(t) + \neg\text{propertyVar}(i, t))
\end{aligned}$$

One line takes care of the cost for one modification type each. Here, $\text{dist}(c, c')$ denotes the distances between two classes c and c' as explained above. We can compute these statically for each of the contexts.

Example 4. Consider for example the interpretation

$$\mathcal{I} = \{\text{addition}(\text{RollingContainer}, i_1), \text{deletion}(\text{Child}, i_2), \text{deletion}(\text{Child}, i_3)\}.$$

Intuitively, this means that we add the object i_1 to the concept *RollingContainer* and remove the objects i_2 and i_3 from the concept *Child*.

We expect that this interpretation comes with a cost of $\text{cost}(Add) + 2 \cdot \text{cost}(Del)$ since we add one object to a concept and remove two.

Accordingly, we obtain $\llbracket \alpha_{cost} \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) = \text{cost}(Add) + 2 \cdot \text{cost}(Del)$. This can be seen as follows. First, observe that the last three rows of the definition of α_{cost} are equal to e_{\otimes} since $\neg\text{displacement}(i)$, $\neg\text{classVar}(i, c, c')$, $\neg\text{propertyVar}(i, t)$ hold for all i, c, c', t and thus

$$\begin{aligned}
& \llbracket \text{displacement}(i) * \text{cost}(Disp) + \neg\text{displacement}(i) \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) \\
& = \llbracket \text{displacement}(i) * \text{cost}(Disp) \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) \oplus \llbracket \neg\text{displacement}(i) \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) \\
& = \llbracket \text{displacement}(i) \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) \otimes \llbracket \text{cost}(Disp) \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) \oplus e_{\otimes} \\
& = e_{\oplus} \otimes \llbracket \text{cost}(Disp) \rrbracket_{\mathcal{R}_{\min,+}}(\mathcal{I}) \oplus e_{\otimes} \\
& = e_{\oplus} \oplus e_{\otimes} = e_{\otimes}
\end{aligned}$$

The same can be observed for the last two rows.

On the other hand, by the same reasoning, since the interpretation \mathcal{I} contains $\text{addition}(\text{RollingContainer}, i_1)$ the first row evaluates to $\text{cost}(Add)$ and the second row evaluates to $\text{cost}(Del) \otimes \text{cost}(Del)$, resulting in

$$\text{cost}(Add) \otimes \text{cost}(Del) \otimes \text{cost}(Del).$$

Since we use $\mathcal{R}_{\min,+}$ the operation \otimes is $+$ and we obtain $\text{cost}(Add) + 2 \cdot \text{cost}(Del)$ as the final cost, as expected.

4.4 Translation to Weak Constraints

While algebraic measures are a useful tool, to specify quantitative measures for the answer sets of programs, most solvers for ASP currently do not support optimization of the weight of an algebraic measures. However, the algebraic measure that we use can be translated to *weak constraints* [5].

Recall that intuitively the weighted formula α_{cost} corresponds to the sum of the cost of the modifications that were performed on the scene. E.g. for

$$(\text{addition}(c, i) * \text{cost}(\text{Add}) + \neg\text{addition}(c, i))$$

we either have cost 0 if $\neg\text{addition}(c, i)$ holds or cost $\text{cost}(\text{Add})$ if $\text{addition}(c, i)$ holds.

This is exactly the kind of costs that can be modelled and optimized using weak constraints of the form

$$:\sim a_1, \dots, a_n, \text{not } b_1, \dots, \text{not } b_m.[C, t_1, \dots, t_k],$$

where a_i and b_j are atom formulas, C is the cost and t_l are terms. This weak constraints means that satisfying a_1, \dots, a_n but not b_1, \dots, b_m incurs a cost of C . The terms t_1, \dots, t_k intuitively group different weak constraints. That is, if there are multiple weak constraints with the same terms, then only the one with the highest cost is triggered.

This means, we can use the weak constraint

$$:\sim\text{addition}(C, I).[cost(\text{Add}), \text{add}, C, I]$$

to add a cost of $\text{cost}(\text{Add})$ for each individual i and concept c such that $\text{addition}(c, i)$ holds, i.e., such that we add i to concept c .

We can do the same for the other factors of α_{cost} to translate it to ASP with weak constraints.

