Inconsistency Management for Traffic Regulations: Formalization and Complexity Results[⋆]

Harald Beck, Thomas Eiter, and Thomas Krennwallner

Institute of Information Systems, Vienna University of Technology
Favoritenstrasse 9–11, A-1040 Vienna, Austria
{hbeck,eiter,tkren}@kr.tuwien.ac.at

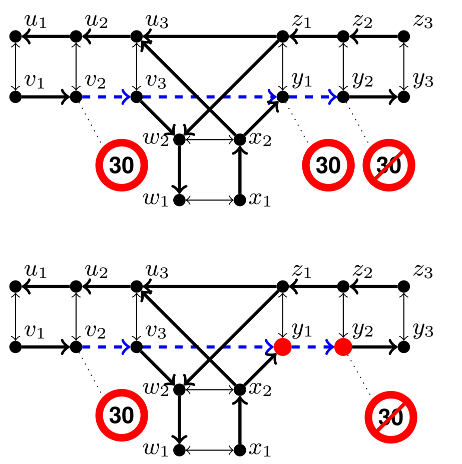
Abstract. Smart Cities is a vision driven by the availability of governmental data that fosters many challenging applications. One of them is the management of inconsistent traffic regulations, i.e., the handling of inconsistent traffic signs and measures in urban areas such as wrong sign posting, or errors in data acquisition in traffic sign administration software. We investigate such inconsistent traffic scenarios and formally model traffic regulations using a logic-based approach for traffic signs and measures, and logical theories describe emerging conflicts on a graph-based street model. Founded on this model, we consider major reasoning tasks including consistency testing, diagnosis, and repair, and we analyze their computational complexity for different logical representation formalisms. Our results provide a basis for an ongoing implementation of the approach.

1 Introduction

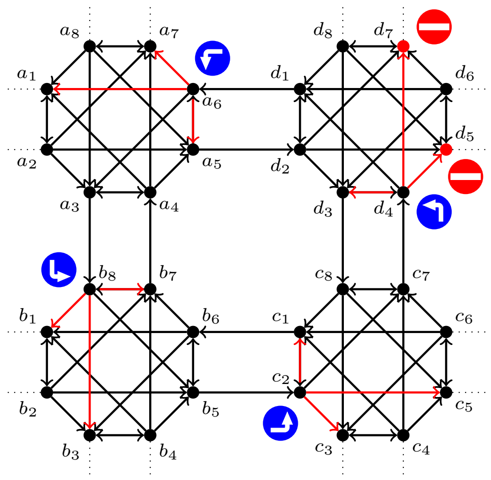
The advent of the World Wide Web and distributed systems brought numerous new methods for intelligent management of data and knowledge. With initiatives such as Open Government Data (http://opengovernmentdata.org/) the idea of Smart Cities has been gaining interest in research communities, with many innovative applications in ecological and city planning areas. Local governments manage their posted traffic signs and measures using software tools, i.e., authorities enact rules how traffic on urban streets and places should be regulated, and employees increasingly maintain this information with the help of specialized software. An important task is the management of inconsistent traffic regulations.

Example 1. Consider the T-junction in the top of Fig. 1a. It has three arms, each represented by two parallel lanes: u_3 to u_1 and v_1 to v_3, w_2 to w_1 and x_1 to x_2, and y_1 to y_3 and z_3 to z_1. We can turn from one arm to each other and may reverse between nodes connected by edges with two arrows. The traffic signs at v_2, y_1, and y_2 symbolize a correct sign posting for a speed limit measure of 30 km/h, indicated by the dashed blue path from v_2 to y_2. The *effect* expressed by both the measure and the signs is that along the edges (v_2, v_3), (v_3, y_1), (y_1, y_2), the maximum speed allowed for any road user is 30 km/h. The recurrent start sign at y_1 is necessary, as road users coming from x_2,

⋆ Supported by PRISMA solutions EDV-Dienstleistungen GmbH, and the Austrian Science Fund (FWF) projects P20841 and P24090.



(a) Top: correct sign posting for a speed limit measure (dashed blue path). Bottom: Inconsistency: no recurrence of start sign at y_1



(b) Loop caused by mandatory left turns

Fig. 1: Traffic Regulation Scenarios

turning into the lane starting at y_1, would otherwise be unaware of the speed limit. This situation is shown in the bottom of Fig. 1a. The effect of the start sign at v_2 can only be propagated to the arm starting at y_1, since y_1 is also reachable from the arm ending in x_2. We get an inconsistent traffic regulation due to two conflicts: the speed limit effect ends at y_1 without an end sign, and the effect discontinuing at y_2 does not properly start.

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 software may lead to wrong assumptions on the state of traffic regulations. Tools that detect, prohibit, and correct such errors are in need to help public administration with their traffic management tasks.

In order to gain new insights from available sign posting data, formal methods from knowledge representation and reasoning proved to be a key to attack issues that arise when data is inconsistent [16, 12, 10]. Many issues arise in the context of traffic regulations. Traffic measures, i.e., intended constraints given as regulations on the traffic, may oppose the state of traffic sign posting, which can be seen as real-world constraints that announce what is allowed on the street. One natural question is how to find inconsistencies when combining traffic measures and street signs. Such questions become even more complex in dynamic environments, i.e., when so-called *active traffic management* comes into play. For instance, variable-message signs on motorways manage the traffic flow by varying speed limits based on events like traffic congestions, or weather conditions like fog or black ice. Contradicting speed limits may be posted by operators of such message signs, leading to aforementioned legal issues.

