How to Identify Boundary Conditions with Contrasty Metric?

1st Weilin Luo
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, China
luowlin3@mail2.sysu.edu.cn

2nd Hai Wan†
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, China
wanhai@mail.sysu.edu.cn

3rd Xiaotong Song
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, China
songxt5@mail2.sysu.edu.cn

4th Binhao Yang
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, China
yangbh7@mail2.sysu.edu.cn

5th Hongzhen Zhong
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, China
zhonghzh5@mail2.sysu.edu.cn

6th Yin Chen
School of Computer Science
South China Normal University
Guangzhou, China
ychen@scnu.edu.cn

2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) | 978-1-6654-0296-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSE43902.2021.00132

***Abstract*—The *boundary conditions (BCs)* have shown great potential in requirements engineering because a BC captures the particular combination of circumstances, *i.e.*, *divergence*, in which the goals of the requirement cannot be satisfied as a whole. Existing researches have attempted to automatically identify lots of BCs. Unfortunately, a large number of identified BCs make assessing and resolving divergences expensive. Existing methods adopt a coarse-grained metric, *generality*, to filter out less general BCs. However, the results still retain a large number of redundant BCs since a general BC potentially captures *redundant* circumstances that do not lead to a divergence. Furthermore, the *likelihood* of BC can be misled by redundant BCs resulting in costly repeatedly assessing and resolving divergences.**

In this paper, we present a fine-grained metric to filter out the redundant BCs. We first introduce the concept of *contrasty* of BC. Intuitively, if two BCs are contrastive, they capture different divergences. We argue that a set of contrastive BCs should be recommended to engineers, rather than a set of general BCs that potentially only indicates the same divergence. Then we design a post-processing framework (`PPFc`) to produce a set of contrastive BCs after identifying BCs. Experimental results show that the contrasty metric dramatically reduces the number of BCs recommended to engineers. Results also demonstrate that lots of BCs identified by the state-of-the-art method are redundant in most cases. Besides, to improve efficiency, we propose a joint framework (`JFc`) to interleave assessing based on the contrasty metric with identifying BCs. The primary intuition behind `JFc` is that it considers the search bias toward contrastive BCs during identifying BCs, thereby pruning the BCs capturing the same divergence. Experiments confirm the improvements of `JFc` in identifying contrastive BCs.

***Index Terms*—Goal-Oriented Requirement Engineering, Boundary Conditions, Goal-Conflict Identification**

I. Introduction

Goal-oriented requirement engineering (GORE) [31] is an essential phase of the software development life cycle, the important task of which is to attain correct software requirements specifications. Many researches have demonstrated the significant advantages that formal and goal-oriented approaches help generate correct specifications [1], [7], [11]. In such approaches, *domain properties* and *goals* are represented in *linear-time temporal logic (LTL)* because LTL is proved convenient for abstracting specifications of a large class of requirements, assumptions, and domain properties [31].

The identify-assess-control cycle in GORE aims at identifying, assessing, and resolving inconsistency in which the goals of the requirement cannot be satisfied as a whole. The *divergence* is a weak inconsistency, *i.e.*, particular circumstances where the satisfaction of some goals inhibits the satisfaction of others. A divergence is captured by *boundary conditions (BCs)* which explain why the divergence happens. Various approaches [9], [10], [32] have been proposed to automatically identify BCs in the context of GORE.

As the number of identified BCs in the identification stage increases, for example, there are more than 100 BCs in the case named London Ambulance Service in [9], the assess-

†Corresponding author.

This paper was supported by the Guangdong Province Science and Technology Plan projects (No. 2017B010110011), National Natural Science Foundation of China (No. 61976232), National Key R&D Program of China (No. 2018YFC0830600), Guangdong Province Natural Science Foundation (No. 2018A030313086 and 2017A070706010 (soft science)).

978-1-6654-0296-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSE43902.2021.00132

ment stage and the resolution stage become very expensive, and even impractical. In order to provide engineers with an acceptable number of BCs to analyze, the *generality* metric (Definition 2) [9] has been proposed to automatically filter out the less general BCs. The generality metric qualitatively distinguishes the importance of BC using the implication relationship of BCs. Intuitively, a more general (also known as weaker) BC is more important because it potentially covers more circumstances to represent a divergence. Therefore, the less general BCs can be filtered out by the more general one.

Unfortunately, we observe that a set of general BCs still retains a large number of redundant BCs. The reason is that the generality metric can be considered as a coarse-grained metric. A general BC potentially captures *redundant circumstances* that do not lead to a divergence.

Furthermore, the accuracy of the assessment step based on *likelihood* is sensitive to the redundant circumstances, so a set of general BCs can lead to mistakes in the assessment step (an example shown in Section III). The assessment stage is concerned with evaluating how likely the identified conflicts are, and how likely and severe are their consequences. Degiovanni et al. [8] proposed an automatically assessing method based on model counting, which can be used to prioritize BCs to be resolved. However, a set of general BCs misleads to prioritize the BCs because a general BC potentially captures redundant circumstances that do not lead to a divergence.

In this paper, we present a new metric to assess the differences among the divergences captured by BCs. Our approach is novel in the following respects: (1) It is a fine-grained metric because it can filter out not only the less general BCs but also the BCs that capture the same divergence; (2) and it measures the differences between BCs from the different divergences captured by them. We first introduce the concept of *contrasty* of BCs motivated by avoiding boundary conditions [32] in resolving divergences. More precisely, given two BCs ϕ and φ, we consider whether $\phi' = \phi \wedge \neg\varphi$ and $\varphi' = \varphi \wedge \neg\phi$ are BCs. ϕ' (*resp.* φ') represents the circumstances left by removing the circumstances captured by φ (*resp.* ϕ) from that captured by ϕ (*resp.* φ). If neither ϕ' nor φ' is BC, ϕ and φ are contrastive. Intuitively, if two BCs are contrastive, they capture different divergences. We argue that a set of contrastive BCs should be recommended to engineers, rather than a set of general BCs since they potentially only indicate the same divergence.

Based on the contrasty metric, we design a post-processing framework (PPFc) to produce a set of contrastive BCs after identifying BCs. Experimental results show that the contrasty metric can filter out all the BCs that capture the same divergence, which dramatically reduces the number of BCs recommended to engineers. Furthermore, experiments show that the BCs identified by the state-of-the-art method are not contrastive in most cases. In other words, these BCs capture the same divergence, in which engineers only consider one BC to resolve a divergence while others are redundant.

In order to improve efficiency, we propose a joint framework (JFc) to interleave assessing based on the contrasty metric with identifying BCs. Specifically, when a BC is identified during the search, we add its negation as an additional constraint to domain properties. The additional constraint makes the domain properties dynamically change so that it prevents the same circumstances from being identified as a BC again. The insight behind this is that it produces the search bias towards the BCs that capture different divergences. Besides, we propose a sufficient condition for the case where there not exist BCs. It guarantees that if we resolve the divergences captured by the BCs in the set of contrastive BCs, there not exist divergences under the original domain properties and goals. Experiments confirm the improvements of JFc in identifying contrastive BCs.

Our main contributions are summarized as follows.

- We present the novel contrasty metric to evaluate the differences between BCs , which can filter out more redundant BCs that capture the same divergence.
- We design a post-processing framework (PPFc) to produce a set of contrastive BCs. In order to improve efficiency, we also design a joint framework (JFc) to capture different divergences during the search.
- Experiments show that the contrasty metric is better than the generality metric for filtering out redundant BCs.

II. Background

In this section, we introduce the background of goal-conflict analysis and linear-time temporal logic. We briefly recall some basic notions for the rest of the paper.

A. Goal-Conflict Analysis

In GORE [31], goals are prescriptive statements that the system must achieve, and domain properties are descriptive statements that capture the domain of the problem world. In practice, it is unrealistic to require requirements specifications to be complete or all goals to be satisfiable, because inconsistencies may occur. *Goal-conflict analysis* [31], [33] deals with the inconsistencies via the following identify-assess-control cycle:

1) the *identification stage* is to identify a condition whose occurrence makes some inconsistencies;
2) the *assessment stage* is to assess and prioritize the identified inconsistencies according to their likelihood and severity;
3) the *resolution stage* is to resolve the identified inconsistencies by providing appropriate countermeasures.

Goal-Conflict Identification. In this paper, we focus on a weak inconsistency, *i.e.*, *divergence*. A divergence essentially represents a *boundary condition (BC)* whose occurrence results in the loss of satisfaction of the goals, which makes the goal divergence [32].

