# Master Thesis (Extended Abstract)
# Inconsistency Management for Traffic Regulations

**Harald Beck**

Institut für Informationssysteme 184/3, Technische Universität Wien

Favoritenstraße 9-11, A-1040 Vienna, Austria

beck@kr.tuwien.ac.at

Advisor: O. Prof. Dr. Thomas Eiter, Technische Universität Wien

**Abstract.** In modern day road traffic management, government officials are supported by software to store and visualize traffic signs and their underlying legal intentions (so-called traffic measures) on top of digital street maps. To date, however, methods are missing that help to ensure the correctness of traffic regulations. Driven by the vision of Smart Cities, this lack of tools is becoming increasingly problematic due to the rapid shift towards dynamic regulations.

To fill this gap, we develop the first formal model of traffic regulations and formalize practically important use cases as reasoning tasks, including consistency evaluation, diagnosis, and different kinds of repairs. We analyze the computational complexity for different logical representation formalisms and study the relation between diagnoses and repairs. Moreover, we characterize when an inconsistent scenario can be decomposed into unrelated contexts. Finally, we also provide an easily readable and highly modular prototypical implementation using Answer Set Programming, which is both a rule-based logic and a programming paradigm.

## 1 Introduction

Government officials, e.g., in Lower Austria, are aided by software tools to keep track of enacted traffic regulation orders (so-called *traffic measures*) and posted traffic signs. However, no mechanisms exist to support authorities in ensuring the logical consistency of such regulations, and their compliance with the law. Finding and correcting errors is not trivial in real life situations and many subtle inconsistencies may occur, as already the following simple example shows.

**Example 1** Fig. 1 schematically depicts a junction with three arms. The blue line from $a$ to $b$ illustrates an intended 30 km/h speed limit restriction along on the horizontal street. To make this measure legally binding we have to announce it by the start sign at $a$ and terminate its validity by the end sign at $b$. Now, consider a road user who turns right from arm C to arm B. There is no start sign in this direction of travel. Thus, the sign posting for the 30 km/h restriction is insufficient. ■

Such inconsistencies create problems in daily traffic. Officials are confronted with legal issues (e.g., challenging of speeding tickets) when two dissenting speed limits are announced. Even more delicate is the aspect of legal responsibility in case of accidents caused by wrong sign posting. Different from that, errors in the data acquisition in traffic sign

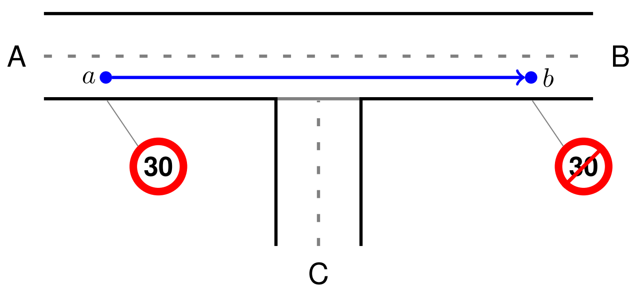

Figure 1: Inconsistent sign posting of a 30 km/h speed limit measure from $a$ to $b$

software may lead to wrong assumptions on the state of traffic regulations. Driven by the vision of Smart Cities, there is a rapid shift towards *active traffic management*, where regulations dynamically adjust to the current traffic based on sensor data. Such regulations must be consistent with each other, and their utilization further complicates the maintenance and planning tasks of traffic authorities.

Therefore, we need tools that help to ensure the compliance of traffic regulations with flexibly editable specifications.

### 1.1 State of the Art

In traffic regulation management software like SKAT,[1] virtual traffic measures and traffic signs can be posted on street maps (similar to Google Maps) to support traffic authorities in their maintenance tasks.

However, to the best of our knowledge, systematic inconsistency management for traffic regulations has been an entirely unexplored domain prior to this work.

- No formal model of traffic regulations is available.
- In particular, there is no scientific treatment of the complex logic implied by traffic signs or their semantic relation to traffic measures.
- No software tools exist to detect, explain or repair inconsistent configurations of measures and signs.

We turn these observations into goals of the thesis.