Finding such errors is not trivial in real life situations and many subtle inconsistencies may occur. When an inconsistency is found, one usually wants to diagnose and repair it. To the best of our knowledge, there is no automated support for inconsistency finding in complex traffic regulations. Already the seemingly simple scenario in Example 1 shows the need for (semi-)automatic tool support in traffic regulation maintenance software. Different from the issues above is the problem of modeling transportation and traffic in a formal representation. Legal texts are ambiguous and often implicitly understood, and no single characterization has yet shown to be advantageous over others.

This motivates this work, which makes the following contributions:

- We analyze the problem domain and identify main concepts and notions such as traffic signs, measures, effects, and inconsistencies in traffic regulation orders.
- Building upon well-known literature in abductive reasoning and model-based diagnosis [16, 12, 10], we develop a formal model using predicate logic for traffic signs and measures, and introduce the formal notion of a *traffic regulation problem*.
- For traffic regulation problems, we consider major reasoning tasks, viz. inconsistency detection, diagnosis, correspondence between measures and signs, and repair.
- We then study and characterize the computational complexity of these reasoning tasks, for different representation formalisms. In particular, we consider first-order (FO) logic under domain closure (as the domain of discourse is fixed), and answer set programming (ASP). The latter is convenient for developing executable specifications, and provides attractive features that can be used for default rules and exception handling.

This work is embedded in an industrial context, dealing with specialized software, which is used by local government departments and allows for the visualization and administration of traffic regulations. The results of this research may assist to find inconsistencies and should give a clear advantage over simple traffic sign acquisition and storage tools. A prototype implementation using answer set programming is in progress.

2 Domain Analysis

In this section we briefly analyze the domain of traffic regulations, measures and signs.

A *traffic regulation* is a legal document describing how road users can use the street and how these usages can be restricted by means of *traffic signs*. The legal act to introduce new traffic signs, or to remove existing ones, is a *traffic regulation order*, which comes in form of a document describing (in natural language) a *traffic measure* that has to be taken to reach a desired effect, i.e., a restriction of road usage. This measure has to be *announced* by traffic signs and becomes legally effective as soon as the corresponding signs are posted on the street. We view road markings as special cases of traffic signs.

The restrictions described by measures and signs include speed limits, driving bans, parking or halting bans, prohibited or mandatory driving directions, information about zones like residential areas and pedestrian zones, motorways, and so on. We base our work on the Austrian traffic regulation and its potential measures and signs, which can be found at http://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10011336. However, we focus on general aspects that are not bound to regional differences.

Inconsistencies. In general, a set of traffic regulation orders, resp. the resulting measures and signs, can lead to *conflicts* with respect to the traffic regulation. The aim of our work is to *detect* such inconsistencies, to *diagnose* and to *repair* them.

For instance, in Austria it is not allowed that a motorway overlaps with a residential area. In view of traffic signs this means that, when driving on a motorway, the end sign must precede the start sign of the residential area. In addition to such illustrative cases, complications arise quickly when many different kinds of restrictions are expressed.

What we understand by a conflict does not necessarily stem from the traffic regulation, but can also come from supplementary documents of expert knowledge such as traffic planning experience. It is thus our aim to provide a system that can detect different sorts of conflicts in a modular and easily extendable way. Whenever a conflict is detected, we want to provide the user with diagnostic information, explaining which measures or signs caused it. Finally, we want to offer a repair mechanism that suggests by which modifications compliance with the specification can be established.

Data Model and Approach. To achieve these goals, we first need a street model based on which we can express measures and signs, and the restrictions expressed by them. We will view streets as *directed graphs*, where edges represent the potential direction of traffic. Each edge will get a unique label to discern whether it represents a part of a lane, a turn over a junction or a U-turn. Any digital street map from which this view can be generated can be used as potential database.

By an *effect* of both measures and signs we understand the implicit restrictions they express. To reflect measures and signs (from a database or user input) in the street graph, we will use predefined labels on the edges (for measures) and nodes (for signs). Similarly, we will represent arising inconsistencies by associating nodes with specific *conflict labels*. Both the mapping from measures and signs to effects and from effects to conflicts will be established in a modular way by means of logic formulas. The conflict labeling can be used to visualize inconsistencies on a street map, followed by user interaction in connection with diagnosis and repair.

Challenges. Many conflicts will not be strictly illegal as defined by the traffic regulatory orders or additional legal documents, but arise from expert knowledge or common sense. Consequently, both the inclusion and the kind of definition of many conflicts will be a matter of preference, and shall in principle be configurable by domain experts.

To show the need for advanced reasoning support, consider the traffic regulation problem in Fig. 1b, where a *loop* is induced by four mandatory left turns. We consider loops as special dead ends which we want to detect whenever a node v has a way in, but no way out. We say a node v has a *way in*, if it is predefined as *in-node*, or if it is reachable from an in-node. Similarly, a node v has a *way out*, if it is a predefined *out-node*, or an out-node is reachable from v.

A node w is *reachable* from v, if (i) (v, w) is an edge, where neither the node v is prohibited for traffic (e.g., through a no-entry sign), nor the edge itself (e.g., through a mandatory turn in a different direction), or (ii) if a node x is reachable from v, from which w is reachable.

Example 2. The mandatory left turns in Fig. 1b induce a loop along the nodes $L = \{a_6, a_3, b_8, b_5, c_2, c_7, d_4, d_1\}$. Respective in-nodes and out-nodes are not depicted and assumed to be reachable from the nodes with dotted lines. For instance, from a_1, an out-node is reachable, and a_2 is reachable from an in-node. Each mandatory left turn prohibits the right turn, U-turn, and edge straight ahead over the junction. E.g., the mandatory left turn at a_6 prohibits moves along the edges (a_6, a_7), (a_6, a_5) and (a_6, a_1).