Definition 1. *Let* $G = \{g_1, \ldots, g_n\}$ *be a set of goals and* Dom *a set of domain properties. A* divergence *occurs within*

Dom iff there exists a boundary condition φ *under Dom and G such that the following conditions hold:*

$$Dom \wedge G \wedge \varphi \models \bot \quad \text{(logical inconsistency)}$$
$$Dom \wedge G_{-i} \wedge \varphi \not\models \bot, \text{ for each } 1 \leq i \leq n \quad \text{(minimality)}$$
$$\neg G \not\equiv \varphi \quad \text{(non-triviality)}$$

where $G = \bigwedge_{1 \leq i \leq n} g_i$ *and* $G_{-i} = \bigwedge_{j \neq i} g_j$.

Intuitively, a BC captures a particular combination of circumstances in which the goals cannot be satisfied as a whole. The logical inconsistency property means the conjunction of goals becomes inconsistent when φ holds. The minimality property states that disregarding any of the goals no longer results in inconsistency. The non-triviality property forbids a BC to be a trivial condition which is the negation of the conjunction of the goals. Note that BCs are not false due to the minimality property.

Specifying software requirements in the LTL formulation allows us to employ automated LTL satisfiability solvers to check for the feasibility of the corresponding requirements. With an efficient LTL satisfiability solver, we can automatically check if the generated candidate formulae are valid BCs or not by checking if they satisfy the properties.

In the identification stage, the *generality* [9] metric has been proposed to reduce the redundant BCs. It is defined as follows.

Definition 2. *Let S be a set of BCs. A BC* $\varphi_i \in S$ *is* more general *than another BC* $\varphi_j \in S$ *if* φ_j *implies* φ_i.

Intuitively, a more general BC φ captures all the particular combinations of circumstances captured by the less general BCs than φ. Therefore, it is important to provide engineers with more general BCs. As far as we know, the generality metric is the only metric to filter out BCs.

Goal-Conflict Assessment. In the assessment stage, in order to give engineers more guidance on which BCs need to get attention, probabilities [3] of their occurrence are considered as an important indicator. For systems without extra probabilistic information, there is an approach [8] based on model counting to analyze the likelihood of BCs. It is defined as follows.

Definition 3. *Let* ϕ *be a BC, Dom domain properties, and k a positive integer. The likelihood of* ϕ *is* $L(\phi) = \frac{\#(Dom \cup \phi, k)}{\#(Dom, k)}$ *where* $\#(C, k)$ *denotes that the total number of models bases of length k satisfying constraints in C.*

Intuitively, the larger likelihood of a BC indicates that the divergence captured by the BC is more likely to happen.

Goal-Conflict Resolution. In the resolution stage, as the BCs malfunction the system when the system reaches the circumstances captured by BCs, the engineers need some strategies to resolve the divergences captured by the BCs.

Definition 4. *Let Dom be domain properties, G goals, and* ϕ *a BC under Dom and G.* Resolving divergences *aims to modify Dom and G to get Dom′ and G′, so that* ϕ *under Dom′ and G′ does not fulfill at least one of the following constraints:*

1) $Dom' \wedge G' \wedge \varphi \models \bot$;
2) $Dom' \wedge G'_{-i} \wedge \varphi \not\models \bot$, *for each* $1 \leq i \leq n$;
3) $\neg G' \not\equiv \varphi$.

Intuitively, after resolving divergences, the circumstances captured by the BC do not happen under the new system expressed by updated domain properties and goals. Van Lamsweerde et al. [32] proposed that generating reasonably updated domain properties and goals is an open problem because it requires a lot of experience. We will illustrate an example of resolving divergences in VIII. Therefore, a large number of identified BCs make the resolution stage very expensive.

A straightforward strategy can be adopted to avoid the circumstances captured by a BC. The *avoid pattern* [32] is therefore introduced: $\Box(Dom \rightarrow \Box\neg B)$ where B denotes a BC to be inhibited.

B. Linear-Time Temporal Logic

Linear-Time Temporal Logic (LTL) [29] is widely used to describe infinite behaviors of discrete systems, which is suitable for specifying software requirements [32]. Throughout this paper, we use lower case letters (*e.g.*, p, h) to denote propositions. The syntax of LTL for a finite set of propositions $\mathbb{P}$ includes the standard logical connectives ($\wedge$, $\vee$, $\neg$), $\mathbb{B} = \{\bot, \top\}$, and temporal operators *next* ($\bigcirc$), *until* ($\mathcal{U}$).

$$\varphi := \bot \mid \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

Operator release ($\mathcal{R}$), eventually ($\Diamond$), always ($\Box$), and weak-until ($\mathcal{W}$) are commonly used, and can be defined as $\varphi_1 \mathcal{R} \varphi_2 := \neg(\neg\varphi_1 \mathcal{U} \neg\varphi_2)$, $\Diamond\varphi := \top \mathcal{U} \varphi$, $\Box\varphi := \neg(\top \mathcal{U} \neg\varphi)$, and $\varphi_1 \mathcal{W} \varphi_2 := \varphi_1 \mathcal{U} (\varphi_2 \vee \Box\varphi_1)$, respectively. We use $|\varphi|$ to denote the *size* of the formula φ, *i.e.*, the number of temporal operators, logical connectives, and literals in φ.

LTL formulae are interpreted over a *linear-time structure*. A linear-time structure is a pair of $W = (S, \varepsilon)$ where S is a state sequence and $\varepsilon : S \rightarrow 2^{\mathbb{P}}$ is a function mapping each state s_i to a set of propositions. Let W be a linear-time structure, $i \geq 1$ a position, and φ_1, φ_2 two LTL formulae. The *satisfaction relation* $\models$ is defined as follows:

$W, i \models p$	iff	$p \in \varepsilon(s_i)$, where $p \in \mathbb{P}$
$W, i \models \neg\phi$	iff	$W, i \not\models \phi$
$W, i \models \varphi_1 \wedge \varphi_2$	iff	$W, i \models \varphi_1$ and $W, i \models \varphi_2$
$W, i \models \bigcirc\varphi$	iff	$W, i+1 \models \varphi$
$W, i \models \varphi_1 \ \mathcal{U} \ \varphi_2$	iff	$\exists k \geq i$ s.t. $W, k \models \varphi_2$ and $\forall i \leq j < k, W, j \models \varphi_1$

An LTL formula φ is called *satisfiable* if and only if there is a linear-time structure (model) satisfying φ. An LTL formula φ *implies* an LTL formula φ', noted $\varphi \rightarrow \varphi'$, if the models of φ are also models of φ'. The LTL satisfiability problem is to check whether an LTL formula is satisfiable, which is PSPACE-complete [30]. Recently, LTL satisfiability checkers based on different techniques have been developed. Among these checkers, nuXmv [6] and Aalta [22] have achieved better performance.

III. Motivating Example

In this section, we will illustrate the drawbacks of the generality metric through an example and discuss the insights behind the contrasty metric. Below we illustrate an example, MinePump [20].

Example 1. *Consider a system to control a pump inside a mine. The main goal of the system is avoiding flood in the mine. The system has two sensors. One detects the high water level (h), the other detects methane in the environment (m). When the water level is high, the system should turn on the pump (p). When there is methane in the environment, the pump should be turned off. Domain property (Dom) and goals (G) are represented via the following LTL formulae.*

Domain Property:

1) ***Name**: PumpEffect (d_1)*
   ***Description**: The pump is turned on for two time steps, then in the following one the water level is not high.*
   ***Formula**: $\square((p \wedge \bigcirc p) \rightarrow \bigcirc(\bigcirc \neg h))$*

Goals:

1) ***Name**: NoFlooding (g_1)*
   ***Description**: When the water level is high, the system should turn on the pump.*
   ***Formula**: $\square(h \rightarrow \bigcirc(p))$*
2) ***Name**: NoExplosion (g_2)*
   ***Description**: When there is methane in the environment, the pump should be turned off.*
   ***Formula**: $\square(m \rightarrow \bigcirc(\neg p))$*

Although the specification is consistent, *i.e.*, all domain properties and goals can simultaneously be satisfied, this specification exhibits some goal divergences. One of the BCs is $\varphi_1 = \Diamond(h \wedge m)$, which captures the circumstances where the high water level and the methane occur at the same time. Under this situation, two goals are unsatisfiable simultaneously within domain property.

We also consider other two BCs $\varphi_2 = h \wedge m$ and $\varphi_3 = \Diamond(h \wedge \neg m \wedge p \wedge \bigcirc(\neg h \wedge \neg p \vee h \wedge (m \vee \neg p)))$. φ_2 captures the circumstances that the water level is high and the methane occurs at the beginning. Through equivalent transformation, we can obtain $\varphi_3 = \Diamond((h \wedge \neg m \wedge p) \wedge \bigcirc((\neg h \wedge m \wedge \neg p) \vee (h \wedge m \wedge p) \vee (\neg h \wedge \neg m \wedge \neg p) \vee (h \wedge m \wedge \neg p) \vee (h \wedge \neg m \wedge \neg p)))$. Clearly, φ_3 captures five circumstances, where, in the future, the system will migrate from the state where the high water level occurs, the methane does not occur, and the pump is turned on ($h \wedge \neg m \wedge p$) to the state described as follows.