[1] http://www.prisma-solutions.at/index.php/en/references/skat-province-of-lower-austria

## 1.2 Goals

The general aim of this thesis is

- to lay the theoretical *foundation* of future software tools for traffic management such that
- the stored traffic signs and measures can be guaranteed to *comply with flexibly editable specifications* like the Road Traffic Regulation; and moreover to
- provide a modular, prototypical *implementation* as a proof of concept.

In particular, we want to provide computer assistance for the following major use cases:

1. *Evaluation.* Determine whether the stored traffic signs/measures are compliant with the specification. In case of inconsistency, we want to know which *conflicts* occur, and where.
2. *Correspondence.* Ensure that measures and signs express the same restrictions, i.e., that all measures are sufficiently announced and all traffic signs are legally justified.
3. *Diagnosis.* In case of inconsistencies, determine the causes of a given set of conflicts.
4. *Repair.* Automatically suggest modifications of regulations such that the result is free of conflicts.

To account for the intrinsic complexity of the domain, we require a highly *readable* and *modular* implementation.

## 1.3 Contributions

**Domain Analysis**. We discern in a comprehensive domain analysis relevant aspects in traffic measure and traffic sign information, categorize different classes of errors and identify major use cases to deal with them.

**Formal Model for Traffic Regulations**. We develop a high-level, logic-based model for traffic signs and measures. As a result, we can (i) describe the semantic relation between measures and signs and (ii) test their compliance with a formal specification without being bound to legal interpretations, particular data representations or the expressiveness required to describe certain properties. We also present an exemplary instantiation of this model using Answer Set Programming (ASP) [1; 6] as underlying logic. (Section 2)

**Reasoning Tasks**. We formalize the envisaged use cases in form of reasoning tasks and further investigate various practically relevant special cases of so-called strict repairs, where measures and signs can be repaired at the same time. Moreover, we characterize *contexts* due to which conflicts arise and examine relations between diagnoses and repairs. (Section 3)

**Computational Complexity**. We determine the computational complexity of associated decision problems for different representation formalisms, viz. first order logic and three classes of ASP [4], namely stratified programs, normal programs, and disjunctive programs. (Section 4)

**Implementation**. We develop a systematic, uniform approach for an ASP-based implementation and show formally that they compute the solutions of the defined reasoning tasks.

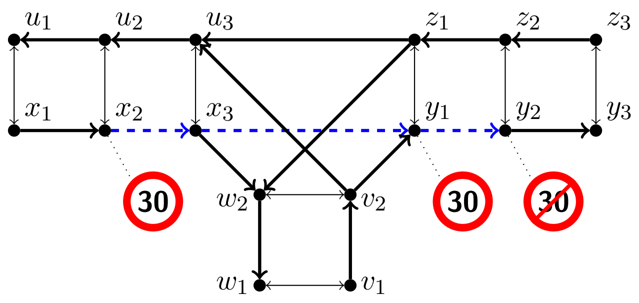

Figure 2: Correct traffic sign posting for a 30 km/h measure

We give prototypical examples and demonstrate how the fully declarative semantics of ASP assists readability and modular composition. (Section 5)

Due to space reasons, we skip the domain analysis, some formal definitions, and present only a selection of the results.

## 2 Formal Model

The first ingredient of a formal model for traffic regulations is a *street graph*. We employ directed, labelled graphs $G$ which we encode as sets of atoms of form $e(t, v, w)$, where $t \in \{\mathit{left}, \mathit{right}, \mathit{straight}, \mathit{lane}, \mathit{uturn}\}$ is the label of the edge from node $v$ to node $w$.

A key insight is that measures and signs can be seen as two input languages that describe restrictions on road usage. Thus, if we represent them uniformly by means of their *effects*, we can compare them semantically. We can then also study their compliance w.r.t. a specification like the Road Traffic Regulation in a uniform way. The aim is to derive *conflicts* in case of inconsistent configurations.

We start by giving the representation of these entities.