We try to keep the street model as simple as possible. Reversing along lanes is not a typical road usage. Therefore, we do not model any intermediate nodes along streets (and thus no U-turns within lanes), unless we need to represent a sign, or the start or end of a measure. In reality, we could in principle escape the loop in Example 2 by reversing somewhere along a lane. Since this is not supposed to be necessary, we still want an evaluation to report that there are problems, by noting conflicts on the nodes $v \in L$.

Intuitively, the unique minimal explanation (i.e., a *diagnosis*) for each of these problematic nodes—which can be seen as one conflict across many nodes—consists of all four mandatory left turns. Note that additional signs that do not restrict the reachability, like speed limits, would not change this diagnosis.

To *repair* the scenario, i.e., make modifications such that the result is free of conflicts, we may delete one of the mandatory left turns on nodes a_6, b_8, or c_2. However, other possibilities exist (see Section 6).

3 Formal Model

In this section we formalize our data model and formulate a traffic regulation problem based on it. Throughout, we assume that a version of predicate logic $\mathcal{L}$ with negation is fixed, in which the desired specification can be expressed (e.g., FO logic or ASP).

Definition 1 (Street graph). *A* (street) graph *is a connected, labeled, directed graph $G = (V, E, \ell)$ of nodes V, edges $E \subseteq V \times V$, and a labeling function ℓ that assigns each edge $(v, w) \in E$ a unique label $\ell(v, w) \in \{\mathit{left}, \mathit{straight}, \mathit{right}, \mathit{lane}, \mathit{uturn}\}$.*

We identify G with the set of atoms $e(t, v, w)$, where t is the label of the edge (v, w). Several assumptions about the structure of these graphs are made. For instance, we intend to model junctions by means of edges with labels left, right and $\mathit{straight}$. For each such edge (v, w), all incoming edges $(x, v) \in E$ to node v are labeled uturn or lane.

Example 3 (cont'd). Fig. 1a suggests how the edge labels ought to be used. For instance, the edge (v_3, y_1) models the direction straight ahead over a junction and thus gets the label $\mathit{straight}$. All other edges (v_i, v_{i+1}) and (y_i, y_{i+1}) are labeled with lane. The incoming street from below has a turn to the right starting at x_2 and ending at y_1, which will be modeled by an atom $e(\mathit{right}, x_2, y_1)$. Similarly, we use $e(\mathit{left}, x_2, u_3)$ for the left turn at x_2. The edges with arrows on both ends depict U-turns in both directions.

In the formulation of measures in traffic regulation orders concepts like street names, addresses and cardinal points are used to describe the intended topological dimensions. We assume that for the description at hand, a preprocessing (or specification) maps this scope to edges. We thus reduce measure descriptions to sets of such "atomic measures."

To describe measures, signs, their effects, as well as conflicts, we build upon disjoint sets of ground terms M, S, F and C called the *measure types, sign types, effect types* and *conflict types*, respectively. For instance, M may contain a set of terms $\mathit{spl}(k)$ for each speed limit value k that is needed, e.g., $\mathit{spl}(5)$, $\mathit{spl}(10)$, $\ldots$, $\mathit{spl}(130)$ in Austria.

Definition 2 (Measures, Signs, Effects, Conflicts). *Given a street graph G, we define the following 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* $F_G = \{f(t,v,w) \mid t \in \mathsf{F}, (v,w) \in E\}$*; and*
- *Conflicts* $C_G = \{c(t,v) \mid t \in \mathsf{C}, v \in V\}$.

For instance, to represent the prohibited case that a motorway overlaps with a residential area at a node v we might use $c(\mathit{overlap}(\mathit{motorway}, \mathit{residential\text{–}area}), v)$. Similarly, the fact that one is caught in a dead end or loop at u can be represented as $c(\mathit{no\text{-}way\text{-}out}, u)$.

Definition 3 (Scenario). *Let G be a street graph,* $M \subseteq M_G$ *be a set of measures on G, and* $S \subseteq S_G$ *be a set of signs on G. Then,* $Sc = (G, M, S)$ *is called a* scenario.

Example 4 (cont'd). In Fig. 1a, the dashed blue path from v_2 to y_2 symbolizes a 30 km/h speed limit measure. We formalize this as a set of atomic measures $\{m(\mathit{spl}(30), v_2, v_3)$, $m(\mathit{spl}(30), v_3, y_1)$, $m(\mathit{spl}(30), y_1, y_2)\}$. The depicted traffic signs are defined at nodes as the set $\{s(\mathit{start}(\mathit{spl}(30)), v_1)$, $s(\mathit{start}(\mathit{spl}(30)), y_1)$, $s(\mathit{end}(\mathit{spl}(30)), y_2)\}$.

Effects and Conflicts. The meaning of both measures and signs is captured by a mapping of the according languages to a common target language of effects. To assist modular composition, we define $\overline{X}_Y = X \cup \{\neg x \mid x \in Y \setminus X\}$ as the *closed world operator* applied to a set of ground atoms X relative to a *base set* $Y \supseteq X$. We always use the according base set of Definition 2, and thus omit the subscript, e.g., $\overline{M}$ for a set measures M on G abbreviates $\overline{M}_{M_G}$. The base set assumed for the completion $\overline{G}$ of any graph G is the set of all atoms $e(t,v,w)$. We introduce another operator Cn that maps between atoms. Let X and Y be sets of atoms (on G) and let T be a set of formulas in $\mathcal{L}$.

Definition 4 ($Cn_G(T,X,Y)$). *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 (logical) consequence relation in the underlying logic $\mathcal{L}$. The closed world operator makes sure that atoms that are not entailed are set to false, and thus ensures that valuations of atoms in Y are unique. This restriction will enable modular composition by means of a two-stage approach, which we will describe next.

Definition 5 (Effect mapping). *An* effect mapping *is a set* P *of formulas in* $\mathcal{L}$ *that associates with each input* $I \subseteq I_G$ *on a street graph G the set* $\mathcal{F}^P_G(I) = Cn_G(P, I, F_G)$ *of atoms, called* effects of I (on G).