1) the high water level does not occur, the methane occurs, and the pump is not turned on ($\neg h \wedge m \wedge \neg p$);
2) the high water level and the methane occur and the pump is turned on ($h \wedge m \wedge p$);
3) the high water level and the methane do not occur and the pump is not turned on ($\neg h \wedge \neg m \wedge \neg p$);
4) the high water level and the methane occur and the pump is not turned on ($h \wedge m \wedge \neg p$);
5) the high water level occurs, the methane does not occur, and the pump is not turned on ($h \wedge \neg m \wedge \neg p$).

Existing methods can search for a large number of BCs. It makes the assessment and resolution stages very expensive, and even impractical. In order to provide engineers with an acceptable number of BCs to be analyzed, it is necessary to proposed metric to filter out the redundant BCs.

If we apply the generality metric, we filter out φ_2 because φ_1 is more general than φ_2. However, the generality metric cannot evaluate φ_1 and φ_3 since the generality relationship between them does not hold. In the assessment stage, if we compute the likelihood based on the method [8], we can classify φ_3 as being more likely than φ_1 in the long term, and engineers should prioritize φ_3 in the search for mechanisms that would allow us to reduce the chances of reaching φ_3.

Unfortunately, the assessment method [8] lacks the accuracy to compute the likelihood by model counting because some models are meaningless, *i.e.*, there does not exist the circumstances to lead the divergence in reality. Considering the circumstances captured by φ_3, we observe that the circumstances (1), (3), (4), and (5) violate g_1. Therefore, they cannot satisfy the minimality of BC, which means that they cannot capture the divergence in reality. These circumstances are *redundant*, so $\varphi_3' = \Diamond((h \wedge \neg m \wedge p) \wedge \bigcirc(h \wedge p \wedge m))$ stands for the circumstances captured by φ_3. We find that φ_1 is more likely than φ_3' using the assessment method [8], so φ_1 should be prioritized. Situations like this show that the accuracy of the assessment method based on likelihood is sensitive to redundant circumstances.

In addition, we find that φ_1, φ_2, and φ_3 capture the same divergence, in which the high water level and the methane occur at the same time, *i.e.*, the circumstance captured by φ_1. It is very useful to identify the BC like φ_1 in resolving divergences. Engineers only resolve φ_1 instead of resolving φ_3 first and then φ_1. It avoids wasting computing resources caused by assessing and resolving redundant BCs.

In this paper, motivated by avoiding boundary conditions [32] in resolving divergence, we introduce the concept of witness (Definition 5) and contrasty (Definition 6) of BCs. Intuitively, the witness of a BC indicates the cause of divergence. If the two BCs are not mutual witnesses, then the two BCs are contrastive, *i.e.*, they capture different divergences. In this case, φ_1 and φ_3 are not contrastive because φ_1 is a witness of φ_3, but not vice versa, which means that the divergences captured by φ_1 are wider than that captured by φ_3. Therefore, we recommend φ_1 to engineers and filter out φ_3.

IV. Identifying Boundary Conditions with Contrasty Metric

In this section, we first introduce the concept of contrasty of BCs. Then, we design a post-processing framework to identify a set of contrastive BCs.

A. Contrasty

We first introduce the concepts of *witness* and *contrasty*.

Definition 5. *Let f be an LTL formula and φ a BC. f is a* witness *of φ iff $\varphi \wedge \neg f$ is not a BC.*

In the definition, motivated by avoiding boundary conditions [32] in resolving divergences, we use a negative LTL formula to avoid some circumstances, *i.e.*, resolving the divergence. Therefore, the witness f of a BC φ indicates why φ is a BC. If f is a BC, it means that the divergence captured by φ is also captured by f.

Definition 6. *Let ϕ and φ be BCs. ϕ and φ are* contrastive, *iff ϕ is not a witness of φ and φ is not a witness of ϕ.*

Definition 7. *Let $\mathcal{B}_c$ be a set of BCs. $\mathcal{B}_c$ is contrastive, iff $\forall \phi, \varphi \in \mathcal{B}_c \wedge \phi \neq \varphi$, ϕ and φ is contrastive.*

Intuitively, the contrastive BCs capture different divergences. We use an example to illustrate the definition of witness and contrasty.

Example 2 (Example 1 cont.)**.** *$\varphi_1 = \Diamond(h \wedge m)$, $\varphi_2 = h \wedge m$, and $\varphi_3 = \Diamond(h \wedge \neg m \wedge p \wedge \bigcirc(\neg h \wedge \neg p \vee h \wedge (m \vee \neg p)))$. Because $\varphi_1 \wedge \neg\varphi_3$ is also a BC, e.g., it captures the circumstances where the high water level and the methane occur at the beginning, φ_3 is not a witness of φ_1. φ_1 is a witness of φ_3 since $\varphi_3 \wedge \neg\varphi_1$ does not satisfy the minimality constraint of BC, i.e., $d_1 \wedge g_1 \wedge (\varphi_3 \wedge \neg\varphi_1)$ is unsatisfiable. Therefore, φ_1 and φ_3 are not contrastive. φ_1 is a witness of φ_2 and φ_2 is not a witness of φ_1, so φ_1 and φ_2 are not contrastive. Intuitively, φ_1 is more important than φ_2 since the divergence captured by φ_1 is wider than that captured by φ_2 (φ_2 is a special case of φ_1). φ_2 and φ_3 are contrastive since they express the occurrence of the high water level and the methane in different situations.*

Based on the definition, we have the following theorems. These theorems indicate the highlight of the contrasty metric.

Theorem 1. *Let ϕ and φ be BCs. If $\phi \rightarrow \varphi$, then φ is a witness of ϕ.*

It is straightforward to prove Theorem 1 because $\phi \wedge \neg\varphi$ is unsatisfiable. Because of Theorem 1, we have Theorem 2.

Theorem 2. *Let $\mathcal{B}_c$ be a set of contrastive BCs. $\forall \phi, \varphi \in \mathcal{B}_c \wedge \phi \neq \varphi$, $\phi \not\rightarrow \varphi \wedge \varphi \not\rightarrow \phi$.*

Theorem 2 shows that there is not a general relation between any two BCs in a contrastive BC set, while there can be a witness relation between some two BCs in a general BC set. According to Theorem 2, the contrasty metric can be regarded as a finer-grained metric than the generality metric because contrasty metric can filter out more redundant BCs than the generality metric. Let us recall Example 1. $\{\varphi_1, \varphi_3\}$ is general, but not contrastive. If the contrasty metric is considered, then $\{\varphi_1\}$ is a contrastive.

Property 1. *Let ϕ and φ be BCs. If ϕ is a witness of φ and φ is not a witness of ϕ, then resolving the divergence captured by ϕ leads to resolving the divergence captured by φ.*

Property 1 shows that it is reasonable that engineers prioritize ϕ to resolve since the circumstances captured by ϕ include the circumstances captured by φ.

Theorem 3. *Let ϕ and φ be two BCs. If ϕ and φ are contrastive, then ϕ and φ capture different divergences.*

Sketch of proof. ϕ and φ are contrastive, so ϕ (*resp.* φ) is not the witness of φ (*resp.* ϕ), which means that $\phi \wedge \neg\varphi$ (*resp.* $\varphi \wedge \neg\phi$) is still a BC. The primary intuition behind $\phi \wedge \neg\varphi$ (*resp.* $\varphi \wedge \neg\phi$) is that after resolving the divergences captured by φ (*resp.* ϕ), there are still divergences captured by ϕ (*resp.* φ). Therefore, ϕ and φ capture different divergences. □

Theorem 3 shows that contrastive BCs capture different divergences. Therefore, it is meaningful to recommend a set of contrastive BCs to engineers.

According to the above analysis, we argue that a set of contrastive BCs should be recommended to engineers, rather than a set of general BCs since they potentially only indicate the same divergences. In Section V, we will discuss the different divergences captured by contrastive BCs and report the advantage of the contrasty metric.

B. Post-Processing Framework

We design a post-processing framework for filtering the BCs based on the contrasty metric (PPFc). It takes a set of BCs ($\mathcal{B}$) identified by a BC solver as inputs. Its output is a set of contrastive BCs ($\mathcal{B}_c$).