5 Implementation Prototype for Scene Generation in Autonomous Driving

We implemented and tested our approach using an example from Autonomous Driving. Here, we reconstructed and slightly extended the base ontology from [7,15].⁵ It features different scenes each annotated with the objects included in it. The included objects are annotated with information about them, such as their type. Additionally, the ontology includes axioms that add additional knowledge about the relationship between different concepts in the ontology. Overall, the base ontology has more than 3 million axioms, concerning knowledge of 41 concepts, more than 100 scenes, and more than 50 thousand objects.

In our prototype we restrict ourselves to limited scene variations, i.e., addition and deletion of objects from concepts, and assign them cost 1 each. However, also the other modifications detailed above could be added without problems.

⁵ https://github.com/boschresearch/ad_cskg

5.1 Overview

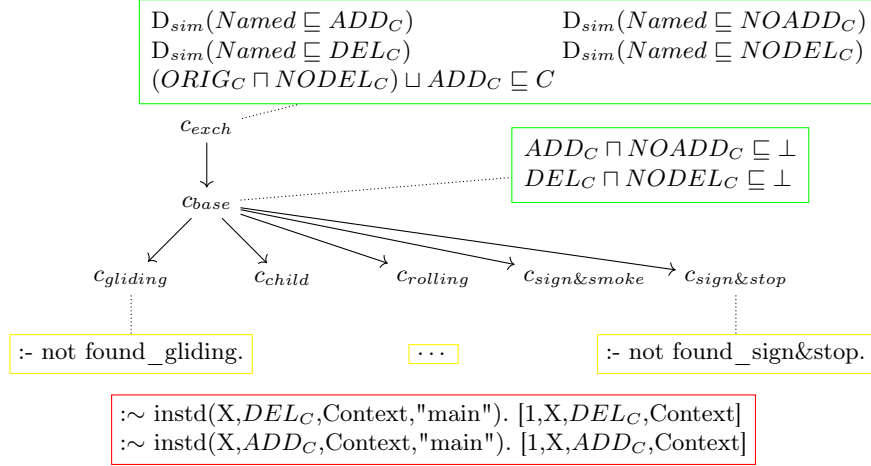


Fig. 4. MR-CKR and additional constraints used in our prototype for autonomous driving. c_i denotes the different contexts, arrows between contexts denote higher specificity. Ontology axioms and additional ASP constraints belonging to some context are shown in green and yellow boxes, respectively, and are connected to their context with a dotted line. The red box contains the additional weak constraints that optimize for similarity.

We provide an overall sketch of the MR-CKR and its interplay with ASP constraints in Figure 4. We go over the different parts step by step.

As diagnoses for network failure we use the following (using existing classes from the ontology):

1. The scene contains an object that is in the class *GlidingOnWheels*. Dangerous due to lack of no examples.
2. The scene contains an object that is in the class *Child*. Dangerous due to unpredictable behaviour compared to other humans.
3. The scene contains an object that is in the class *RollingContainer* but no object in the class *Human*. Dangerous due to unpredictable behaviour of the rolling container.
4. The scene contains an object that is in the class *Sign* and an object in the class *Smoke*. Dangerous due to harder recognition of the sign due to smoke.
5. The scene contains an object that is in the class *Sign* and an object in the class *StopLineMarking*. Dangerous because the network does not predict stopping at the stop line properly.

This means that we have one context c_i for each diagnosis i .

Recall the overall framework from Figure 1. In order to model the possible scene modifications, we have an exchange context c_{exch} , that contains for each original (modifiable) concept C in the ontology the following default axioms:

$$\begin{array}{ll} D_{sim}(Named \sqsubseteq ADD_C) & D_{sim}(Named \sqsubseteq NOADD_C) \\ D_{sim}(Named \sqsubseteq DEL_C) & D_{sim}(Named \sqsubseteq NODEL_C) \\ (ORIG_C \sqcap NODEL_C) \sqcup ADD_C \sqsubseteq C & \end{array}$$

Here,

- $Named$ is a concept we define based on the original ontology that contains all modifiable objects,
- ADD_C is a concept that represents the individuals that should be added to the concept C ,
- $NOADD_C$ is a concept that represents the individuals that should *not* be added to the concept C ,
- DEL_C is a concept that represents the individuals that should be removed from the concept C ,
- $NODEL_C$ is a concept that represents the individuals that should *not* be removed from the concept C ,
- $ORIG_C$ is a concept that represents the individuals that were in the concept C in the original ontology.