We implicitly assume that effect mappings use the ranges of terms appropriately.

Example 5. The first-order sentence

$$\forall k, x, y\, (m(\mathit{spl}(k), x, y) \supset f(\mathit{max\text{-}speed}(k), x, y))$$

of an effect mapping P captures the meaning of speed limit (spl) measures. We informally describe when this effect label is obtained by signs: first, an edge (x,y) is labeled with $\mathit{max\text{-}speed}(k)$, if an start sign $s(\mathit{start}(\mathit{spl}(k), x))$ is placed at x. From there, the effect is propagated in the direction of traffic, i.e., along the edges with label $lane$, until an end sign or a junction is reached. For the latter case, let $e(lane, u', u)$ be the last edge before the junction and $e(straight, u, v)$ be the next edge in the direction ahead. The effect continues after the crossroads on the (unique) edge $e(lane, v, w)$ only if another start sign is posted on v, or no edge (x, v) with label *left* or *right* permitted for traffic exists (and neither an end sign nor the start sign for a different speed limit is at v).

The effect mapping uses measures and signs on a graph to derive effects. Likewise, these effect atoms will then be used to infer conflicts by means of a specification.

Definition 6 (Conflict specification). *A* conflict specification over an effect mapping P *is a set* Sp *of formulas in* $\mathcal{L}$ *that associates with each input* $I \subseteq I_G$ *on a street graph* G *the set* $\mathcal{C}_G^{P,Sp}(I) = Cn_G(Sp, \mathcal{F}_G^P(I), C_G)$ *of atoms, called* conflicts of I (on G).

Thus, the setup to compute conflicts based on effects given a conflict specification, is the same as computing effects from measures and signs, given an effect mapping. The first stage builds a context-dependent model of the input, the second stage establishes the basis for reasoning tasks. There is no explicit support for query answering on top of conflicts; however, queries on aspects of interest may be encoded in the conflict specification, using designated conflicts and formulas defining them (e.g. rules) in Sp. This way, given an input I on a graph G, querying for a certain conflict type (or aspect of interest) $t \in \mathsf{C}$ amounts to computing the set $\{c(t, v) \in \mathcal{C}_G^{P,Sp}(I) \mid v \in V\}$.

Example 6 (cont'd). Fig. 1a (bottom) depicts the situation in which the intended speed limit is not sufficiently announced. Road users coming from node x_2, turning right into the lane starting at y_1 are not informed about the speed limit. Hence, according to the sign posting, the $max\text{-}speed(30)$ effect cannot be associated with edge (y_1, y_2). Since we have a $max\text{-}speed(30)$ effect until node y_1 but no end sign mapped to it, we have a conflict which we may represent as $c(no\text{-}end(max\text{-}speed(30)), y_1)$. The end sign posted at y_2 leads to a second conflict, since there is no "open" effect anymore: $c(cant\text{-}end(max\text{-}speed(30)), y_2)$. Using answer set programming, with uppercase letters denoting variables as usual, the latter conflict can be defined by the rule

$$c(cant\text{-}end(F), V_2) \leftarrow in\text{-}dir(V_1, V_2), s(end(T), V_2), m2f(T, F), \text{not } f(F, V_1, V_2)\ ;$$

where $in\text{-}dir$ represents an edge of type *straight* or *lane*, the atom $s(end(T), V_2)$ stands for a traffic sign posted at node V_2, ending a measure of type T, and $m2f$ encodes domain knowledge that T is associated with effect type F.

Note that ASP solvers like DLV support query functionalities in the aforementioned sense. In the previous example, we might ask $c(cant\text{-}end(F), V)$? and get the terms $max\text{-}speed(30), y_2$ as result, matching $c(cant\text{-}end(max\text{-}speed(30)), y_2)$.

Definition 7 (Traffic Regulation Problem). *Let* Sp *be a conflict specification over an effect mapping* P*, and let* Sc *be a scenario. Then, the pair* $\Pi = (Sp, P)$ *is called a* traffic regulation *and the pair* (Π, Sc) *a* traffic regulation problem.

4 Reasoning Tasks

We now use the preceding definitions to specify some practically relevant use cases in form of reasoning tasks. In the sequel, we let $\mathcal{T} = (\Pi, Sc)$ be a traffic regulation problem with a traffic regulation $\Pi = (Sp, P)$ and a scenario $Sc = (G, M, S)$, and $I = M \cup S$.

Definition 8 (Inconsistency). *The* conflicts *of $\mathcal{T}$ are given by $\mathcal{C}(\mathcal{T}) = \mathcal{C}_G^{P,Sp}(I)$. If $\mathcal{C}(\mathcal{T}) \neq \emptyset$, we call $\mathcal{T}$* inconsistent*.*

Additionally, we call every set of measures or signs $X \subseteq I_G$ on graph G *inconsistent*, if $\mathcal{C}_G^{P,Sp}(X)$ is non-empty. Given an inconsistent $\mathcal{T}$, we are interested which part of the input, i.e., which hypotheses, explain the conflict observations.

Definition 9 (Diagnosis). *For inconsistent $\mathcal{T}$, a* diagnosis *of a set of conflicts $C \subseteq \mathcal{C}(\mathcal{T})$ is a set $J \subseteq I$, such that $C \subseteq \mathcal{C}_G^{P,Sp}(J)$.*

To see the relation of diagnosis with the usual notion of abductive diagnosis [15, 3], we recall the definition of the latter.[1] An *abductive diagnosis problem (ADP)* is a triple $\langle T, H, O \rangle$, where T is a set of formulas in $\mathcal{L}$, called the theory, and H and O are sets of literals, called the hypotheses and observations, respectively. A *(complete) abductive diagnosis for* $\langle T, H, O \rangle$ is a set $A \subseteq H$, such that $T \cup \overline{A} \not\models \bot$ and $T \cup \overline{A} \models O$. We note that an input $J \subseteq I$ is the abductive diagnosis for the ADP $\langle P \cup \overline{G}, I, \mathcal{F}_G^P(J) \rangle$.