Algorithm 1: PPFc

Input: a set of BCs $\mathcal{B}$.
Output: a set of contrastive BCs $\mathcal{B}_c$.

```
1 B_c ← B;
2 for each BC φ ∈ B_c do
3 |   B'_c ← B_c/φ;
4 |   isContrastive, W ←
  |     ExternalContrastyFilter(φ, B'_c);
5 |   if isContrastive then
6 |   |   B_c ← B_c/W;
7 |   else
8 |   |   B_c ← B_c/φ;
9 return B_c;
```

The pseudo code is outlined in Algorithm 1. At each iteration, we choose a BC $\phi \in \mathcal{B}_c$ (Alg. 1 of line 2), then discuss its relationship with other BCs φ in $\mathcal{B}_c$ (Alg. 1 of line 4). If ϕ and φ are witnesses of each other (Alg. 2 of line 3), which means that ϕ and φ capture the same divergences, we select the one with smaller size[1] to stay in $\mathcal{B}_c$. If ϕ is a witness of φ and φ is not a witness of ϕ (Alg. 2 of line 8), which means that the divergences captured by ϕ is wider than that captured by φ, we retain ϕ; otherwise (Alg. 2 of line 10), we remove ϕ. If ϕ and φ are not witnesses of each other, we do not delete any one because they are contrastive.

Theorem 4. *When Algorithm 1 terminates, $\mathcal{B}_c$ is contrastive.*

[1]The BC with smaller size is more compact, and easier to interpret.

Algorithm 2: EXTERNALCONTRASTYFILTER

Input: a BC ϕ and a set of BCs $\mathcal{B}$.
Output: whether ϕ is contrastive in $\mathcal{B}$ and a set of BCs W filtered by ϕ.

```
W ← ∅;
for each BC φ ∈ B do
    if φ is a witness of φ and φ is a witness of φ then
        if the size of φ is larger then that of φ then
            return False, ∅;
        else
            W ← W ∪ {φ};
    else if φ is a witness of φ and φ is not a witness of φ then
        W ← W ∪ {φ};
    else if φ is not a witness of φ and φ is a witness of φ then
        return False, ∅;
return True, W;
```

It is straightforward to prove Theorem 4. Theorem 4 guarantees that Algorithm 1 returns a set of contrastive BCs. We illustrate our method through a running example as follows.

Example 3 (Example 1 cont.)**.** *Assume that the BC solver returns the set of BC* $\mathcal{B} = \{\varphi_1, \varphi_2, \varphi_3\}$*, where* $\varphi_1 = \Diamond(h \wedge m)$*,* $\varphi_2 = h \wedge m$*, and* $\varphi_3 = \Diamond(h \wedge \neg m \wedge p \wedge \bigcirc(\neg h \wedge \neg p \vee h \wedge (m \vee \neg p)))$*.* $\mathcal{B}_c$ *is initialized to* $\{\varphi_1, \varphi_2, \varphi_3\}$*. At the first iteration, assume that* PPFc *chooses* φ_2*.* φ_2 *will be compared with* φ_1 *and* φ_3*. Because* φ_2 *is not a witness of* φ_1 *and* φ_1 *is a witness of* φ_2*,* EXTERNALCONTRASTYFILTER *returns False and an empty set.* $\mathcal{B}_c$ *will be updated as* $\{\varphi_1, \varphi_3\}$*. At the second iteration, assume that* PPFc *chooses* φ_1*.* φ_1 *will be compared with* φ_3*. Because* φ_1 *is a witness of* φ_3 *and* φ_3 *is not a witness of* φ_1*,* EXTERNALCONTRASTYFILTER *returns True and* $\{\varphi_3\}$*.* $\mathcal{B}_c$ *will be updated as* $\{\varphi_1\}$*. Then* PPFc *returns* $\{\varphi_1\}$ *and terminates.*

C. Discussion about completeness and Performance

In this paper, we are not concerned with the completeness of identifying contrastive BCs, *i.e.*, the divergences captured by contrastive BCs cover all the divergences captured by BCs that have been found. The reason is as follows.

Firstly, we focus on filtering out redundant BCs for better resolving divergences which is the fundamental purpose of GORE. In general, the better the identification result is, the easier the resolution stage is. Therefore, we argue that the identified BCs should be conducive to resolving divergences as much as possible rather than completeness.

Furthermore, the completeness of the BC set does not help to resolve divergences. Resolving divergences is a dynamic process. After resolving a BC, some BCs in the original BC set are no longer BCs under the updated domain properties and goals. In this way, for resolving divergences, it is meaningless to get the complete BC set in the BC identification stage.

For example, a set of general BCs fulfills the completeness, but it still retains a large number of redundant BCs that capture the same divergences, so that engineers will do a lot of meaningless work for resolving divergences. In other words, although the generality metric satisfies the completeness, it will also increase the burden of resolving divergences. By comparison, the contrasty metric first considers the optimization of BC resolving divergences.

PPFc only begins to filter out redundant BCs after the BC solver returns a set of BCs. A natural idea is to directly identify contrastive BCs during searching for BCs. In this way, pruning the BCs capturing the same divergence can be performed directly in the search process, thereby speeding up the searching process. Based on this idea, we will discuss a joint framework for identifying contrastive BCs in Section VI.

TABLE I
THE DETAILS OF CASES

Case	#Dom	#Goal	#Var	Size
RetractionPattern1 (RP1)	0	2	2	9
RetractionPattern2 (RP2)	0	2	4	10
Elevator (Ele)	1	1	3	10
TCP	0	2	3	14
AchieveAvoidPattern (AAP)	1	2	4	15
MinePump (MP)	1	2	3	21
ATM	1	2	3	22
Rail Road Crossing System (RRCS)	2	2	5	22
Telephone (Tel)	3	2	4	31
London Ambulance Service (LAS)	0	5	7	32
Prioritized Arbiter (PA)	6	1	6	57
Round Robin Arbiter (RRA)	6	3	4	77
Simple Arbiter (SA)	4	3	6	84
Load Balancer (LB)	3	7	5	85
LiftController (LC)	7	8	6	124
ARM's Advanced Microcontroller Bus Architecture (AMBA)	6	21	16	415

V. EVALUATION OF CONTRASTY

In this section, we reported the advantage of the contrasty metric. Here, we presented the first research question.

RQ 1. *Compared with the generality metric, what are the advantages of the contrasty metric?*

Given a set of BCs $\mathcal{B}$ identified by a BC solver, we applied different metrics to filter out redundant BCs. For the competitor, we combined the generality metric and the likelihood to filter and sort the BCs. Specifically, we first filtered out the less general BCs to produce a set of general BCs $\mathcal{B}_g$ and then sorted them according to the likelihood of BC from high to low. Based on PPFc, we computed a set of contrastive BCs $\mathcal{B}_c$ and sorted them by the likelihood. We analyzed the shortcomings of the generality metric and reported the advantages of the contrasty metric by comparing $\mathcal{B}_g$ and $\mathcal{B}_c$.