Thus, we assert for each modifiable individual that they should be (not) added/removed to/from the concept C . If they were originally in C and are not removed or are added to C , then they should be in C , as the last axiom asserts. Clearly on its own, this does not make sense, since every modifiable individual will be in every concept. Therefore, we add a base context c_{base} that contains for each (modifiable) concept C in the ontology the following axioms:

$$\begin{array}{l} ADD_C \sqcap NOADD_C \sqsubseteq \perp \\ DEL_C \sqcap NODEL_C \sqsubseteq \perp \end{array}$$

These ensure that we either add (resp. remove) or do not add (resp. remove) an individual but not both. Then if c_{exch} is less specific than c_{base} with respect to \succ_{sim} , we can override the defaults of c_{exch} in c_{base} , such that we can satisfy the disjointness requirements in c_{base} .

In order to give the contexts c_i for the diagnoses access to the possibility of modification, we then declare each context c_i more specific than c_{base} with respect to \succ_{sim} .

What is left, are the additional ASP constraints, that (i) ensure minimal modifications and (ii) ensure the presence of the diagnosis in the given contexts.

For (i), we use the following rules for each modifiable concept C :

```
:~ instd(X, DEL_C, Context, "main"). [1, X, DEL_C, Context]
:~ instd(X, ADD_C, Context, "main"). [1, X, ADD_C, Context]
```


This ensures that a penalty of 1 is added every time we add or delete an individual X to C . Note that the penalty is applied for every context.

For (ii), we simply add constraints that ensure that the diagnosis is derived. For example, for the third diagnosis in context C_3 , where we need to derive that there is an object in the concept *RollingContainer* but none in the concept *Human*, we add the rules

```
found_rolling_no_human_1 :- instd(X, RollingContainer, C3, "main").
:- not found_rolling_no_human_1.
found_rolling_no_human_2 :- instd(X, Human, C3, "main").
:- found_rolling_no_human_2.
```

Example 5. Assume our input scene contains four objects i_1, \dots, i_4 and

$$Child(i_2), Child(i_3), Car(i_4)$$

hold.

Due to the ontology axiom $Child \sqsubseteq Human$, we could derive $Human(i_2)$ and $Human(i_3)$.

Thus, in context C_3 , where we need a rolling container but no human, we need to remove i_2 and i_3 from the *Child* concept and add an object to the *RollingContainer* concept. Thus, a potential modification (restricted to context C_3) is represented by the interpretation

$$\mathcal{I} = \{\text{instd}(i_1, ADD_{RollingContainer}, C_3, \text{"main"}), \\ \text{instd}(i_2, DEL_{Child}, C_3, \text{"main"}), \\ \text{instd}(i_3, DEL_{Child}, C_3, \text{"main"})\}.$$

As discussed in the previous example, it has cost $\text{cost}(Add) + 2 \cdot \text{cost}(Del)$, which is 3 since we assign addition and deletion cost 1.

Since there is no modification of a lower cost, one of the possible generated scenes for C_3 consists of

$$RollingContainer(i_1), Car(i_4),$$

i.e., it contains a rolling container i_1 , no humans, but a car i_4 .

5.2 Practical Implementation

In order to transfer the idea that we sketched above into a formal encoding of the problem that we can solve with an ASP solver such as clingo [11], we proceed in the following steps:

1. Build an MR-CKR encoding
2. Translate the MR-CKR encoding to ASP
3. Add strong constraints to ensure danger
4. Add weak constraints to ensure similarity

We implemented all these steps and made them available online.⁶ In more detail, we tackle them as follows.

⁶ <https://github.com/raki123/MR-CKR>

Building an MR-CKR encoding. Here, we read the base ontology and gather (i) axioms that generally hold and (ii) knowledge regarding some particular scene. Then, we build the MR-CKR as sketched in Figure 4. There are two details to note here. First, we additionally add the general axioms from the base ontology that we previously gathered to c_{base} . This ensures that the generated scenes are realistic. Second, we do not add defaults to add/delete individuals to concepts for every concept but only a subset of relevant ones. This helps us by reducing the size of the problem encoding and by improving inference performance.

Translating the MR-CKR encoding to ASP. The CKRew software⁷ is an existing tool from previous work [2] that performs the desired translation of MR-CKRs to ASP. The original translation is capable of handling highly complex relations between contexts and supports arbitrary defaults and flexible ontological background knowledge. However, this comes at the cost of an encoding in ASP that is not suitable for our purposes, using large scene graphs with many contexts and concepts.

