1 Compositional Fault Analysis

1.1 Summary of the Algorithms

Max fault hypothesis for each top level property

(i) Step 1: Verify Nominal Model and Generate Minimal Inductive Validity Cores (MIVCs)

Assuming the nominal model is defined using AADL and AGREE and the nominal behaviors and top level safety properties are verified, the user must also define the faulty behavior using the Safety Annex. Each of the subcomponents of the system have faults defined and each top level property has a Max Fault Hypothesis statement (e.g. max N fault for $N \in \mathbb{N}$).

At this point, the user can select to perform compositional max fault verification and behind the scenes, the verification process is as follows. The verification proceeds in a top-down compositional approach and all properties at all layers are verified valid. In the IVC analysis, the model elements considered depend on the layer of the architecture currently being verified. For a leaf level, only fault activation literals are considered for the IVC analysis. At an intermediate layer, the current layer fault activation literals are considered as well as contracts (if there exists a subcomponent of this layer). During verification, the MIVCs are collected.

The MIVCs collected for each verification layer are the Minimal Unsatisfiable Subsets for the constraint system with the top level property set to false. For the bottom layers, the MIVCs only contain fault activation literals constrained to false. For the upper levels, the MIVCs contain both properties and fault activation literals constrained to false.

(ii) Step 2: Compute Minimal Cut Sets (MinCutSets) from the IVCs

For each of the MIVCs (MUSs) collected at each verification layer, we compute the MinCutSets from the IVCs from the bottom up. Using the Hitting Set Algorithm, the Minimal Correction Sets (MCSs) are collected. Then we transform the MCSs into MinCutSets.

Bottom layer: For each MCS of an upper level property, all fault activation literals in the MCS are constrained to true in order to obtain the MinCutSets for the violation of that upper level property. Since we are looking at only some maximum number of faults, we can eliminate from consideration the MCSs with cardinality greater than N from the Max N Hypothesis.

Upper layer: For each MCS, any element that is not a fault activation literal (i.e. any property violation) is replaced/inlined with the MinCutSet boolean expression obtained at the lower level for the violation of that property. If any of the MinCutSets from the lower level are empty, replace this property with

false (indicating that this property cannot happen). If any of the MinCutSets have cardinality greater than the threshold, we disregard these sets.

What remains is the MinCutSets with cardinality less than or equal to the max fault hypothesis for the violation of each top level property.

(iii) Step 3: Determine Compositional Verification Results for a Property

If no MinCutSets are obtained using Step 2, then the property is valid with this hypothesis.

If some MinCutSets are obtained using Step 2, then the property is invalid with this hypothesis and each of the MinCutSets shows a counterexample indicating that when these faults are active, the property is violated.

Probabilistic fault hypothesis for each top level property

(i) Step 1: Verify Nominal Model and Generate MIVCs

This process is the same as described in Step 1 for Max Fault Hypothesis algorithm.

(ii) Step 2: Compute Fault Combinations based on Fault Probabilities and Hypotheses

Using monolithic analysis, we collect all fault combinations whose probability exceeds or is equal to the specified threshold.

(iii) Step 3: Compute Minimal Cut Sets from the IVCs

For each of the MIVCs (MUSs) collected at each verification layer, we compute the MinCutSets from the IVCs from the bottom up. Using the Hitting Set Algorithm, the Minimal Correction Sets (MCSs) are collected. Then we transform the MCSs into MinCutSets.

Bottom layer: For each MCS of an upper level property, all fault activation literals in the MCS are constrained to true in order to obtain the MinCutSets for the violation of that upper level property. If a MinCutSet is not a subset of any fault combinations computed in the last step, we can eliminate this MinCutSet from consideration.

Upper layer: For each MCS, any element that is not a fault activation literal (i.e. any property violation) is replaced/inlined with the MinCutSet boolean expression obtained at the lower level for the violation of that property. If any of the MinCutSets from the lower level are empty, replace this property with *false* (indicating that this property cannot happen). If any of the MinCutSets have probability greater than the threshold, we disregard these sets.