**Definition 1 (Measures, Signs, Effects, Conflicts)** *Let $G$ be a street graph and* $\mathsf{M}, \mathsf{S}, \mathsf{F}, \mathsf{C}$ *be sets of ground terms for measure/sign/effect/conflict types. We define sets of atoms:*

- Measures $M_G = \{m(t, v, w) \mid t \in \mathsf{M}, (v, w) \in E\}$
- Signs $S_G = \{s(t, v) \mid t \in \mathsf{S}, v \in V\}$
- Input $I_G = M_G \cup S_G$
- Effects $F_G = \{f(t, v, w) \mid t \in \mathsf{F}, (v, w) \in E\}$
- Conflicts $C_G = \{c(t, v) \mid t \in \mathsf{C}, v \in V\}$

If $M \subseteq M_G$ and $S \subseteq S_G$, $Sc = (G, M, S)$ is called a *scenario*.

**Example 2** Fig. 2, depicts the representation of the junction of Fig. 1 as graph. For instance, the right turn from arm $\mathsf{C}$ to arm $\mathsf{B}$ is encoded by the atom $e(\mathit{right}, v_2, y_1)$. The dashed blue path from $x_2$ to $y_2$ symbolizes a 30 km/h speed limit measure which we formalize as a set of the following atoms:

$$\begin{aligned} M = \{ & m(\mathit{spl}(30), x_2, x_3), m(\mathit{spl}(30), x_3, y_1), \\ & m(\mathit{spl}(30), y_1, y_2) \} \subseteq M_G \end{aligned}$$

The traffic signs are defined similarly:

$$\begin{aligned} S = \{ & s(\mathit{start}(\mathit{spl}(30)), x_2), s(\mathit{start}(\mathit{spl}(30)), y_1), \\ & s(\mathit{end}(\mathit{spl}(30)), y_2) \} \subseteq S_G \end{aligned}$$ ■

## 2.1 Traffic Regulations

We leave open in which form of predicate logic traffic regulations and supplementary evaluation criteria are formalized. Given any such specification, we will first map signs/measures to effects, and then effects to conflicts.

**Definition 2 ($Cn_G$)** *Let $X$ and $Y$ be sets of atoms on $G$ and let $T$ be a set of formulas in a fixed predicate logic $\mathcal{L}$. The* $Y$-consequences of $T$ and $X$ (on $G$) *is the set of atoms*

$$Cn_G(T, X, Y) = \{y \in Y \mid T \cup \overline{G} \cup \overline{X} \models y\}.$$

Here, $\models$ is the consequence relation in the underlying logic $\mathcal{L}$, and $\overline{G}$ (resp. $\overline{X}$) closes the set $G$ (resp. $X$) w.r.t. the set of all edges (resp. an implicit base set w.r.t. $G$) to ensure unique valuations. Informally, $Cn_G(T, X, Y)$ is the subset of $Y$ entailed by a specification $T$, given input facts $X$ on $G$.

Let $I \subseteq I_G$ be any input on graph $G$. Then, by an

- *effect mapping* $P$, we understand a set of formulas that yields a set $\mathcal{F}^P_G(I) = Cn_G(P, I, F_G)$ of *effects of* $I$, and a
- *conflict specification* $Sp$ over $P$ is a set of formulas that gives the *conflicts of* $I$, defined by $\mathcal{C}^{P,Sp}_G(I) = Cn_G(Sp, \mathcal{F}^P_G(I), C_G)$.

**Example 3** We give an illustration of how an effect mapping may look like in ASP. Given a start sign of a speed limit of value $K$ at node $X$, i.e., atom $s(\mathit{start}(\mathit{spl}(K)), X)$, the next edge in direction of travel ($\mathit{dir}$) is labelled with a maximum speed ($\mathit{ms}$) restriction of $K$ km/h. This is expressed by rule 1. The effect is recursively propagated by rule 2, unless this propagation is blocked ($\mathit{blk}$). For speed limits, this is the case, if there is an end sign (rule 3) or a start sign of a different speed limit $J$ (rule 4).

