

Efficient Explanations for Inconsistent Constraint Sets

Alexander Felfernig¹ and Monika Schubert¹ and
Monika Mandl¹ and Gerhard Friedrich² and Erich Teppan²

Abstract. Constraint sets can become inconsistent in different contexts. We are interested in identifying minimal sets of constraints that have to be adapted or deleted in order to restore consistency. In this paper we sketch a highly efficient divide-and-conquer based diagnosis approach which identifies minimal sets of faulty constraints in a given over-constrained problem. This approach is specifically applicable in scenarios where the *efficient identification of leading (preferred) diagnoses* is crucial.

1 Introduction

There are many scenarios where constraint sets can become over-constrained. For example, when implementing a configuration knowledge base, constraints can become inconsistent with a given set of test cases [1]. Alternatively, when interacting with a configurator application [2], the given customer requirements can become inconsistent with the configuration knowledge base. In both situations there is a need of an intelligent assistance functionality that actively supports users (end users or knowledge engineers). A widespread approach to support users in the identification of minimal sets of faulty constraints is to combine conflict detection (see, e.g., [3]) with a corresponding hitting set algorithm [4].

In their original form these algorithms are applied for the calculation of *minimal cardinality diagnoses* which are typically determined with breadth-first search. Further diagnosis algorithms have been developed that follow a best-first search regime where the expansion of the hitting set search tree is guided by failure probabilities of components. Another example for such an approach is presented in [2] where similarity metrics are used to guide the (best-first) search for a preferred (plausible) minimal diagnosis (including repairs). Both, simple breadth-first search and best-first search diagnosis approaches are predominantly relying on the calculation of conflict sets [3].

The determination of a minimal diagnosis of cardinality n requires the identification of at least n minimal conflict sets. In this paper we introduce a diagnosis approach that allows to determine *one minimal diagnosis at a time with the same computational effort related to the calculation of one conflict set at a time*. The approach supports the identification of preferred diagnoses given predefined preferences regarding a set of decision alternatives. It is not restricted to constraint-based systems but it is also applicable, for example, in the context of SAT solving and reasoning with description logics.

2 Example Domain

Since we exploit configuration problems for the discussion of our diagnosis approach, we first introduce a formal definition of a con-

figuration task. This definition is based on the definition of [1] but is given in the context of a basic constraint satisfaction problem (CSP).

Definition 1 (Configuration Task). A configuration task can be defined as a CSP (V, D, C) . $V = \{v_1, v_2, \dots, v_n\}$ represents a set of finite domain variables. $D = \{\text{dom}(v_1), \text{dom}(v_2), \dots, \text{dom}(v_n)\}$ represents a set of variable domains $\text{dom}(v_i)$ where $\text{dom}(v_i)$ represents the domain of the variable v_i . $C = C_{KB} \cup C_R$ where $C_{KB} = \{c_1, c_2, \dots, c_q\}$ is a set of domain specific constraints (the configuration knowledge base) that restrict the possible combinations of values assigned to the variables in V . $C_R = \{c_{q+1}, c_{q+2}, \dots, c_t\}$ is a set of customer requirements also represented as constraints.

A simplified example of a configuration task in the automotive domain is the following. In this example, *type* represents the car type, *pdcc* is the parc distance control functionality, *fuel* represents the fuel consumption per 100 kilometers, a *skibag* allows the ski stowage inside the car, and *4-wheel* represents the corresponding actuation type. These variables describe the potential set of requirements that can be specified by the user. The possible combinations of these requirements are defined by a set of constraints which are denoted as *configuration knowledge base*, $C_{KB} = \{c_1, c_2, c_3, c_4\}$. Furthermore, we assume the set of *customer requirements* $C_R = \{c_5, c_6, c_7\}$.

- $V = \{\text{type}, \text{pdcc}, \text{fuel}, \text{skibag}, \text{4-wheel}\}$
- $D = \{\text{dom}(\text{type}) = \{\text{city}, \text{limo}, \text{combi}, \text{xdrive}\}, \text{dom}(\text{pdcc}) = \{\text{yes}, \text{no}\}, \text{dom}(\text{fuel}) = \{4l, 6l, 10l\}, \text{dom}(\text{skibag}) = \{\text{yes}, \text{no}\}, \text{dom}(\text{4-wheel}) = \{\text{yes}, \text{no}\}\}$
- $C_{KB} = \{c_1: \text{4-wheel} = \text{yes} \Rightarrow \text{type} = \text{xdrive}, c_2: \text{skibag} = \text{yes} \Rightarrow \text{type} \neq \text{city}, c_3: \text{fuel} = 4l \Rightarrow \text{type} = \text{city}, c_4: \text{fuel} = 6l \Rightarrow \text{type} \neq \text{xdrive}\}$
- $C_R = \{c_5: \text{type} = \text{combi}, c_6: \text{fuel} = 4l, c_7: \text{4-wheel} = \text{yes}\}$

On the basis of this configuration task definition, we can now introduce the definition of a concrete *configuration*.