What remains is the MinCutSets with probability equal to or greater than the probabilistic threshold specified for a given top level property.

(iv) Step 4: Determine Compositional Verification Results for a Property

If no MinCutSets are obtained using Step 2, then the property is valid with this hypothesis.

If some MinCutSets are obtained using Step 2, then the property is invalid with this hypothesis and each of the MinCutSets shows a counterexample indicating that when these faults are active, the property is violated.

1.2 Theory

Definitions:

Given a constraint system C where C is an ordered set of abstract constraints over some set of variables, $\{C_1, ..., C_n\}$. The satisfiability problem is in conjuntive normal form (CNF): $C = \wedge_{i=1,...,n} C_i$; and each C_i is a disjunction of literals $C_i = l_{i1} \vee ... \vee l_{ik_i}$ where each literal $l_{ij}$is either a Boolean variable x or its negation $\neg x$.

Satisfiability (SAT) : A CNF is satisfiable iff there exists an assignment of truth values to its variables such that the formula evaluates to true. If not, it is unsatisfiable (UNSAT).

Intuitively a constraint system contains the contracts that constrain component behavior and faults that are defined over these components. In the case of the nominal model, a constraint system is defined as follows:

Let F be the set of all faults defined in the model and G be the set of all component contracts (guarantees). $C = \{C_1, C_2, ..., C_n\}$ where for $i \in \{1, ..., n\}$, C_i has the following constraints for any $f_j \in F$ and $g_k \in G$ with regard to the top level property P:

$$C_i \in \begin{cases} f_j : inactive \\ g_k : true \\ P : true \end{cases}$$

Given a state space S, a transition system (I, T) consists of the initial state predicate $I : S \to \{0, 1\}$ and a transition step predicate $T : S \times S \to \{0, 1\}$. Reachability for (I, T) is defined as the smallest predicate $R : S \to \{0, 1\}$ which satisfies the following formulas:

$$\forall s. I(s) \Rightarrow R(s)$$
$$\forall s, s'. R \wedge T(s, s') \Rightarrow R(s')$$

A safety property $\mathcal{P} : S \rightarrow \{0, 1\}$ is a state predicate. A safety property $\mathcal{P}$ holds on a transition system (I, T) if it holds on all reachable states. More formally, $\forall s.R(s) \Rightarrow \mathcal{P}(s)$. When this is the case, we write $(I, T) \vdash \mathcal{P}$.

Given a transition system which satisfies a safety property P, it is possible to find which parts of the system are necessary for satisfying the safety property through the use of Ghassabani's *All Minimal Inductive Validity Cores* algorithm [1, 2]. This algorithm makes use of the collection of all minimal unsatisfiable subsets of a given transition system in terms of the negation of the top level property, i.e. the top level event. This is a minimal subset of the activation literals such that the constraint system C with the formula $\neg P$ is unsatisfiable when these activation literals are held true. Formally:

MUS : A Minimal Unsatisfiable Subset M of a constraint system C is : $\{M \subseteq C | M$ is UNSAT and $\forall c \in M$: $M \setminus \{c\}$ is SAT$\}$. This is the minimal explaination of the constraint systems infeasability.

A closely related set is a *minimal correction set* (MCS). The MCSs describe the minimal set of model elements for which if constraints are removed, the constraint system is satisfied. For C, this corresponds to which faults are not constrained to inactive (and are hence active) and violated contracts which lead to the violation of the safety property. In other words, the minimal set of active faults and/or violated properties that lead to the top level event.

MCS : A Minimal Correction Set M of a constraint system C is : $\{M \subseteq C | C \setminus M$ is SAT and $\forall S \subset M : C \setminus M$ is UNSAT$\}$. A MCS can be seen to "correct" the infeasability of the constraint system.

A duality exists between MUSs of a constraint system and MCSs as established by Reiter [4]. This duality is defined in terms of *hitting sets*. A hitting set of a collection of sets A is a set H such that every set in A is "hit" be H; H contains at least one element from every set in A [3].

Hitting Set: Given a collection of sets K, a hitting set for K is a set $H \subseteq \cup_{S \in K} S$ such that $H \cap S \neq \emptyset$ for each $S \in K$. A hitting set for K is minimal if and only if no proper subset of it is a hitting set for K.