$$\begin{aligned}
f(\mathit{ms}(K), X, Y) &\leftarrow s(\mathit{start}(\mathit{spl}(K)), X), \mathit{dir}(X, Y) && (1)\\
f(F, Y, Z) &\leftarrow f(F, X, Y), \mathit{dir}(Y, Z), \\
&\qquad \text{not } \mathit{blk}(F, Y, Z) && (2)\\
\mathit{blk}(\mathit{ms}(K), X, Y) &\leftarrow s(\mathit{end}(\mathit{spl}(K)), X), \mathit{dir}(X, Y) && (3)\\
\mathit{blk}(\mathit{ms}(K), X, Y) &\leftarrow s(\mathit{start}(\mathit{spl}(J)), X), \mathit{dir}(X, Y)\\
&\qquad \mathit{speed}(K), K \neq J && (4)
\end{aligned}$$

Note that rule 2 is generic, i.e., not tailored for speed limits. ∎

Similarly, conflict specifications utilize effect atoms to derive conflict atoms. For instance, the rule

$$\begin{aligned}
c(\mathit{overlap}(F_1, F_2), X) &\leftarrow f(F_1, X, Y), f(F_2, X, Y),\\
&\qquad \mathit{contr}(F_1, F_2) && (5)
\end{aligned}$$

formalizes a conflict at node $X$, when two contradicting effects overlap on its outgoing edge to node $Y$, e.g., two dissenting speed limits:

$$\mathit{contr}(\mathit{ms}(K), \mathit{ms}(J)) \leftarrow \mathit{speed}(K), \mathit{speed}(J), K < J \quad (6)$$

When $P$ is an effect mapping, $Sp$ is a conflict specification and $Sc$ is a scenario, then $\Pi = (P, Sp)$ is called a *traffic regulation* and $\mathcal{T} = (\Pi, Sc)$ is a *traffic regulation problem* (TRP).

## 3 Reasoning Tasks

The domain analysis allowed us to capture the ingredients of traffic regulation problems $\mathcal{T}$ in precise terms. On top of this, we now formalize the identified logic-oriented use cases in form of reasoning tasks.

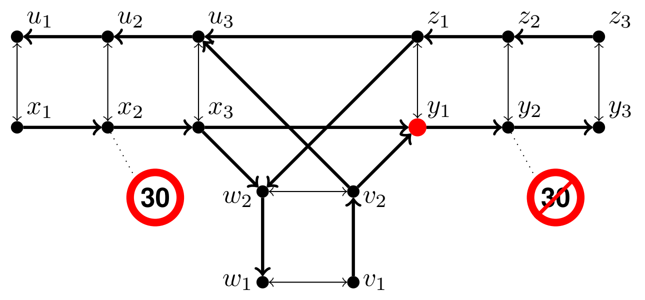

Figure 3: Conflict at $y_1$ due to a missing 30 km/h start sign

### 3.1 Formalization of Use Cases

**Definition 3 (Inconsistency)** *The* conflicts $\mathcal{C}(\mathcal{T})$ *of a TRP $\mathcal{T}$ with input $I$ are defined by $\mathcal{C}^{P,Sp}_G(I)$. If $\mathcal{C}(\mathcal{T}) \neq \emptyset$, we call $\mathcal{T}$, its scenario $Sc$, and $I$* inconsistent.

In this case, we want to determine the causes of conflicts.

**Definition 4 (Diagnosis)** *A* diagnosis *for a (non-empty) set of conflicts $C \subseteq \mathcal{C}(\mathcal{T})$ is a set $J \subseteq I$, such that $C \subseteq \mathcal{C}^{P,Sp}_G(J)$.*

The set of all diagnoses for $C$ with fixed $\mathcal{T}$ is denoted by $\mathcal{D}_{\mathcal{T}}(C)$, and $\mathcal{D}^{\subseteq}_{\mathcal{T}}(C)$ denotes the $\subseteq$-minimal diagnoses.

**Example 4** Due to space reasons, we will only informally define subsequent traffic regulations $\Pi = (P, Sp)$. Intuitively, $\Pi$ should specify that for $\mathcal{T}$, the conflicts $\mathcal{C}(\mathcal{T})$ should contain an atom $c(\mathit{bad\text{-}end}(F), v)$, whenever an effect of type $F$ ends at a node $v$ without an explicit end sign.