TABLE II
THE BCS PRODUCED BY DIFFERENT METRICS

Case	GL		CL		
	Rank	BC	Rank	BC	Witness
RP1	1	$\Diamond(((p \wedge (\Box(\neg q))) \mathcal{U}(\Diamond(q \wedge (\neg p)))) \vee (\Box(p \wedge (\Box(\neg q)))))$	1	$(p \wedge (\Box(\neg q))) \vee (\Diamond(q \wedge (\neg p)))$	1,2,3,4
	2	$\bigcirc((p \wedge (\Box(\neg q))) \vee (\Diamond(q \wedge (\neg p))))$			
	3	$((\neg q \mathcal{U}(q \wedge \neg p)) \mathcal{U}(\Diamond(p \wedge (\Box(\neg q))))) \vee (\Box(\neg q \mathcal{U}(q \wedge \neg p)))$			
	4	$(p \wedge (\Box(\neg q))) \vee (\Diamond(q \wedge \neg p))$			
RP2	1	$\Diamond((p \wedge (\neg q \wedge \neg s)) \vee (q \wedge \neg r))$	1	$\Diamond((q \wedge \neg r) \vee (\bigcirc(p \wedge (\neg s \mathcal{U}(\neg q \wedge \neg s)))))$	1,2,3
	2	$\bigcirc((p \wedge (\neg s \mathcal{U}(\neg q \wedge \neg s))) \vee (q \wedge \neg r))$			
	3	$(p \wedge (\neg s \mathcal{U}(\neg q \wedge \neg s))) \vee (q \wedge \neg r)$			
Ele	1	$\bigcirc(\Diamond(call \wedge (\Box(\neg open))))$	1	$\bigcirc(\Diamond(call \wedge (\Box\neg open)))$	1,3
	2	$((\Diamond(\neg atfloor \wedge (\bigcirc open))) \mathcal{U}(call \wedge (\Box(\neg open)))) \vee (\Box(\Diamond(\neg atfloor \wedge (\bigcirc open))))$	2	$open \mathcal{U}(call \wedge (\Box\neg open))$	2,4
	3	$\Box(\neg atfloor \wedge (\bigcirc call))$	3	$(call \wedge (\Box(\neg open))) \vee (\bigcirc(\Box(call \wedge (\Box\neg open))))$	2,3,5
	4	$open \mathcal{U}(call \wedge (\Box\neg open))$			
	5	$(call \wedge (\Box(\neg open))) \vee (\bigcirc(\Box(call \wedge (\Box(\neg open)))))$			
TCP	1	$(delivered \wedge (send \wedge \neg ack)) \vee (send \wedge (ack \wedge \neg delivered))$	1	$\Diamond(((send \wedge \neg ack)\mathcal{U}(send \wedge (ack \wedge \neg delivered))) \vee (\Box(send \wedge \neg ack)))$	1,2,3
	2	$\Diamond(((send \wedge \neg ack)\mathcal{U}(send \wedge (ack \wedge \neg delivered))) \vee (\Box(send \wedge \neg ack)))$			
	3	$((delivered \wedge \neg ack)\mathcal{U}(send \wedge ack \wedge \neg delivered)) \vee$ $(\Box(delivered \wedge \neg ack))$			
AAP	1	$\Diamond(r \wedge p)$	1	$\Diamond(r \wedge p)$	1,2,3,5
	2	$(r \wedge p) \vee (\Diamond(r \wedge (\Diamond s)))$	2	$r \wedge (\bigcirc(p))$	4,5
	3	$(r \wedge p) \vee (\bigcirc(r \wedge (\Diamond q)))$			
	4	$r \wedge (\bigcirc(p))$			
	5	$((r \wedge p)\mathcal{U}(\Box(p \wedge (\Box(\neg q))))) \vee (\Box(r \wedge p))$			
MP	1	$\Diamond(h \wedge \neg m \wedge p \wedge \bigcirc(\neg h \wedge \neg p \vee hw \wedge (m \vee \neg p)))$	1	$\Diamond(h \wedge m)$	1,2,3,4,5,6
	2	$\Diamond(m \wedge h)$			
	3	$(m \wedge h) \vee \Diamond(h \wedge (\bigcirc\neg p))$			
	4	$(m \wedge h) \vee (m \wedge (\bigcirc p))$			
	5	$(m \wedge (\bigcirc p)) \mathcal{U}(\bigcirc(m \wedge h)) \vee \Box(m \wedge (\bigcirc p))$			
	6	$\Box(h) \vee (m \wedge h)$			
ATM	1	$\Diamond((\neg p \wedge (\bigcirc\neg l)) \vee (\neg m \wedge (p \wedge \neg l)))$	1	$\Diamond((\neg p \wedge (\bigcirc\neg l)) \vee (\neg m \wedge (p \wedge \neg l)))$	1,2,3,4,5,6,7,8
	2	$\Diamond((\neg p \wedge (m \vee (\Box\neg l))) \vee (\neg m \wedge (p \wedge \neg l)))$			
	3	$\Diamond(((\neg m \wedge (p \wedge (\neg l))) \mathcal{U}(\neg p \wedge (m \vee (\bigcirc\neg l)))) \vee \Box(\neg m \wedge (p \wedge \neg l)))$			
	4	$\Diamond(\Box((\neg p \mathcal{U}(\neg m \wedge (p \wedge \neg l))) \vee \Box\neg p))$			
	5	$((\neg m \wedge p) \mathcal{U}(\neg p \wedge (m \vee (\bigcirc\neg l)))) \vee \Box(\neg m \wedge p)$			
	6	$(\neg p \wedge (m \vee (\bigcirc\neg l))) \vee (\neg m \wedge (p \wedge \neg l))$			
	7	$\bigcirc((\neg p \wedge (m \vee (\bigcirc\neg l))) \vee (\neg m \wedge (p \wedge \neg l)))$			
	8	$\Box(((\neg p \mathcal{U}(\bigcirc l)) \vee \Box\neg p) \vee (\neg m \wedge \neg l))$			
RRCS	1	$\Diamond((\Diamond(cc \wedge tc)) \vee (\bigcirc(go \wedge ta)))$	1	$(cc \wedge tc) \vee (\Diamond(go \wedge ta))$	1,2,3,4
	2	$(\Diamond(cc \wedge tc)) \vee (go \wedge ta)$			
	3	$(cc \wedge tc) \vee (\Diamond(go \wedge ta))$			
	4	$(\Diamond(cc \wedge tc)) \vee (\Box(go \wedge tc))$			
Tel	1	$\Diamond(((\top \wedge \neg d) \mathcal{U}(\neg c \wedge (\top \wedge \neg d))) \wedge c)$	1	$\Diamond((\neg d \mathcal{U}(f \wedge \neg d)) \wedge c)$	1,2,3,4
	2	$\Diamond((\neg d \mathcal{U}(o \wedge \neg d)) \wedge c)$			
	3	$\Diamond((\neg d \mathcal{U}(f \wedge \neg d)) \wedge c)$			
	4	$\Diamond(\bigcirc(((\neg d \mathcal{U}(\neg c \wedge \neg d)) \vee (\Box\neg d)) \wedge c))$			
RRA	1	$\Diamond((((\bigcirc(r1 \wedge (\Box\neg g1))) \mathcal{U}(\bigcirc(g0 \wedge g1))) \vee \Box(\bigcirc(r1 \wedge (\Box\neg g1)))) \vee (r0 \wedge (\Box g1)))$	1	$\Diamond((\bigcirc(r1 \wedge (\Box\neg g1))) \vee (r0 \wedge (\Box g1)))$	1,2,3,4,5
	2	$\Diamond((\bigcirc(r1 \wedge (\Box\neg g1))) \vee (r0 \wedge (\Box g1)))$			
	3	$(((\bigcirc(r1 \wedge (\Box\neg g1))) \mathcal{U}(g0 \wedge g1)) \vee \Box(\bigcirc(r1 \wedge (\Box\neg g1)))) \vee (r0 \wedge (\Box g1))$			
	4	$\bigcirc(((\bigcirc(r1 \wedge (\Box g0))) \mathcal{U}(r0 \wedge (\Box r1))) \vee \Box(\bigcirc(r1 \wedge (\Box g0))))$			
	5	$((((\bigcirc(r1 \wedge (\Box\neg g1))) \mathcal{U}(\bigcirc(r1 \wedge g1))) \vee \Box(\bigcirc(r1 \wedge (\Box\neg g1)))) \mathcal{U}(r0 \wedge (\Box g1)))\vee$ $\Box((((\bigcirc(r1 \wedge (\Box\neg g1))) \mathcal{U}(\bigcirc(r1 \wedge g1))) \vee \Box(\bigcirc(r1 \wedge (\Box\neg g1)))))$			

A. Benchmarks

We evaluated contrasty on 16 different cases introduced by [9]. The details of each case are shown in Table I including the numbers of domain properties (column '#Dom'), goals (column '#Goal'), variables (column '#Var'), and the total size of all formulae (column 'Size') for the specification of each case. The order of the cases is sorted by the size of all formulae from small to large.

B. Experimental Setups

We used the following experimental setups.

- We employed the state-of-the-art BC solver[2] [9] denoted by GA to identify BCs. It is based on a genetic algorithm to search BCs.
- We followed the configuration of GA described in the paper [9] including the size of the initial population generated from such a specification and the limit of 50 generations, *i.e.*, 50 evolutions of the genetic algorithm population.
- We invoked Aalta [22] as the LTL satisfiability checker to check whether an LTL formula is a BC, whether one BC is more general than the other, and whether one BC is a witness of the other. Note that GA [9] also used Aalta as the LTL satisfiability checker.
- We computed the likelihood of a BC by the method [8]. And we set k to 1000, which is used in the paper [8] for good accuracy.
- For each case, we ran the algorithm 10 times and reported the mean data.
- All the experiments were run on the 2.13GHz Intel E7-4830, with 128 GB memory under GNU/Linux (Ubuntu 16.04).

C. Experimental Results

Table III summarizes the number of BC in $\mathcal{B}$ ('$|\mathcal{B}|$'), $\mathcal{B}_g$ ('$|\mathcal{B}_g|$'), and $\mathcal{B}_c$ ('$|\mathcal{B}_c|$'), where the column '#suc.' means the number of successful runs (out of 10 runs). If GA fails in all 10 runs, the results are marked by 'N/A'. Overall, our method can solve all the cases that can be solved by GA to identify BCs. Clearly, if the solver cannot identify BCs, our method cannot perform the post-processing.