Proposition 1. *Let $C \subseteq \mathcal{C}(\mathcal{T})$ and $J \subseteq I$. The effects $\mathcal{F}_G^P(J)$ are an abductive diagnosis for the ADP $\langle Sp \cup \overline{G}, \mathcal{F}_G^P(I), C \rangle$ iff J is a consistent diagnosis of C, i.e., $Sp \cup \overline{G} \cup \overline{\mathcal{F}_G^P(J)} \not\models \bot$.*

Since I is always a trivial (but non-informative) diagnosis for any set of conflicts, we are interested in (subset-)*minimal* diagnoses. We omit a formal definition of Π serving the forthcoming examples.

Example 7 (cont'd). The missing sign at y_1 leads to two conflicts. The minimal diagnosis for the missing sign $\{c(\textit{no-end}(\textit{max-speed}(30)), y_1)\}$ is $\{s(\textit{start}(\textit{spl}(30)), v_2)\}$. Independently, the other conflict $\{c(\textit{cant-end}(\textit{max-speed}(30)), y_2)\}$ is minimally explained by $\{s(\textit{end}(\textit{spl}(30)), y_2)\}$.

Usually, we desire that measures and signs in a scenario express the same effects.

Definition 10 (Correspondence). *A set of measures M and a set of signs S* correspond *with respect to P and G, if it holds that $\mathcal{F}_G^P(M) = \mathcal{F}_G^P(S)$.*

The next example shows the significance of correspondence besides consistency.

Example 8 (cont'd). Suppose a no-right turn sign on x_2 is added to the traffic regulation problem in Fig. 1a, bottom. This results in a consistent scenario, since in this case, the node y_1 can only be reached from v_3 (reversing at z_1 along the U-turn (z_1, y_1) is disregarded). Hence, another start sign at y_1 is not necessary and the effect propagation of the start sign at v_2 continues through y_1. However, the prohibition of traffic along the edge (x_2, y_1) as supported by the no-right turn sign is not supported by a corresponding measure. This problem can only be seen by additionally testing for correspondence.

For each of the definitions in this section, we immediately obtain a *reasoning task* which requires the computation of the respective concept.

Repair. Complementary to diagnoses explaining the cause for inconsistency, a natural question is how to *repair* an inconsistent traffic regulation problem, i.e., by means of which deletions and additions of measures and signs consistency can be established. In the general case, we might delete and add both measures and signs.

[1] Strictly speaking, we present a slightly modified version using the closed world operator.

Definition 11 (Repair). *A* repair *of an (inconsistent) $\mathcal{T}$ is a pair (I^-, I^+) such that $I^- \subseteq I$, $I^+ \subseteq I_G \setminus I$, and $\mathcal{C}_G^{P,Sp}(I') = \emptyset$, where $I' = (I \setminus I^-) \cup I^+$.*

A repair yields a traffic regulation problem $\mathcal{T}'$ replacing I with consistent I' in $\mathcal{T}$.

Example 9 (cont'd). A repair for the inconsistent $\mathcal{T}$ with the traffic regulation scenario as shown in Fig. 1a (bottom) is $(\emptyset, \{s(start(spl(30), x_4)\})$.

Usually, one requires correspondence after a repair, i.e., that $M' = I' \cap M_G$ and $S' = I' \cap S_G$ correspond with respect to P and G; we call such repairs *strict*. Furthermore, the candidate space (I^-, I^+) might be restricted; in this way, further practically relevant reasoning tasks can be formulated as special cases of repair. For instance, if we are given a scenario with consistent M, but inconsistent S, we may *adjust* the signs by restricting the repair to modify only signs. Or, related to data imports, we may want to *generate* measures from scratch, given only signs, or vice versa a sign posting, given only measures. Note that these additional reasoning tasks need no separate implementation.

5 Computational Complexity

In this section, we analyze the computational complexity of decision problems associated to the reasoning tasks above. In particular, we consider for a given traffic regulation problem $\mathcal{T} = (\Pi, Sc)$, with $\Pi = (Sp, P)$ and $Sc = (G, M, S)$

- CONS: decide whether $\mathcal{T}$ is consistent, i.e., $\mathcal{C}(\mathcal{T}) = \mathcal{C}_G^{P,Sp}(I) = \emptyset$;
- 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 $\mathcal{T}$ 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 these problems for different mapping formalisms $\mathcal{L}$, viz. (1) FO predicate 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 (2) (function-free) Answer Set Programs[2] under cautious consequence, i.e., $P \models \alpha$ iff α is true in all answer sets of P; here, we consider various classes, including (a) stratified programs ($\mathrm{ASP}^{\neg_s}$), (b) normal programs (i.e., arbitrary negation, $\mathrm{ASP}^{\neg}$), and (c) disjunctive programs (with head disjunction and arbitrary negation, $\mathrm{ASP}^{\vee,\neg}$).

We assume that the reader is familiar with the basic concepts of complexity theory (cf. [14]),and recall that P^O (resp. NP^O) is (nondeterministic) polynomial time computability with an oracle for complexity class O, and Σ_i^p, $i \geq 1$, are classes of the polynomial hierarchy where $\Sigma_1^p = \mathsf{NP}$ and $\Sigma_{i+1}^p = \mathsf{NP}^{\Sigma_i^p}$. Furthermore, $\mathsf{P}_\parallel^O$ is the restriction of P^O that all oracle queries are independent of each other, i.e., they are evaluable in parallel.