Thus, if we remove the repeated start sign at $y_1$ in Fig. 2, we obtain the variant shown in Fig. 3 which depicts the conflict $c(\mathit{bad\text{-}end}(\mathit{ms}(30)), y_1)$. This conflict has a unique minimal diagnosis, namely $\{s(\mathit{start}(\mathit{spl}(30)), x_2)\}$. ∎

Towards repairs, we consider *update* pairs $(I^-, I^+)$ of signs/measures that are deleted and added, respectively. Replacing $I$ with $I' = (I \setminus I^-) \cup I^+$ in $\mathcal{T}$ yields the *updated TRP*, denoted by $\mathcal{T}[I^-, I^+]$. In general, we might delete and add both measures and signs.

**Definition 5 (Repair)** *An update $(I^-, I^+)$ for $\mathcal{T}$ is a* (local) repair for conflicts $C \subseteq \mathcal{C}(\mathcal{T})$, *if $C \cap \mathcal{C}(\mathcal{T}[I^-, I^+]) = \emptyset$. It is called a* repair for $\mathcal{T}$, *if $\mathcal{C}(\mathcal{T}[I^-, I^+]) = \emptyset$.*

We observe that a repair for $\mathcal{C}(\mathcal{T})$ is not necessarily a repair for $\mathcal{T}$, since updates $(I^-, I^+)$ may introduce new conflicts. Second, we note that an update $(I, \emptyset)$, which deletes the entire input, is always a trivial, but rarely desired repair.

**Example 5** The TRP of Fig. 3 has two minimal repairs: adding the missing start sign, i.e., the update

$$(\emptyset, \{s(\mathit{start}(\mathit{spl}(30)), y_1)\}),$$

or removing the present start sign by

$$(\{s(\mathit{start}(\mathit{spl}(30)), x_2)\}, \emptyset).$$

(We did not specify that an end sign without a start sign is inconsistent.) ∎

**Definition 6 (Correspondence)** *A set of measures $M$ and a set of signs $S$* correspond *w.r.t. effect mapping $P$ on graph $G$, if they express the same effects, i.e., $\mathcal{F}^P_G(M) = \mathcal{F}^P_G(S)$.*

We have *unannounced* measures, if some of their effects are not expressed by any sign, i.e., when the set $\mathcal{F}_G^P(M) \setminus \mathcal{F}_G^P(S)$ is non-empty. Similarly, elements in $\mathcal{F}_G^P(S) \setminus \mathcal{F}_G^P(M)$ point to legally *unjustified* restrictions.

Requiring correspondence in addition to consistency gives us further practically important use cases.

- **Strict Repairs** are repairs such that measures and signs correspond in the updated TRP.
- **Adjustments**, used to correct imported data, are strict repairs that modify only atoms of one language, measures or signs, where the input of the other is consistent.
- **Generation** assumes consistent input of only one language, and creates data from the other one from scratch. For instance, the current version of SKAT can list correct sign posting options for newly created traffic measures. This ad-hoc feature cannot deal with the context of existing signs/measures. In the presented general approach, this feature is derived as special case.
- **Custom preferences** may also be regarded, as the next example shows.

**Example 6** A government official is facing the scenario of Fig. 3 and knows that the two traffic signs are intended by a traffic measure that is not in the database. She specifies that these signs must not be deleted and obtains as single strict repair the scenario of Fig. 2, i.e., also the measure. ■

We now turn to some theoretical considerations.

## 3.2 Relation of Diagnoses and Repairs

We present an observation regarding *delete-only repairs*. If we delete a subset $J \subseteq I$ that meets every diagnosis of a set of conflicts $C$, then all causes of $C$ will be eliminated.

**Proposition 1** *Let $C \subseteq \mathcal{C}(\mathcal{T})$ be a set of conflicts and $\mathcal{D}_{\mathcal{T}}(C)$ be the set of its diagnoses. If a subset $J \subseteq I$ of the input is a hitting set for $\mathcal{D}_{\mathcal{T}}(C)$, then $C \not\subseteq \mathcal{C}(\mathcal{T}[J, \emptyset])$.*

For single conflicts, we directly obtain local repairs from their diagnoses.

**Corollary 1** *Let $c \in \mathcal{C}(\mathcal{T})$ be a single conflict. If $J \subseteq I$ is a hitting set for $\mathcal{D}_{\mathcal{T}}(c)$, then $(J, \emptyset)$ is a repair for $\{c\}$.*

For repairing multiple conflicts, we get the following result.