For most cases, GA returns a large number of BCs thanks to the development of search-based methods. Note that such a large set of BC can cause a huge burden in the assessment stage and the resolution stage. Seeing the columns '$|\mathcal{B}_g|$' and '$|\mathcal{B}_c|$', we observe that the size of $\mathcal{B}_c$ is much smaller than that of $\mathcal{B}_g$ for all cases. It means that, compared with the generality metric, the contrasty metric can considerably reduce the number of BCs to be analyzed by engineers.

Table II summarizes the results of the different metrics, for the BCs identified for each of the case studies. We selected the data that GA got the most number of BC from 10 times experiments for display. The column 'GL' (*resp.* 'CL') illustrates the BCs ('BC') in $\mathcal{B}_g$ (*resp.* $\mathcal{B}_c$) and their rank

[2]http://dc.exa.unrc.edu.ar/staff/rdegiovanni/ASE2018.html

TABLE III
THE NUMBER OF BC RECOMMENDED BY DIFFERENT METRICS

Case	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{B}_g\rvert$	$\lvert\mathcal{B}_c\rvert$	#suc.
RP1	37.1	3.2	**1.2**	10
RP2	35.1	2.6	**1.2**	10
Ele	28	3.2	**2.6**	10
TCP	53.9	2.1	**1.5**	10
AAP	50.3	3.7	**1.8**	10
MP	40.7	4.5	**1.4**	10
ATM	64.4	3.4	**1.2**	10
RRCS	27.9	3	**1**	10
Tel	36.5	3	**1**	2
LAS	N/A	N/A	N/A	N/A
PA	N/A	N/A	N/A	N/A
RRA	40.571	3.14	**1**	7
SA	N/A	N/A	N/A	N/A
LB	N/A	N/A	N/A	N/A
LC	N/A	N/A	N/A	N/A
AMBA	N/A	N/A	N/A	N/A

('Rank') based on the likelihood metric. We also use the column 'Rank' as the identification of BCs. For every BCs ϕ in $\mathcal{B}_c$, we report which BCs in $\mathcal{B}_g$ ('Witness') ϕ is a witness of and the identification of ϕ in $\mathcal{B}_g$ is marked in red.

For all cases, $\mathcal{B}_c$ is much smaller than $\mathcal{B}_g$ and $\mathcal{B}_c$ is a subset of $\mathcal{B}_g$, which confirms that the contrasty metric is a more finer-grained metric than the generality metric. The results also show that a set of general BCs still retains the BCs that represent the same divergence. Particularly, for MP, ATM, and RRA, the redundant BCs are too much to assess and resolve divergences efficiently.

From the column 'Witness', every BC in $\mathcal{B}_g$ can find a witness of it in $\mathcal{B}_c$. This observation means that the BCs in $\mathcal{B}_c$ capture all the divergences captured by the BCs in $\mathcal{B}_g$. Therefore, engineers only need to consider the BCs in $\mathcal{B}_c$ when resolving divergences. In addition, we also observe that the contrastive BCs rank lower in $\mathcal{B}_g$ in Ele, TCP, AAP, MP, RRCS, TEL, and RRA. The reason, as mentioned above, is that the circumstances that cannot describe the divergence lead to mistakes of likelihood. Such mistakes are serious, which will prevent engineers from grasping the main cause of the divergence quickly. It leads to costly assessing and resolving the same divergence repeatedly.

In summary, the generality metric cannot capture the difference between BCs. Surprisingly, lots of BCs identified by the state-of-the-art BC solver are redundant in most cases. It puts an expensive burden on assessing and resolving divergences. The method we propose can compare this well and give a recommendation that is more conducive to saving the costs of assessing and resolving divergences.

VI. JOINT FRAMEWORK

In this section, we design a joint framework to interleave filtering based on the contrasty metric with identifying BCs (JFC). We first introduce the termination condition for identifying BCs and then propose JFC.

Motivated by the blocking clause approach to solving All-SAT problem [25], we consider excluding the circumstances

captured by identified BCs in the search process to generate a search bias towards the BCs that capture different divergences. Specifically, in the process of searching for BCs, once a BC ϕ is identified, we add $\neg\phi$ as an additional constraint to domain properties. The additional constraint makes the domain properties dynamically change so that it can prevent the same circumstances from being identified as a BC again (Theorem 7). Moreover, we will prove that the BCs under the additional constraint are also BCs under the original domain properties and goals (Theorem 6).

Before introducing JFc, We first propose a sufficient condition for the case where there does not exist a BC (called *BC termination condition*).

Theorem 5. *Let Dom be domain properties and G goals. If $\exists 1 \leq i \leq |G|, Dom \wedge G_{-i} \wedge \neg G_i \models \bot$, then there does not exist a BC under Dom and G.*

Sketch of proof. We prove that if there exists a BC, then $\forall 1 \leq i \leq |G|, Dom \wedge G_{-i} \wedge \neg G_i \not\models \bot$. If there is a BC ϕ under Dom and G, then $Dom \wedge G \wedge \phi \models \bot$ (logical inconsistency) and $\forall 1 \leq i \leq |G|, Dom \wedge G_{-i} \wedge \phi \not\models \bot$ (minimality). Because of the logical inconsistency, we have $\phi \rightarrow \neg(Dom \wedge G)$. Therefore, $Dom \wedge G_{-i} \wedge \phi \rightarrow Dom \wedge G_{-i} \wedge \neg(Dom \wedge G)$. Consider the minimality, we have $\forall 1 \leq i \leq |G|, Dom \wedge G_{-i} \wedge \neg(Dom \wedge G) \not\models \bot$, *i.e.*, $\forall 1 \leq i \leq |G|, Dom \wedge G_{-i} \wedge \neg G_i \not\models \bot$. □

Based on Theorem 5, we can check whether there still exists a BC under the dynamical domain properties and goals.

Algorithm 3: JFc

Input: domain properties Dom and goals G.
Output: a set of contrastive BCs $\mathcal{B}_c$.

```
1  B_c ← ∅;
2  while True do
3      isEnd, φ ← CALLBCSOLVER(Dom ∪ {¬φ|φ ∈ B_c}, G);
4      if isEnd then
5          return B_c;
6      else
7          W ← INTERNALCONTRASTYFILTER(φ, B_c);
8          B_c ← B_c/W;
9          if there is not a BC under
             Dom ∪ {¬φ|φ ∈ B_c} and G then
10             return B_c;
```

JFc takes the domain properties Dom and goals G as inputs. Its output is a set of contrastive BCs $\mathcal{B}_c$. The pseudo code is outlined in Algorithm 3. In order to identify BCs, we involve existing BC solvers, *e.g.*, GA [9] and Tab [10] (Alg. 3 of line 3). Note that we consider the dynamical domain properties ($Dom \cup \{\neg\varphi|\varphi \in \mathcal{B}_c\}$). If the BC solver terminates, we return $\mathcal{B}_c$ (Alg. 3 of line 5). Otherwise, unlike PPFc, we update $\mathcal{B}_c$ when identifying a new BC (Alg. 3 of line 7-8).

Algorithm 4: INTERNALCONTRASTYFILTER

Input: a BC ϕ and a set of BCs $\mathcal{B}$.
Output: a set of BCs W filtered by ϕ.

```
1  W ← ∅;
2  for each BC φ ∈ B do
3      if φ is a witness of φ then
4          W ← W ∪ {φ};
5  return True, W;
```

Note that we only remove the BCs which the new BC is a witness of (Alg. 4 of line 3) because none of the BCs in $\mathcal{B}_c$ is a witness of the new BC (Theorem 7). Afterward, if there still exists a BC under $Dom \cup \{\neg\varphi|\varphi \in \mathcal{B}_c\}$ and G, we continue to involve BC solver; otherwise, return $\mathcal{B}_c$ (Alg. 3 of line 9).

Theorem 6. *Let Dom be domain properties, G goals, and $\mathcal{B}$ a set of BCs that has been identified. A LTL formula ϕ is a BC under Dom and G, if ϕ is a BC under $Dom \cup \{\neg\varphi|\varphi \in \mathcal{B}\}$ and G.*

Sketch of proof. Because $\forall\varphi \in \mathcal{B}$ is a BC under Dom and G, we have $Dom \wedge (\bigwedge_{\varphi\in\mathcal{B}} \neg\varphi) \wedge G \wedge \phi \equiv Dom \wedge G \wedge \phi$. Therefore, $Dom \wedge (\bigwedge_{\varphi\in\mathcal{B}} \neg\varphi) \wedge G \wedge \phi \models \bot$ (logical inconsistency) holds. Because $Dom \wedge (\bigwedge_{\varphi\in\mathcal{B}} \neg\varphi) \wedge G_{-i} \wedge \phi \rightarrow Dom \wedge G_{-i} \wedge \phi$, $\forall 1 \leq i \leq |G|, Dom \wedge (\bigwedge_{\varphi\in\mathcal{B}} \neg\varphi) \wedge G_{-i} \wedge \phi \not\models \bot$ (minimality) holds. The non-triviality obviously holds. □

Theorem 6 shows that although the additional constraint is considered, the results are still BCs under the original domain properties and goals.