The computational complexity of the atom entailment problem in these logics (IMPL), i.e., deciding $T \models \alpha$ for a set of formulas (resp. program) T and an atom α, is shown in the first column of Table 1. Besides the general case, also the one of bounded predicate

[2] The use of function symbols as in the examples is convenient but not essential for this domain.

Logic $\mathcal{L}$	IMPL	CONS	CORR	UMINDIAG	REPAIR
FO+DCA	co-NExp / PSpace	$\mathsf{P}_{\parallel}^{\mathsf{NExp}}$ / PSpace			$\mathsf{NP}^{\mathsf{NExp}}$ / PSpace
$\mathrm{ASP}^{\neg_s}$	Exp / P^{NP}	Exp / P^{NP}		Exp / in $\mathsf{P}_{\parallel}^{\Sigma_2^p}$, Π_2^p-hard	Exp / Σ_2^p
$\mathrm{ASP}^{\neg}$	co-NExp / Π_2^p	$\mathsf{P}_{\parallel}^{\mathsf{NExp}}$ / $\mathsf{P}_{\parallel}^{\Sigma_2^p}$		$\mathsf{P}_{\parallel}^{\mathsf{NExp}}$ / in $\mathsf{P}_{\parallel}^{\Sigma_3^p}$, Π_3^p-hard	$\mathsf{NP}^{\mathsf{NExp}}$ /Σ_3^p
$\mathrm{ASP}^{\vee,\neg}$	co-$\mathsf{NExp}^{\mathsf{NP}}$ / Π_3^p	$\mathsf{P}_{\parallel}^{\mathsf{NExp}^{\mathsf{NP}}}$ / $\mathsf{P}_{\parallel}^{\Sigma_3^p}$		$\mathsf{P}_{\parallel}^{\mathsf{NExp}^{\mathsf{NP}}}$ / in $\mathsf{P}_{\parallel}^{\Sigma_4^p}$, Π_4^p-hard	$\mathsf{NP}^{\mathsf{NExp}^{\mathsf{NP}}}$ / Σ_4^p

Table 1: Complexity of reasoning tasks (general case / bounded predicate arities); unless stated otherwise, entries are completeness results

arities (BPA) is printed, i.e., when the arities of predicates is bounded by some constant. Briefly, as for FOL+DCA, a countermodel of $T \models \alpha$ (of exponential size) can be guessed and verified in polynomial space (in the size of T and α); hardness follows, e.g., from the complexity of satisfiability of the Bernays-Schönfinkel fragment of FOL. Under BPA, the model guess has polynomial size, and thus the whole countermodel check is feasible in polynomial space; PSpace-hardness is inherited from evaluation of a given FOL formula over a finite structure. For the ASP^X languages, see [4, 5].

Theorem 1. *For* CONS*,* UMINDIAG*,* CORR*, and* REPAIR *the results in Table 1 hold.*

In the rest of this section, we explain the results and outline how they can be derived. We make use of the following known fact; let $\mathsf{P}_{\parallel[k]}^{O}$ be the restriction of P^{O} such that the oracle calls amount to k rounds of parallel (independent) oracle calls.

Lemma 1. *For $O = \mathsf{NExp}, \mathsf{NExp}^{\mathsf{NP}}, \Sigma_i^p$, $i \geq 1$, and constant k, $\mathsf{P}_{\parallel[k]}^{O} = \mathsf{P}_{\parallel[1]}^{O} = \mathsf{P}_{\parallel}^{O}$.*

CONS. To decide problem CONS, we must test whether some of the (polynomially many) conflict facts $c(t, v)$ is derivable from the conflict specification Sp over the effect mapping P. To this end, we can determine for each of the (polynomially many) effect facts $f(t, v, w) \in F_G$ whether $f(t, v, w) \in \mathcal{F}_G^P(I) = Cn_G(P, I, F_G)$ with an oracle for IMPL, and then decide whether $c(t, v) \in \mathcal{C}_G^{P,Sp}(I) = Cn_G(Sp, \mathcal{F}_G^P(I), C_G)$ with an oracle for IMPL. This is a polynomial time computation with two rounds of parallel evaluation of oracle queries with complexity O; this puts the problem in the respective complexity class $\mathsf{P}_{\parallel[2]}^{O}$. For $O = \mathsf{Exp}, \mathsf{PSpace}$, this class coincides with O and for $O = \mathsf{P}^{\mathsf{NP}}$ with P^{NP} (as oracles can be simulated, in case of an P^{NP} oracle with an O oracle); for the other classes, by the lemma it coincides with $\mathsf{P}_{\parallel}^{O}$.

The $\mathsf{P}_{\parallel}^{O}$-hardness results for CONS are derived by a reduction from the following $\mathsf{P}_{\parallel}^{O}$-complete problem EVEN: given instances $I_0, \ldots, I_{2n+1}$, $n \geq 0$, of a (fixed) O-complete problem, decide whether the number of yes-instances among them is even. Here, without loss of generality, we can assume that all yes-instances precede all no-instances. To encode this problem, we use problem IMPL and restrict (wolog) to instances $I_j : T^{(j)} \models \alpha_j$ in a way such that their answers amount to $P \models \phi_j$ for effect mapping P and effect ϕ_j. We then design the conflict specification Sp to derive a conflict fact χ if and only if the maximum index of a yes-instance is even. To this end, we can use in the ASP cases rules $\chi \leftarrow \phi_{2j}, \mathrm{not}\, \phi_{2j+1}$, where $0 \leq j \leq n$, and in the FOL case simply the formula $\chi \leftrightarrow \bigvee_{j=0}^{n} \phi_{2j} \wedge \neg\phi_{2j+1}$.