Definition 2 (Configuration). A configuration for a given configuration task (V, D, C) is an instantiation $I = \{v_1=\text{ins}_1, v_2=\text{ins}_2, \dots, v_n=\text{ins}_n\}$ where $\text{ins}_k \in \text{dom}(v_k)$.

A configuration is *consistent* if the assignments in I are consistent with the $c_i \in C$. Furthermore, a configuration is *complete* if all variables in V are instantiated. Finally, a configuration is *valid* if it is consistent and complete.

3 Diagnosing Over-Constrained Problems

For the configuration task introduced in Section 2 we are *not* able to find a solution, for example, a *combi*-type car does not support a fuel consumption of *4l per 100 kilometers*. Consequently, we want to identify minimal sets of constraints ($c_i \in C_R$) which have to be deleted (adapted) in order to be able to identify a solution (restore the consistency). Since both, the notion of a (*minimal*) *conflict* and the notion of a (*minimal*) *diagnosis* will be used in the following, we provide the corresponding definitions here.

¹ TU Graz, Austria, email: {felfernig, schubert, mandl}@ist.tugraz.at

² University of Klagenfurt, Austria, email: {friedrich, teppan}@uni-klu.ac.at

Definition 3 (Conflict Set). A conflict set is a set $CS \subseteq C_R$ s.t. $C_{KB} \cup CS$ is inconsistent. A conflict set CS is a *minimal* if there does not exist a conflict set CS' with $CS' \subset CS$.

In our working example we can identify three minimal conflict sets which are $CS_1=\{c_5, c_6\}$, $CS_2=\{c_5, c_7\}$, and $CS_3=\{c_6, c_7\}$.

Definition 4 (CR Diagnosis Problem). A customer requirements diagnosis (CR diagnosis) problem is defined as a tuple (C_{KB}, C_R) where C_R is the set of given customer requirements and C_{KB} represents the constraints part of the configuration knowledge base.

The definition of a *CR diagnosis* that corresponds to a given CR Diagnosis Problem is the following (see Definition 5).

Definition 5 (CR Diagnosis). A CR diagnosis for a CR diagnosis problem (C_{KB}, C_R) is a set $\Delta \subseteq C_R$, s.t., $C_{KB} \cup (C_R - \Delta)$ is consistent. Δ is *minimal* if there does not exist a diagnosis $\Delta' \subset \Delta$ s.t. $C_{KB} \cup (C_R - \Delta')$ is consistent.

The HSDAG algorithm for determining minimal diagnoses is discussed in detail in [4] (the calculation of diagnoses for our working example is shown in Figure 1).

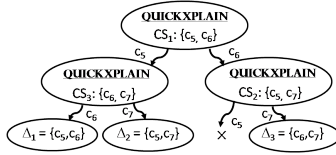


Figure 1. HSDAG (Hitting Set Directed Acyclic Graph) [4] for the CR diagnosis problem ($C_R=\{c_5, c_6, c_7\}$, $C_{KB}=\{c_1, c_2, c_3, c_4\}$). The sets $\{c_5, c_6\}$, $\{c_6, c_7\}$, and $\{c_5, c_7\}$ are the minimal diagnoses – the conflict sets CS_1 , CS_2 , and CS_3 are determined on the basis of QUICKXPLAIN [3].

4 Calculating Preferred Diagnoses

Preferred Diagnoses. Users prefer to keep the important requirements and to change or delete the less important ones [3]. For the characterization of a preferred diagnosis we will rely on the definition of a total ordering of a set of constraints in C (respectively C_R). The following definition of a *lexicographical ordering* (Definition 6) is based on the definition of total orderings for constraints that has been applied in [3] for the determination of *preferred conflict sets*.

Definition 6 (Total Lexicographical Ordering). Given a total order $<$ on C , we enumerate the constraints in C in increasing $<$ order $c_1, \dots, c_n$ starting with the least important ones (i.e., $c_i < c_j \Rightarrow i < j$). We compare two subsets X and Y of C lexicographically: $X <_{lex} Y$ iff $\exists k: c_k \in Y - X$ and $X \cap \{c_{k+1}, \dots, c_t\} = Y \cap \{c_{k+1}, \dots, c_t\}$.

Based on this definition of a lexicographical ordering, we can now introduce the definition of a *preferred diagnosis*.

Definition 7 (Preferred Diagnosis). A minimal diagnosis Δ for a given CR requirements diagnosis problem (C_R, C_{KB}) is a preferred diagnosis for (C_R, C_{KB}) iff there does not exist another minimal diagnosis Δ' with $\Delta' <_{lex} \Delta$.