To circumvent this, we specialized the encoding to our setting. Namely, for a given concept C , the (defeasible) axioms

$$\begin{aligned} D_{sim}(Named \sqsubseteq ADD_C) & \quad D_{sim}(Named \sqsubseteq NOADD_C) \\ ADD_C \sqcap NOADD_C & \sqsubseteq \perp \end{aligned}$$

tell us that we guess *either* the addition *or* the non-addition of any named individual X to C , as long as there is no other reason in the ontology that prevents both. Since our base ontology is consistent, there can never be a reason in our ontology that prevents the non-addition. Therefore, the either-or really holds in our setting.

This allows us to use the following rules to encode the (defeasible) axioms above:

```

instd(X,ADD_C,Con,"main") :- instd(X,"Named",Con,"main"),
                               not instd(X,NOADD_C,Con,"main").
instd(X,NOADD_C,Con,"main") :- instd(X,"Named",Con,"main"),
                               not instd(X,ADD_C,Con,"main").

```

We do the same for deletion and non-deletion.

This specialized translation for our setting leads to a significant performance improvement. While the original encoding only allows us to generate new scenes using tiny starting scenes, the improved strategy allows inference of the real world scenes from the ontology within seconds.

Adding ASP constraints The addition of the strong and weak constraints is surprisingly simple. We can refer to the derived knowledge by making use of the vocabulary that CKRew uses for translation. Thus, we can provide all additional constraints in a separate program file and solve the combination of the ASP encoding of the MR-CKR and the additional constraints.

⁷ <https://github.com/dkmfbk/ckrew>

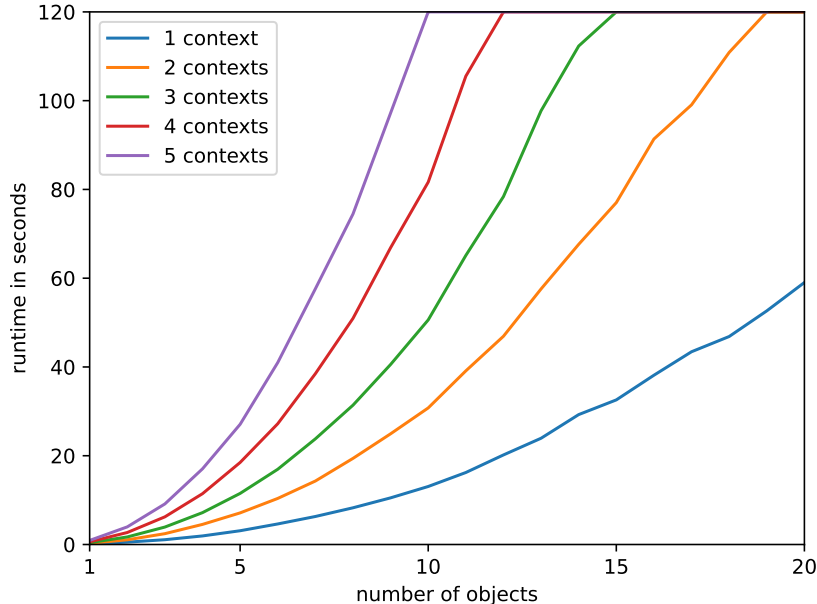


Fig. 5. Solving time after using the GENERAL translation of MR-CKR to ASP.

5.3 Scalability

We briefly investigate how large the instances that we can solve can become, while maintaining a low runtime. Here, we consider on the one hand the original translation of MR-CKR to ASP, denoted GENERAL, and on the other hand the specialized translation that makes use of the restricted use of defaults, denoted SPECIALIZED. The aim here is to show that is not only helpful but even necessary to use SPECIALIZED over GENERAL, when solving real world problems.

Secondly, we investigate the dependence of the runtime on (i) the number of objects in the scene and (ii) the number of contexts. We vary (i) between 1 and 20 and (ii) between 1 and 5 on a randomly chosen example scene from the ontology.

We only measure the solving time, since building the MR-CKR and translating the MR-CKR to an ASP encoding only consumed insignificant time (less than 1 second) regardless of the translation and number of objects/contexts.

For the solving phase, we use “clingo” [11] on the non-ground program with input option “-t 3” to specify that three threads in parallel should be used to solve the problem. We apply a time limit of 120 seconds and assign runs that do not finish during this time a runtime of 120 seconds.