UMinDiag. To test whether some set $J \subseteq I$ is a diagnosis, we need to check $C \subseteq \mathcal{C}_G^{P,Sp}(I)$; similarly as deciding CONS, the latter problem can be shown to be in $\mathsf{P}_{\|}^{C}$. For $O = \mathsf{Exp}, \mathsf{PSpace}$ it is then clear that an algorithm can cycle through all $J \subseteq I$ and compute two minimal diagnoses in exponential time (resp. polynomial space), provided two exist (one always exists). For other O, the following more involved method works.

Like for CONS, we first compute with a round of parallel O oracle calls the effects $\mathcal{F}_G^P(I)$. Then, in a next round, we ask oracles, for $k = 0, \ldots, |I|$, whether for all $J \subseteq I$ of size $|J| = k$, it holds that $C \not\subseteq \mathcal{C}_G^{P,Sp}(J)$; for the considered $O \supseteq \mathsf{NExp}$, each oracle call is in O (exponentially many J's can be considered without complexity increase), while for the other classes it is in NP^O. The smallest k for which the oracle answers "no" is the size k^* of a *smallest* (in terms of cardinality) diagnosis, and thus of some minimal diagnosis. In a further round we then ask an oracle whether for all $J_1, J_2 \subseteq I$ such that $|J_1| = k^*$ and $J_1 \not\subseteq J_2$ it holds that either $C \not\subseteq \mathcal{C}_G^{P,Sp}(J_1)$ or $C \not\subseteq \mathcal{C}_G^{P,Sp}(J_2)$; the answer will be "yes" iff a unique minimal diagnosis exists. Overall, the method uses three rounds of O (resp., NP^O) oracle calls, which puts the problem in the class $\mathsf{P}_{\|[3]}^{O}$ resp. $\mathsf{P}_{\|[3]}^{\mathsf{NP}^O}$; this class, however, by the lemma coincides with $\mathsf{P}_{\|}^{O}$ resp. $\mathsf{P}_{\|}^{\mathsf{NP}^O}$; and for $O = \mathsf{P}^{\mathsf{NP}}$, we have $\mathsf{NP}^{\mathsf{P}^{\mathsf{NP}}} = \mathsf{NP}^{\mathsf{NP}}$, thus $\mathsf{P}_{\|}^{\mathsf{NP}^{\mathsf{P}^{\mathsf{NP}}}} = \mathsf{P}_{\|}^{\Sigma_2^p}$.

The hardness results for $O \supseteq \mathsf{NExp}$ are obtained by a reduction of the complement of the EVEN problem (which is also $\mathsf{P}_{\|}^{O}$-hard) that is a variant of the reduction for the CONS problem. The conflict rules are extended to $\chi \leftarrow \psi_{2j}, \text{not}\ \psi_{2j+1}, in_1$, $0 \leq j \leq n$, and a further rule $\chi \leftarrow in_0$ is added, where in_0 and in_1 are fresh input facts. Then, $J = \{in_0\}$ is a minimal diagnosis, and it is the single one iff $J = \{in_1\}$ is not a diagnosis, which is the case iff the EVEN instance is a no-instance. For the remaining cases of O, one can show hardness for co-NP^O by a reduction from evaluation of suitable QBFs; however, hardness for NP^O, let alone for $\mathsf{P}_{\|}^{\mathsf{NP}^O}$ is not apparent.

Corr. It is easy to see that CORR is solvable in polynomial time with parallel oracle queries "$f(t, w, v) \in \mathcal{F}_G^P(I)$?" for $P = P^S, P^M$, which have complexity O of problem IMPL for the underlying logic; this shows membership in $\mathsf{P}_{\|}^{O}$. The hardness results for $O \neq \mathsf{PSpace}, \mathsf{Exp}$ are shown similarly as for problem CONS by a reduction from the EVEN problem. We use effect mappings P^M and P^S, which consist of P from there with additional rules (resp. implications for FOL+DCA) as follows: P^M contains $\phi'_j \leftarrow \phi_0, \ldots, \phi_{2j-1}$ and P^S contains $\phi'_j \leftarrow \phi_0, \ldots, \phi_{2j}$, for $0 \leq j \leq n$, where ϕ'_j is a new effect atom. Assuming that I_{2n+1} is a no-instance, the specifications will correspond, i.e., $\mathcal{F}_G^{P^M}(I) = \mathcal{F}_G^{P^S}(I)$, iff the number of yes-instances among $I_0, \ldots, I_{2n+1}$ is even.

Repair. To solve REPAIR, we can guess some change $I^+, I^- \subseteq I_G$ and then check whether $\mathcal{A}(I^+, I^-)$ holds and there are no conflicts for input $I' = (I \setminus I^-) \cup I^+$. For IMPL complexity $O = \mathsf{Exp}, \mathsf{PSpace}$, this stays within O, as one can cycle through all I', and for the other complexity classes O, by the results for CONS, the problem is in $\mathsf{NP}^{\mathsf{P}_{\|}^{O}} = \mathsf{NP}^O$. The hardness results for these O can be obtained by reductions from a reasoning problem for ASP$^{\neg}$ resp. ASP$^{\vee,\neg}$ programs: given a set of facts F, a program P and a fact α, does there exist some $F' \subseteq F$ such that $P \cup F' \models \alpha$ (where $\models$ is cautious consequence); this problem is, as shown with slight extensions of the respective proofs for cautious consequence [4, 5], complete for NP^O. Finally, the results for REPAIR remain unchanged if only strict repairs are considered as CORR has lower complexity.

6 Discussion and Conclusion

Since comprehensible specifications play a major role in this problem domain, we demand that the implementation should be *declarative*. An imperative way of programming such rules will quickly lead to deeply nested conditionals with intransparent dependencies. Further, we need a high degree of *modularity* to enable changes to specifications independent of the implementation of reasoning tasks. Since the domain comprises many patterns with exceptions and special cases, some sort of *default reasoning* would be desirable; e.g., traffic is permitted in a certain direction, *unless* it is explicitly prohibited.