Theorem 7. *In Algorithm 3, $\nexists\varphi \in \mathcal{B}_c$ s.t. φ is a witness of ϕ.*

Sketch of proof. We prove Theorem 7 by inductive hypothesis as follows.

- At the first iteration where $\mathcal{B}_c$ is an empty set, assume we get a BC φ_1, Theorem 7 holds.
- We suppose that at the k-th iteration where we get a BC φ_k, Theorem 7 holds.
- At the k+1-th iteration where $\mathcal{B}_c = \{\varphi_1, \ldots, \varphi_k\}$, assume we get a BC ϕ. Because ϕ is a BC under $Dom \cup \{\neg\varphi|\varphi \in \mathcal{B}_c\}$ and G, $\forall 1 \leq i \leq |G|, Dom \wedge (\bigwedge_{\varphi\in\mathcal{B}_c} \neg\varphi) \wedge G_{-i} \wedge \phi \not\models \bot$. Therefore, for every $\varphi_j \in \mathcal{B}_c$, $\phi \wedge \neg\varphi_j$ is a BC under $Dom \cup \{\neg\varphi|\varphi \in \mathcal{B}_c \wedge \varphi \neq \varphi_j\}$ and G. Because of Theorem 6, $\phi \wedge \neg\varphi_j$ is a BC under Dom and G.

□

Intuitively, based on Theorem 7, JFc can produce a search bias towards the BCs that capture different divergences.

Theorem 8. *In Algorithm 3, the BCs in the final $\mathcal{B}_c$ are not witnesses with each other.*

It is straightforward to prove Theorem 8 because of Theorem 7 and Algorithm 4. Theorem 8 guarantees that Algorithm 3 returns a set of contrastive BCs.

TABLE IV
THE OVERALL PERFORMANCE OF PPFC AND JFC

Case	PPFC					JFC				
	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{B}_c\rvert$	GA t. (s)	t. (s)	#suc.	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{B}_c\rvert$	#T	t. (s)	#suc.
RP1	37.1	1.2	157.4	224.53	10	**1**	**1**	10	**29.5**	10
RP2	35.1	1.2	130.2	206	10	**1.1**	**1.1**	10	**78.9**	10
Ele	28	2.6	45.8	88.01	10	**2.1**	**2.1**	10	**43.4**	10
TCP	53.9	1.5	225.1	**308.26**	10	**1.4**	**1.4**	0	801.6	10
AAP	50.3	1.8	65.3	208.64	10	**1**	**1**	10	**41.3**	10
MP	40.7	1.4	59.3	146.02	10	**1**	**1**	10	**60.8**	10
ATM	64.4	1.2	102.2	259.19	10	**1**	**1**	10	**25.2**	10
RRCS	27.9	1	68.3	91.87	10	**1**	1	10	**15**	10
Tel	36.5	1	35.3	46.53	2	**1**	1	10	**27**	**10**
LAS	N/A	N/A	N/A	N/A	0	N/A	N/A	0	N/A	0
PA	N/A	N/A	N/A	N/A	0	N/A	N/A	0	N/A	0
RRA	40.571	1	696.43	878.7	7	**1**	1	10	**255.1**	**10**
SA	N/A	N/A	N/A	N/A	0	N/A	N/A	0	N/A	0
LB	N/A	N/A	N/A	N/A	0	N/A	N/A	0	N/A	0
LC	N/A	N/A	N/A	N/A	0	N/A	N/A	0	N/A	0
AMBA	N/A	N/A	N/A	N/A	0	N/A	N/A	0	N/A	0

VII. EXPERIMENTS

In this section, we conducted extensive experiments on a broad range of benchmarks shown in Table I to evaluate the performance of JFC by comparing with PPFC. We first presented the research questions.

RQ 2. *What is the performance of the joint framework (JFC) for producing the contrastive BC set compared with the post-processing framework (PPFC)?*

A. Experimental Setups

The experimental setups used in this section was the same as the one described in Section V. In addition, we added the new experimental setups.

- We set the same BC solver (GA [9]) for PPFC and JFC.
- We invoked Aalta [22] to check the BC termination condition.

B. Experimental Results

Table IV shows the overall performance of PPFC and JFC, including the running time of GA ('GA t.'), the running time of the framework ('t.'), and the number of meeting the BC termination condition ('#T'). In JFC, $\mathcal{B}$ records all BCs identified during the search.

From the column '$|\mathcal{B}_c|$', the contrastive BCs obtained by JFC is slightly less than that obtained by PPFC. This is because JFC not only considers the contrasty in BC but also considers the BC termination condition where JFC searches for a set of contrastive BCs that is enough so that there is no BC in the domain properties and goals after avoiding these contrastive BCs. We also observe that the size of $\mathcal{B}$ of JFC is much smaller than that of PPFC. Moreover, for JFC, the size of $\mathcal{B}$ is close to that of $\mathcal{B}_c$. These observations show that JFC produces a strong search bias towards the BCs that are contrastive with the identified BCs.

In PPFC, the running time of GA is approximately the same as that of producing a set of contrastive BCs. And the running time of producing a set of contrastive BCs increases as the number of BCs identified by GA increases. In particular, in AAP, MP, and ATM, the running time of producing a set of contrastive BCs is about 1.5 times that of GA. It indicates the drawback of PPFC, namely, the cost of producing a set of contrastive BCs is proportional to the number of BCs identified by a BC solver. It is foreseeable that the redundant BCs in $\mathcal{B}$ will greatly reduce the efficiency of producing a set of contrastive BCs.

JFC deals with the drawback of PPFC, because JFC uses the identified contrastive BC for pruning during the search process, thereby avoiding searching for the redundant BCs. The shorter running time for meeting the BC termination condition confirms this conclusion. If JFC meets the BC termination condition, JFC will produce a set of contrastive BCs efficiently.

Particularly, in RP1 and ATM, JFC is 10 times faster than PPFC. We also observe that if JFC does not meet the BC termination condition (only TCP), JFC is slower than PPFC. It is reasonable because JFC additionally checks the BC termination condition after finding a new BC.

Conclusively, JFC produces the search bias towards contrastive BCs. In addition, the efficiency of JFC is not limited to the number of BCs identified by a BC solver.

VIII. RELATED WORK

Inconsistency management, *i.e.*, how to deal with inconsistencies in requirements, has also been the focus of several studies, in particular on the formal side. Besides the inconsistency management approaches based on the informal or semi-formal methods, such as [15], [16], [18], [19], a series of formal approaches [11], [12], [14], [27] recently have been proposed, which only focus on logical inconsistency or ontology mismatch. Another related approach is proposed by Nuseibeh and Russo [28], which generates the conjunction of ground literals as an explanation for the unsatisfiable specification based on abduction reasoning. As for consistency checking methods, we

have to mention the approach of Harel et al. [14], which identifies inconsistencies between two requirements represented as conditional scenarios. Moreover, the work [17], [23], [24] studied the reasoning about conflicts in requirements. In this paper, we focus on the situations that lead to goal divergences, which are nothing but weak inconsistencies.

Goal-conflict analysis has been widely used as an abstraction for risk analysis in GORE. It is typically driven by the identify-assess-control cycle, aimed at identifying, assessing and resolving inconsistencies that may obstruct the satisfaction of the expected goals.

In identifying inconsistencies, we have to mention the work on obstacle analysis. An obstacle, first proposed in [34], is a particular goal conflict, which captures the situation that only one goal is inconsistent with the domain properties. Alrajeh et al. [2] exploited the model checking technique to generate tracks that violate or satisfy the goals, and then to compute obstacles from these tracks based on the machine learning technique. Other approaches for obstacle analysis include [3]–[5], [34]. Whereas, as obstacles only capture the inconsistency for single goals, these approaches fail to deal with the situation where multiple goals are conflicting.

In this work, we focus on the other inconsistencies – boundary condition. Let us come back to the problem of identifying BCs. Existing approaches mainly categorize into construct-based approaches and search-based approaches. For construct-based approaches, Van Lamsweerde et al. [32] proposed a pattern-based approach which only returns a BC in a pre-defined limited form. Degiovanni et al. [10] exploited a tableaux-based approach that generates general BCs but only works on small specifications because tableaux are difficult to be constructed.