The results of our investigation are given in Figures 5 and 6. We see that even if only one context is used, the runtime after GENERAL grows quickly. While it

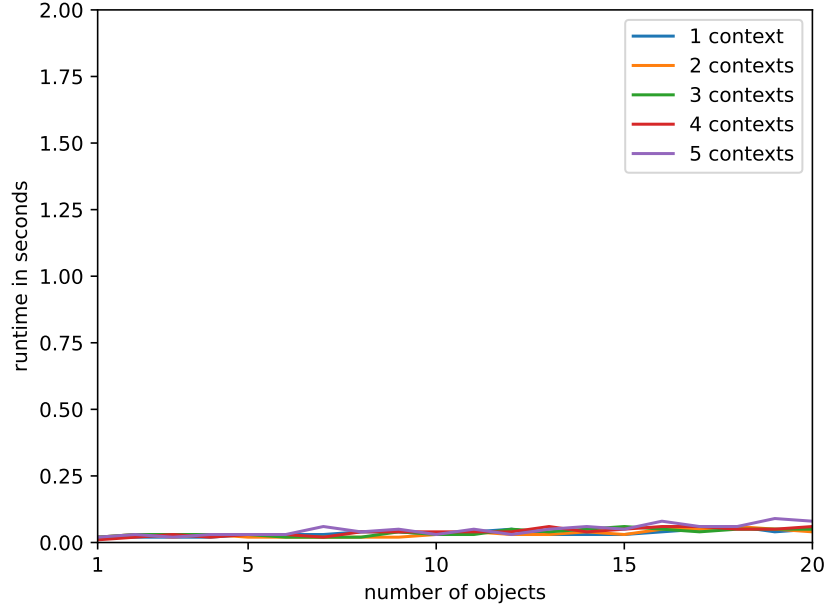


Fig. 6. Solving time after using the SPECIALIZED translation of MR-CKR to ASP.

still remains in a feasible range, when using one context and up to 20 objects in the scene, the same cannot be said when more contexts are used. For five contexts, solving already becomes slow when ten or more objects are included in the scene. Additionally, the original scene has many more objects (more than 300), thus, this translation can only be employed to restricted examples, even if there is only one context.

On the other hand, for SPECIALIZED we see that the solving time is consistently far below one second, even when using all five contexts and 20 objects in the scene. Note here the different limits of the Y-axis, which we adapted to make the runtimes visible. Even when using all objects that were originally included in the scene (more than 300) the solving time remains at around 0.67 seconds.

We see that while the original translation GENERAL is able to handle a broader range of MR-CKRs, it pays off to use the specialized translation SPECIALIZED in our setting. With SPECIALIZED we can generate new scenes in sub-second times, even if the full scene (i.e. all its objects) and all contexts are used. This suggests that with SPECIALIZED we can also generate new inputs for more complex semantic conditions and base ontologies than the ones provided in our prototype, giving us interesting opportunities to extend our work in the future.

6 Conclusion

We introduced a new framework to generate new interesting inputs for neural models based on existing ones, in particular the setting of scene generation for AV scene data. Notably, our framework does so based on symbolic reasoning methods: this allows us, on the one hand, to incorporate real world knowledge (in the form of contextual knowledge) that ensures that the generated inputs are *realistic*, and, on the other hand, to formulate a semantic criterion that should be satisfied by the new input.

We saw that all components that we incorporated in our framework add their respective benefits:

- **MR-CKR** allows us (i) to incorporate ontological knowledge easily and (ii) to perform different modifications in different contexts.
- **Algebraic Measures** allow us to easily specify a cost value to optimize.
- **ASP**, as a declarative programming language to translate to, allows us to perform reasoning/scene generation efficiently using standard solvers.

While we only considered a small example in our prototype, it successfully generates new scene descriptions. Furthermore, as it can be easily generalized to the generation of different types of scenes, it provides a proof of concept of our approach.

In future work, it will be interesting to extend this example with more complicated semantic descriptions of interesting scenes gathered by inspecting poor performing inputs for a prediction task with a neural model: in particular, it would be interesting to use more complex contextual structures to represent different variations of the scenes, but also use inputs performances to give a quantification of the more interesting cases to be generated. Another open challenge is to use the symbolic description of the new scene to generate images that can be fed to the neural model and assess how much training with these new examples improves the network performance.