**Corollary 2** *If $J \subseteq I$ is a hitting set for $\mathcal{D}_{\mathcal{T}}(c)$ for each $c \in C$, then $(J, \emptyset)$ is a repair for $C$.*

In general, we cannot obtain repairs for $\mathcal{T}$ from diagnoses, due to nonmonotonic effects. That is, the removal of a cause of a conflict may introduce a different one.

## 3.3 Contexts and Independence of Conflicts

We now investigate when two sets of conflicts $C$ and $C'$ are not related. Due to space reasons, we occasionally give only informal definitions. Let $I$ be the input of $\mathcal{T}$. Then $C \subseteq \mathcal{C}(\mathcal{T})$ is called *independent* of $Y \subseteq I$, if, after removing from each diagnosis $J$ for $C$ arbitrary elements from $Y$, the result is also a diagnosis. This determines a *context* $ctx(C)$ as smallest subset $X \subseteq I$ that is not independent of $C$.

**Proposition 2** *Each set $C \subseteq \mathcal{C}(\mathcal{T})$ of conflicts has a unique context.*

**Proposition 3** *All $\subseteq$-minimal diagnoses for $C \subseteq \mathcal{C}(\mathcal{T})$ are contained in the context, i.e., $\bigcup \mathcal{D}_{\mathcal{T}}^{\subseteq}(C) \subseteq ctx(C)$.*

In general, the the converse of Prop. 3 does not hold. To fully characterize contexts by means of diagnoses, we need further definitions. Let $S, S'$ be two elements in a collection $\mathcal{X}$ of sets. We say $(S, S')$ is *convex in* $\mathcal{X}$, if $\{S'' \mid S \subseteq S'' \subseteq S'\} \subseteq \mathcal{X}$, and *maximal convex in* $\mathcal{X}$, if in addition it holds that for every pair $(T, T')$, where $T \subset S$ or $T' \supset S'$, $(T, T')$ is not convex in $\mathcal{X}$. By $\Diamond\mathcal{X}$ we denote the collection of maximal convex pairs of sets in $\mathcal{X}$.

The context of a set of conflicts is now characterized by the union of all $\subseteq$-minimal sets in maximal convex pairs of its diagnoses.

**Theorem 4** *Let $C \subseteq \mathcal{C}(\mathcal{T})$ be a set of conflicts. Then, $ctx(C) = \bigcup\{J \mid (J, J') \in \Diamond\mathcal{D}_{\mathcal{T}}(C)\}$.*

Thus, if the entire set of diagnoses is convex, then the converse of Prop. 3 also holds.

**Corollary 3** *If $\mathcal{D}_{\mathcal{T}}(C)$ is convex, then $ctx(C) = \bigcup \mathcal{D}_{\mathcal{T}}^{\subseteq}(C)$.*

We note that $\mathcal{D}_{\mathcal{T}}(C)$ is always convex if the employed logic is monotonic, i.e., the property that for all sets of formulas $T$ and $T'$ and atoms $\alpha$, $T \models \alpha$ implies $T \cup T' \models \alpha$. In such cases, we can calculate diagnoses first to obtain contexts and perform repairs locally there. This gives a first result towards distributed inconsistency management.

## 4 Complexity

We now present the complexity results of the following decision problems:

- CONS: Decide whether $\mathcal{T}$ is consistent, i.e., $\mathcal{C}(\mathcal{T}) = \emptyset$.
- UMINDIAG: Decide, given a set $C \subseteq \mathcal{C}(\mathcal{T})$ of conflicts, whether $C$ has a unique $\subseteq$-minimal diagnosis, i.e., a single minimal $J \subseteq I$ such that $C \subseteq \mathcal{C}_G^{P,Sp}(J)$.
- CORR: Decide whether $M$ and $S$ correspond, i.e., $\mathcal{F}_G^P(M) = \mathcal{F}_G^P(S)$.
- REPAIR: Decide, given that $\mathcal{T}$ is inconsistent, whether some admissible repair exists, i.e., some $I^-, I^+ \subseteq I_G$ such that $\mathcal{C}_G^{P,Sp}((I \setminus I^-) \cup I^+) = \emptyset$ and a polynomial-time admissibility predicate $\mathcal{A}(I^-, I^+)$ holds.