For the search-based approach, Degiovanni et al. [9] presented a genetic algorithm which seeks for BCs and handles specifications that are beyond the scope of previous approaches. Moreover, Degiovanni et al. [9] first proposed the concept of generality to assess BCs. Their work filtered out the less general BCs to reduce the set of BCs. However, the generality is a coarse-grained assessment metric.

As the number of identified inconsistencies increases, the assessment stage and the resolution stage become very expensive and even impractical. Recently, the assessment stage in GORE has been widely discussed to prioritize inconsistencies to be resolved and suggest which goals to drive attention to for refinements. However, some of the work [2]–[5], [34] assume that certain probabilistic information on the domain is provided and analyzes to simpler kinds of inconsistencies (obstacles).

In order to automatically assess BCs, Degiovanni et al. [8] recently have proposed an automated approach to assess how likely conflict is, under an assumption that all events are equally likely. They estimated the likelihood of BCs by counting how many models satisfy a circumstance captured by a BC. However, the number of models cannot accurately indicate the likelihood of divergence, because not all the circumstances captured by a BC result in divergence. In this paper, we discovered the drawbacks and proposed a new metric to avoid evaluation mistakes for the likelihood.

For the resolution of conflicts, Murukannaiah et al. [26] resolved the conflicts among stakeholder goals of system-to-be based on the Analysis of Competing Hypotheses technique and argumentation patterns. Related works on conflict resolution also include [13] which calculates the personalized repairs for the conflicts of requirements with the principle of model-based diagnosis.

However, these approaches presuppose that the conflicts have been already identified and our approach for boundary condition discovery provides a footstone for solving these problems. Let us recall Example 1. Letier et al. [21] resolved the BC by refining the first goal as: the pump is switched on when the water level is high and there is no methane. Formally, $\Box((h \wedge \neg m) \rightarrow \bigcirc(p))$.

IX. Conclusion and Future Work

Providing a reasonable set of BCs for assessing and resolving divergences is of great significance both from an economical perspective and an impact on software quality. In this paper, we have proposed a new metric, contrasty, to deal with the drawbacks caused by the generality metric. Because BCs are ultimately used for resolving divergences, we argue that the identified BCs should help to assess and resolve divergences. The contrasty metric mainly distinguishes the difference between BCs from the point of resolving divergences. Experimental results have shown the advantage of contrasty metric, namely, it filters out the BCs capturing the same divergence. It helps to avoid costly reworks, *i.e.*, assessing and resolving the same divergence captured by redundant BCs. In addition, we have designed a joint framework to improve the performance of the post-processing framework.

Future work will extend our contrasty metric to the assessment stage and the resolution stage.

Acknowledgment

We thank Fangzhen Lin, Yongmei Liu, Jianwen Li, and Ximing Wen for discussion on the paper and anonymous referees for helpful comments.

References

[1] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel, "Learning operational requirements from goal models," in *ICSE*, 2009, pp. 265–275.
[2] D. Alrajeh, J. Kramer, A. Van Lamsweerde, A. Russo, and S. Uchitel, "Generating obstacle conditions for requirements completeness," in *ICSE*, 2012, pp. 705–715.
[3] A. Cailliau and A. Van Lamsweerde, "A probabilistic framework for goal-oriented risk analysis," in *RE*, 2012, pp. 201–210.
[4] ——, "Integrating exception handling in goal models," in *RE*, 2014, pp. 43–52.
[5] A. Cailliau and A. van Lamsweerde, "Handling knowledge uncertainty in risk-based requirements engineering," in *RE*, 2015, pp. 106–115.
[6] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuxmv symbolic model checker," in *CAV*, 2014, pp. 334–342.
[7] R. Degiovanni, D. Alrajeh, N. Aguirre, and S. Uchitel, "Automated goal operationalisation based on interpolation and sat solving," in *ICSE*, 2014, pp. 129–139.

[8] R. Degiovanni, P. Castro, M. Arroyo, M. Ruiz, N. Aguirre, and M. Frias, "Goal-conflict likelihood assessment based on model counting," in *ICSE*, 2018, pp. 1125–1135.
[9] R. Degiovanni, F. Molina, G. Regis, and N. Aguirre, "A genetic algorithm for goal-conflict identification," in *ASE*, 2018, pp. 520–531.
[10] R. Degiovanni, N. Ricci, D. Alrajeh, P. Castro, and N. Aguirre, "Goal-conflict detection based on temporal satisfiability checking," in *ASE*, 2016, pp. 507–518.
[11] C. Ellen, S. Sieverding, and H. Hungar, "Detecting consistencies and inconsistencies of pattern-based functional requirements," in *FMICS*, 2014, pp. 155–169.
[12] N. A. Ernst, A. Borgida, J. Mylopoulos, and I. J. Jureta, "Agile requirements evolution via paraconsistent reasoning," in *CAiSE*, 2012, pp. 382–397.
[13] A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan, "Plausible repairs for inconsistent requirements," in *IJCAI*, 2009, pp. 791–796.
[14] D. Harel, H. Kugler, and A. Pnueli, "Synthesis revisited: Generating statechart models from scenario-based requirements," in *Formal Methods in Software and Systems Modeling*, 2005, pp. 309–324.
[15] J. H. Hausmann, R. Heckel, and G. Taentzer, "Detection of conflicting functional requirements in a use case-driven approach," in *ICSE*, 2002, pp. 105–115.
[16] S. J. Herzig and C. J. Paredis, "A conceptual basis for inconsistency management in model-based systems engineering," *Procedia CIRP*, vol. 21, pp. 52–57, 2014.
[17] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *RE*, 2010, pp. 115–124.
[18] M. Kamalrudin, "Automated software tool support for checking the inconsistency of requirements," in *ASE*, 2009, pp. 693–697.
[19] M. Kamalrudin, J. Hosking, and J. Grundy, "Improving requirements quality using essential use case interaction patterns," in *ICSE*, 2011, pp. 531–540.
[20] J. Kramer, J. Magee, M. Sloman, and A. Lister, "Conic: an integrated approach to distributed computer control systems," *IET Computers & Digital Techniques*, vol. 130, no. 1, pp. 1–10, 1983.
[21] E. Letier *et al.*, "Reasoning about agents in goal-oriented requirements engineering," Ph.D. dissertation, PhD thesis, Université catholique de Louvain, 2001.
[22] J. Li, S. Zhu, G. Pu, and M. Y. Vardi, "Sat-based explicit ltl reasoning," in *HVC*, 2015, pp. 209–224.
[23] C.-L. Liu, "Ontology-based conflict analysis method in non-functional requirements," in *ACIS-ICIS*, 2010, pp. 491–496.
[24] D. Mairiza and D. Zowghi, "Constructing a catalogue of conflicts among non-functional requirements," in *ENASE*, 2010, pp. 31–44.
[25] K. L. McMillan, "Applying sat methods in unbounded symbolic model checking," in *CAV*, 2002, pp. 250–264.
[26] P. K. Murukannaiah, A. K. Kalia, P. R. Telangy, and M. P. Singh, "Resolving goal conflicts via argumentation-based analysis of competing hypotheses," in *RE*, 2015, pp. 156–165.
[27] T. H. Nguyen, B. Q. Vo, M. Lumpe, and J. Grundy, "Kbre: a framework for knowledge-based requirements engineering," *Software Quality Journal*, vol. 22, no. 1, pp. 87–119, 2014.
[28] B. Nuseibeh and A. Russo, "Using abduction to evolve inconsistent requirements specification," *Australasian J. of Inf. Systems*, vol. 7, no. 1; SPI, pp. 118–130, 1999.
[29] A. Pnueli, "The temporal logic of programs," in *Annual Symposium on Foundations of Computer Science*, 1977, pp. 46–57.
[30] A. P. Sistla and E. M. Clarke, "The complexity of propositional linear temporal logics," *J. ACM*, vol. 32, no. 3, pp. 733–749, 1985.
[31] A. Van Lamsweerde, *Requirements engineering: From system goals to UML models to software*. Chichester, UK: John Wiley & Sons, 2009, vol. 10.
[32] A. Van Lamsweerde, R. Darimont, and E. Letier, "Managing conflicts in goal-driven requirements engineering," *IEEE Trans. Software Eng.*, vol. 24, no. 11, pp. 908–926, 1998.
[33] A. Van Lamsweerde and E. Letier, "Integrating obstacles in goal-driven requirements engineering," in *ICSE*, 1998, pp. 53–62.
[34] ——, "Handling obstacles in goal-oriented requirements engineering," *IEEE Trans. Software Eng.*, vol. 26, no. 10, pp. 978–1005, 2000.