Answer Set Programming (ASP) [9] is a natural choice for writing such declarative, executable specifications. Due to the availability of efficient solvers such as DLV [11] and Potassco [8], the ASP paradigm gains increasing popularity [2]. Furthermore, thanks to modularity properties of answer set semantics, it is easy to modularly compose programs P and Sp from a traffic regulation $\Pi = (P, Sp)$ into a single program $P \cup Sp$, provided that P has a unique answer set (as given, e.g., with stratified programs), where rules $\neg x \leftarrow \mathrm{not}\ x$ for $\overline{X}$ are included (resp., if recursion through negation would emerge, rules $\neg x' \leftarrow \mathrm{not}\ x$ and $x' \leftarrow x$, and x is replaced in Sp by x').

Implementation. In cooperation with domain experts, we have written prototypical ASP-programs to evaluate the consistency of scenarios, to diagnose conflicts and to test correspondence using partial realizations of the traffic regulation. In addition, we also dealt with repairs. We have used both the DLV system and Potassco to run these programs on a number of different scenarios, and obtained satisfactory initial results. Further development, regarding a representation of the traffic regulations and engineering the program for scalable execution, is ongoing; the fact that ASP programs serve as executable specification is very helpful for developing the representation.

The use of DLV and the flexible optimization constructs available in its language by weak constraints, has in fact also led us to experiment with preferred notions of diagnoses and repairs. The possibilities range from generic preferences, like favoring deletions over additions, to the encoding of very specific domain knowledge.

Example 10. Deleting the mandatory left turn at d_4 in Fig. 1b gives an alternative repair possibility for Example 2. The options to continue to drive straight over the junctions towards node d_7, as well as turning right towards node d_5, are not available due to the no-entry signs. According to the depicted street model, there is a way out from node d_4 via the U-turn to node d_3, from which an out-node node is reachable via nodes c_8 and c_3. Arguably, it is reasonable to still classify the situation as loop, since paths should not use U-turns. A road user arriving at d_4 for the first time would not know that she will eventually come back to node d_4 (by taking the supposed path).

Thus, one might have different categories of loop conflicts like “strong loop” and “weak loop,” where the latter allows for escapes via U-turns. In this sense, Fig. 1b represents a strong loop, and the repair where the sign at d_4 is removed represents a weak loop. The repair might be less preferred. If yet chosen, we might wish to warn road users who eventually will have to take a U-turn, through a no-through-road sign. However, the ideal position to do so is not obvious. Alternatively, we could add mandatory U-turn signs before junctions one has to eventually return to. While this might often be the desired solution, it is formally not optimal, since it is more restrictive than necessary.

Summary and Outlook. 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 future dynamic regulation settings. We formalized the notion of traffic regulation problem and defined major reasoning tasks on it, whose computational complexity we characterized for different logic languages. Complexity results on abduction from logic programs [6] are not applicable, as the language setting and problems studied there (in view of Prop. 1, consistent diagnosis existence with no effect mapping) are different. Our results provide a basis for implementation and show that some reasoning tasks may be hosted in the underlying logic language, while for others one needs a more expressive one. Finally, we briefly addressed an ASP-based prototype.

The implementation is part of an ongoing industrial project on management of traffic regulation data with *PRISMA solutions GmbH*.[3] Currently, real world traffic regulation data is administrated by software that shall be enhanced by the methods we described, and the resulting application shall be used by several Austrian regions, starting with Lower Austria and Vienna. On the theoretical side, an investigation of preference and properties of traffic regulation problems, under possible restrictions of the constituents, remains to be done. Also other logic formalisms, e.g., defeasible logic [1] (which like $\mathrm{ASP}^{\neg_s}$ allows for efficient reasoning [13]), or HEX-programs [7] may be considered.

References

1. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Logic 2(2), 255–287 (2001)
2. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (2011)
3. Console, L., Torasso, P.: Automated diagnosis. Intelligenza Artificiale 3(1-2), 42–48 (2006)
4. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and Expressive Power of Logic Programming. ACM Comput. Surv. 33(3), 374–425 (2001)
5. Eiter, T., Faber, W., Fink, M., Woltran, S.: Complexity Results for Answer Set Programming with Bounded Predicate Arities. Ann. Math. Artif. Intell. 51(2-4), 123–165 (2007)
6. Eiter, T., Gottlob, G., Leone, N.: Abduction From Logic Programs: Semantics and Complexity. Theoretical Computer Science 189(1-2), 129–177 (1997)
7. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer set programming. In: IJCAI, pp. 90–96. (2005)
8. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.T.: Potassco: The Potsdam answer set solving collection. AI Commun. 24(2), 107–124 (2011)
9. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. Next Generat. Comput. 9(3–4), 365–386 (1991)
10. de Kleer, J., Kurien, J.: Fundamentals of model-based diagnosis. In: IFAC Symposium SAFEPROCESS 2003. pp. 25–36. Elsevier (2003)
11. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM TOCL 7(3), 499–562 (2006)
12. Lucas, P.: Symbolic diagnosis and its formalisation. Knowl. Eng. Rev. 12, 109–146 (1997)
13. Maher, M.J.: Propositional defeasible logic has linear complexity. Theory Pract. Log. Program. 1(6), 691–711 (2001)
14. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
15. Poole, D.: Normality and faults in logic-based diagnosis. In: IJCAI. pp. 1304–1310 (1989)
16. Poole, D.: Representing diagnosis knowledge. Ann. Math. Artif. Intell. 11, 33–50 (1994)

[3] http://www.prisma-solutions.at