Acknowledgements

This work was partially supported by the European Commission funded projects “Humane AI: Toward AI Systems That Augment and Empower Humans by Understanding Us, our Society and the World Around Us” (grant #820437) and “AI4EU: A European AI on Demand Platform and Ecosystem” (grant #825619), and the Austrian Science Fund (FWF) project W1255-N23. The support is gratefully acknowledged.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
2. Bozzato, L., Eiter, T., Kiesel, R.: Reasoning on multirelational contextual hierarchies via answer set programming with algebraic measures. *Theory Pract. Log. Program.* **21**(5), 593–609 (2021). <https://doi.org/10.1017/S1471068421000284>, <https://doi.org/10.1017/S1471068421000284>
3. Bozzato, L., Eiter, T., Serafini, L.: Enhancing context knowledge repositories with justifiable exceptions. *Artif. Intell.* **257**, 72–126 (2018)
4. Bozzato, L., Serafini, L., Eiter, T.: Reasoning with justifiable exceptions in contextual hierarchies. In: KR 2018. pp. 329–338. AAAI Press (2018)
5. Buccafurri, F., Leone, N., Rullo, P.: Strong and weak constraints in disjunctive datalog. In: Dix, J., Furbach, U., Nerode, A. (eds.) *Logic Programming and Nonmonotonic Reasoning*, 4th International Conference, LPNMR'97, Dagstuhl Castle, Germany, July 28-31, 1997, Proceedings. *Lecture Notes in Computer Science*, vol. 1265, pp. 2–17. Springer (1997). https://doi.org/10.1007/3-540-63255-7_2, https://doi.org/10.1007/3-540-63255-7_2
6. Chen, B., Feng, Y., Dai, T., Bai, J., Jiang, Y., Xia, S., Wang, X.: Adversarial examples generation for deep product quantization networks on image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(2), 1388–1404 (2023). <https://doi.org/10.1109/TPAMI.2022.3165024>, <https://doi.org/10.1109/TPAMI.2022.3165024>
7. Chowdhury, S.N., Wickramarachchi, R., Gad-Elrab, M.H., Stepanova, D., Henson, C.A.: Towards leveraging commonsense knowledge for autonomous driving. In: ISWC (Posters/Demos/Industry) (2021)
8. Droste, M., Gastin, P.: Weighted automata and weighted logics. *Theor. Comput. Sci.* **380**(1-2), 69–86 (2007). <https://doi.org/10.1016/j.tcs.2007.02.055>, <https://doi.org/10.1016/j.tcs.2007.02.055>
9. Eiter, T., Kiesel, R.: Weighted LARS for quantitative stream reasoning. In: Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) *ECAI 2020 - 24th European Conference on Artificial Intelligence*, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). *Frontiers in Artificial Intelligence and Applications*, vol. 325, pp. 729–736. IOS Press (2020). <https://doi.org/10.3233/FAIA200160>, <https://doi.org/10.3233/FAIA200160>
10. Gao, J., Lanchantin, J., Soffa, M.L., Qi, Y.: Black-box generation of adversarial text sequences to evade deep learning classifiers. In: 2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018. pp. 50–56. IEEE Computer Society (2018). <https://doi.org/10.1109/SPW.2018.00016>, <https://doi.org/10.1109/SPW.2018.00016>
11. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo = ASP + control: Preliminary report. *CoRR* **abs/1405.3694** (2014), <http://arxiv.org/abs/1405.3694>
12. Kimmig, A., den Broeck, G.V., Raedt, L.D.: Algebraic model counting. *J. Appl. Log.* **22**, 46–62 (2017). <https://doi.org/10.1016/j.jal.2016.11.031>, <https://doi.org/10.1016/j.jal.2016.11.031>
13. Rozsa, A., Rudd, E.M., Boulton, T.E.: Adversarial diversity and hard positive generation. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016.

- pp. 410–417. IEEE Computer Society (2016). <https://doi.org/10.1109/CVPRW.2016.58>, <https://doi.org/10.1109/CVPRW.2016.58>
14. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. *J. of Web Semantics* **12**, 64–87 (2012)
 15. Wickramarachchi, R., Henson, C., Sheth, A.: Knowledge-infused learning for entity prediction in driving scenes. *Frontiers in big Data* p. 98 (2021)
 16. Yang, B., Zhang, H., Li, Z., Zhang, Y., Xu, K., Wang, J.: Adversarial example generation with adabelief optimizer and crop invariance. *Appl. Intell.* **53**(2), 2332–2347 (2023). <https://doi.org/10.1007/s10489-022-03469-5>, <https://doi.org/10.1007/s10489-022-03469-5>