We consider different logics $\mathcal{L}$ for the traffic regulation $\Pi$:

(i) First-order logic under domain closure (FOL+DCA), i.e., an axiom $\forall x. \bigvee_{i=1}^{n}(x = c_i)$, where $c_1, \ldots, c_n$ is the finite set of constant symbols; and

(ii) Answer Set Programs, namely the following syntactic classes [2; 3]:

- *Stratified programs* (ASP$^{\neg_s}$)
- *Normal programs* (ASP$^{\neg}$)
- *Disjunctive programs* (ASP$^{\vee,\neg}$)

To obtain the complexity results for the listed decision problems, we make use of results for the following entailment problem [8]:

- IMPL: Decide, given a set $T$ of formulas (or rules) and an atom $\alpha$ in a fixed logic $\mathcal{L}$, whether $T \models \alpha$ holds.

|  | Logic $\mathcal{L}$ | IMPL | CONS | CORR | UMINDIAG | REPAIR |
|---|---|---|---|---|---|---|
| general | FOL+DCA | co-NEXP | $\mathsf{P}_{\parallel}^{\mathsf{NEXP}}$ |  |  | $\mathsf{P}^{\mathsf{NEXP}}$ |
|  | ASP$^{\neg_s}$ | EXP | EXP |  |  | EXP |
|  | ASP$^{\neg}$ | co-NEXP | $\mathsf{P}_{\parallel}^{\mathsf{NEXP}}$ |  |  | $\mathsf{P}^{\mathsf{NEXP}}$ |
|  | ASP$^{\vee,\neg}$ | co-NEXP$^{\mathsf{NP}}$ | $\mathsf{P}_{\parallel}^{\mathsf{NEXP}^{\mathsf{NP}}}$ |  |  | $\mathsf{P}^{\mathsf{NEXP}^{\mathsf{NP}}}$ |
| BPA | FOL+DCA | PSPACE | PSPACE |  |  | PSPACE |
|  | ASP$^{\neg_s}$ | $\mathsf{P}^{\mathsf{NP}}$ | $\mathsf{P}^{\mathsf{NP}}$ |  | in $\mathsf{P}_{\parallel}^{\Sigma_2^p}$, $\Pi_2^p$-hard | $\Sigma_2^p$ |
|  | ASP$^{\neg}$ | $\Pi_2^p$ | $\mathsf{P}_{\parallel}^{\Sigma_2^p}$ |  | in $\mathsf{P}_{\parallel}^{\Sigma_3^p}$, $\Pi_3^p$-hard | $\Sigma_3^p$ |
|  | ASP$^{\vee,\neg}$ | $\Pi_3^p$ | $\mathsf{P}_{\parallel}^{\Sigma_3^p}$ |  | in $\mathsf{P}_{\parallel}^{\Sigma_4^p}$, $\Pi_4^p$-hard | $\Sigma_4^p$ |

Table 1: Complexity of reasoning tasks. Top: General case. Bottom: With bounded predicate arities (BPA). Unless stated otherwise, entries are completeness results.

**Theorem 5** *The complexity results in Table 1 hold.*

A class $\mathsf{P}_{\parallel}^{O}$ is the restriction of $\mathsf{P}^{O}$ that all oracle queries have to run in parallel. Our examination shows the high computational cost of solving of the presented reasoning tasks. Towards inconsistency management tasks of lower complexity, it would be interesting to study similar reasoning tasks at a less generic level, where further properties may be exploited.

## 5 Implementation

Since comprehensible specifications play a major role in this problem domain, we demand that the implementation should be *declarative*. Further, we need a high degree of *modularity*. Changing specifications should not require changes in the implementation of reasoning tasks. Since the domain comprises many patterns with exceptions, some sort of *default reasoning* would be desirable; e.g., traffic is permitted in a given direction, *unless* it is explicitly prohibited. Therefore, Answer Set Programming (ASP) is a natural choice, given efficient solvers such as DLV [7] and the Potassco suite [5].