In our working example we assumed the ordering ($c_5 < c_6 < c_7$), i.e., the most important requirement is c_7 (the 4-wheel functionality). If we change the ordering to ($c_7 < c_6 < c_5$), our approach would then determine $\{c_6, c_7\}$ as the preferred minimal diagnosis (see the following paragraph).

Diagnosis Approach. For the following discussions we introduce the set $AC = C_{KB} \cup C_R$ which represents the union of customer requirements (C_R) and the configuration knowledge base (C_{KB}). The basic idea of our approach is the following. In our working example, the set of customer requirements $C_R = \{c_5, c_6, c_7\}$ includes at least one minimal diagnosis since C_{KB} is consistent and $C_{KB} \cup C_R$ is

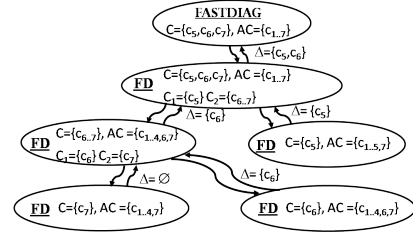


Figure 2. FASTDIAG execution trace for the CR diagnosis problem ($C_R=\{c_5, c_6, c_7\}$, $C_{KB}=\{c_1, c_2, c_3, c_4\}$).

inconsistent. In the worst case C_R itself represents the minimal diagnosis which would mean that *all* constraints in C_R are part of the diagnosis, i.e., each $c_i \in C_R$ represents a singleton conflict. In our case C_R obviously does not represent a minimal diagnosis – the set of diagnoses in our working example is $\{\Delta_1 = \{c_5, c_6\}, \Delta_2 = \{c_5, c_7\}, \Delta_3 = \{c_6, c_7\}\}$ (see Section 3). The next step is to divide the set of customer requirements $C_R = \{c_5, c_6, c_7\}$ into the two sets $C_1 = \{c_5\}$ and $C_2 = \{c_6, c_7\}$ and to check whether $AC - C_1$ is already consistent. If this is the case, we can omit the set C_2 since at least one minimal diagnosis can already be identified in C_1 . In our case, $AC - \{c_5\}$ is inconsistent, which means that we have to consider further elements from C_2 . Therefore, $C_2 = \{c_6, c_7\}$ is divided into the sets $\{c_6\}$ and $\{c_7\}$. In the next step we can check whether $AC - (C_1 \cup \{c_6\})$ is consistent – now this is the case which means that we do not have to further take into account $\{c_7\}$ for determining the diagnosis. Since $\{c_5\}$ does not include a diagnosis but $\{c_5\} \cup \{c_6\}$ includes a diagnosis, we can deduce that $\{c_6\}$ must be part of the diagnosis. The final step is to check whether $AC - \{c_6\}$ leads to a diagnosis without including $\{c_5\}$. We see that $AC - \{c_6\}$ is inconsistent, i.e., $\Delta = \{c_5, c_6\}$ is a minimal diagnosis for the CR diagnosis problem ($C_R = \{c_5, c_6, c_7\}$, $C_{KB} = \{c_1, \dots, c_4\}$). A discussion of the underlying algorithm (FASTDIAG) is provided in an extended version of this paper – an execution trace for our example is given in Figure 2.

A detailed performance analysis of our diagnosis algorithm (FASTDIAG) is omitted due to space limitations. The major properties of FASTDIAG are the following:

- **Runtime:** FASTDIAG is able to determine minimal diagnoses with a logarithmic number of consistency checks. In the worst case it needs $2d \cdot \log_2(\frac{n}{d}) + 2d$ consistency checks, where d is the minimal diagnoses set size and n is the number of constraints (in C).
- **Applicability:** the algorithm clearly outperforms existing diagnosis approaches in terms of runtime and prediction quality. Thus it is a new and valuable alternative especially for interactive settings.
- **Minimality:** the algorithm guarantees the calculation of minimal diagnoses only.
- **Completeness:** the algorithm is able to determine all minimal diagnoses that exist for the given diagnosis problem.

References

- [1] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, 'Consistency-based diagnosis of configuration knowledge bases', *AI Journal*, **152**(2), 213–234, (2004).
- [2] A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan, 'Plausible repairs for inconsistent requirements', in *IJCAI'09*, pp. 791–796, Pasadena, CA, (2009).
- [3] U. Junker, 'Quickxplain: Preferred explanations and relaxations for over-constrained problems', in *AAAI'04*, pp. 167–172, San Jose, CA, (2004).
- [4] R. Reiter, 'A theory of diagnosis from first principles', *AI Journal*, **23**(1), 57–95, (1987).