### 5.1 Uniform Approach for Reasoning Tasks

We first observe that, by encoding the traffic regulation problem $\mathcal{T} = (\Pi, Sp)$ as sketched in Section 2.1, we immediately obtain a solution for consistency evaluation by the program

$$\Pi \cup G \cup I\,, \tag{7}$$

where $\Pi = P \cup Sp$ and $I = M \cup S$. From the resulting unique answer set, we simply read off the atoms of form $c(t, v)$ and get the conflicts $\mathcal{C}(\mathcal{T}) = \mathcal{C}_G^{P,Sp}(I)$.

Next, we observe that computing diagnoses (resp. repairs) for $C \subseteq \mathcal{C}(\mathcal{T})$ amounts to *guessing* measures/signs on $G$ within a certain *pool*, and then *checking* whether all (resp. no) conflicts in $C$ are entailed. That is, we can uniformly solve the reasoning tasks as follows.

- We view all measures/signs $x \in I$ as terms and replace their occurrences in $I$ by atoms of form $\mathit{input}(x)$ and in $\Pi$ by $\mathit{use}(x)$. We denote the resulting programs by $\mathit{input}(I)$ and $\mathit{use}(\Pi)$, respectively.
- Now, consider the following *guess* program for $\mathcal{T}$:

$$\mathit{guess}(\mathcal{T}) \;=\; \mathit{use}(\Pi) \cup G \cup \mathit{input}(I) \cup \mathit{Pool}\,, \tag{8}$$

  where $\mathit{Pool}$ consists at least of the rules

$$\mathit{pool}(I) \;\leftarrow\; \mathit{input}(I) \tag{9}$$

$$\mathit{use}(I) \,\vee\, \neg\mathit{use}(I) \;\leftarrow\; \mathit{pool}(I)\,. \tag{10}$$

  This program computes for each subset $J \subseteq I$ an answer set containing its entailed conflicts $\mathcal{C}_G^{P,Sp}(J)$.
- Given simple additional modules $X$, each reasoning task can be solved by a program of the form

$$\mathit{guess}(\mathcal{T}) \cup X\,. \tag{11}$$

For instance, to solve consistency evaluation, we only evaluate the entire input by taking $X = \{\mathit{use}(I) \leftarrow \mathit{input}(I)\}$. To compute the diagnoses for $C \subseteq \mathcal{C}(\mathcal{T})$ we employ the constraints $X = \{\leftarrow \mathrm{not}\, c(t, v) \mid c(t, v) \in C\}$ to *check* that all conflicts are entailed. Similar ideas work for all reasoning tasks, as shown formally in the thesis.

## 6 Conclusion

To date, tools for advanced inconsistency management of traffic regulations are lacking. We presented a logic-based approach to this problem, which is highly relevant in today's increasingly dynamic settings. We formalized the notion of a traffic regulation problem and defined major reasoning tasks on it, whose computational complexity we characterized for different logic languages. Moreover, we studied several theoretical properties in relation to diagnoses, repairs and contexts of conflicts. Finally, we presented a systematic approach for a modular, flexible implementation using ASP.

A realization is part of an ongoing industrial project with *PRISMA solutions*,[2] where the integration of the presented prototype within the existing Java-based web application remains to be done. The resulting system should give clear benefits for traffic and transport authorities in their planning and maintenance tasks. Moreover, the developed methods may be utilized for coordination and verification in autonomous systems in active traffic management.

[2]`http://www.prisma-solutions.at`

## References

[1] Gerd Brewka, Thomas Eiter, and Miroslaw Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[2] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[3] Thomas Eiter, Wolfgang Faber, Michael Fink, and Stefan Woltran. Complexity Results for Answer Set Programming with Bounded Predicate Arities. *Annals of Mathematics and Artificial Intelligence*, 51(2-4):123–165, 2007.

[4] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web*, volume 5689 of *Lecture Notes in Computer Science*, pages 40–110. Springer, 2009.

[5] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Thomas Schneider. Potassco: The Potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.

[6] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[7] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7:499–562, 2002.

[8] Harry R. Lewis. Complexity results for classes of quantificational formulas. *J. Comput. Syst. Sci.*, 21(3):317–353, 1980.