Utilizing this approach, we can easily collect the MCSs from the MUSs provided through the all MIVC algorithm.

Cut sets and minimal cut sets provide important information about the vulnerabilities of a system. A *Minimal Cut Set* (MinCutSet) is a minimal collection of faults that lead to the violation of the safety property (or in other words, lead to the top level event). We define MinCutSet in terms of the constraint system in question as follows:

MinCutSet : Minimal Cut Set of a constraint system C with all faults in the system denoted as the set F is : $\{cut \subseteq F | C \setminus cut$ is SAT and $\forall c \subset cut : C \setminus c$ is UNSAT $\}$.

NOTE: Explain why the leaf level subcomponents have only faults and the intermediate levels have both faults and guarantees. We need this info in the proof.

When the MCS contains only faults, the MCS is equivalent to the MinCutSet as shown in the first part of the proof. When contracts exist in the MCS, a replacement can be made which transforms the MCS into the MinCutSet.

Theorem: The MinCutSet can be generated by transformation of the MCS.

Proof: For faults in the model F and subcomponent contracts G:
$MCS \cap F \neq \{\} \vee MCS \cap G \neq \{\}$.

Part 1: $MCS \cap G = \{\}$ (Leaf level of system)

(i) MCS $\subseteq$ MinCutSet:

Let $M \in MCS$. Then $C \setminus M$ is SAT. Since $!\exists g \in C$ for $g \in G$, thus $M \subseteq F$ and is a cut set.

By minimality of the MCS, M is a minimal cut set for $\neg P$.

(ii) $MinCutSet \subseteq$ MCS:

Let $M \in MinCutSet$. Then all faults in M cause $\neg P$ to occur by definition. Thus, $C \setminus M$ is SAT.

By minimality of $MinCutSet$, M is also minimal and thus is a minimal correction set.

Part 2: $MCS \cap G \neq \{\}$ (Intermediate level of system)

Assume $\overline{C} = \{F, G, P\}$ with the constraints that all $f \in F$ are inactive and all $g \in G$ are valid with regard to top level property P, i.e. the nominal model proves.

Let $MCS = \{f_1, ..., f_n, g_1, ..., g_m\}$
For $g_1 \in MCS$, $\exists \overline{F}_1 \subseteq F$ where $\overline{F}_1$ is a minimal set of active faults that cause the violation of g_1. Replace g_1 in MCS with $\overline{F}_1$. Then $MCS = \{f_1, ..., f_n, \overline{F}_1, g_2, ..., g_m\}$.

Perform this replacement for all $g_i \in MCS$ until we reach $MCS = \{f_1, ..., f_n, \overline{F}_1, ..., \overline{F}_m\}$.

Since $\overline{F}_i$ is minimal, the MCS retains its minimality. Furthermore $MCS \subseteq F$ and $C \setminus MCS$ is SAT. Therefore the MCS is transformed into MinCutSet.

1.3 Example

Let P be our top level safety property: the "good" behavior we want to happen.

P = (pressure $>$ threshold) $\implies$ shut down command, i.e. shut down when we should.

Our model elements are: $F = \{f1, f2, f3\}$ corresponding to:
$f1$ = sensor 1 fault (stuck at low)
$f2$ = sensor 2 fault (stuck at low)
$f3$ = sensor 3 fault (stuck at low)

For the *no voting* implementation, each sensor can have a stuck at low fault and the system shuts down when one of the sensors indicates high pressure. The constraint system for this example corresponds to : $C = \{\neg f1, \neg f2, \neg f3, \neg P\}$. This constraint system is given to the SAT solver. Assuming that the nominal model holds (P and model elements are satisfied), we get an UNSAT result with this constraint system. The SAT solver provides all counterexamples in the form of IVCs which are our Minimal Unsatisfiable Subsets (MUSs). Since these are all of the sets that show $\neg P$ is UNSAT, they also are the sets that prove P. For example $IVC_1 = \{\neg f1\}$: if sensor 1 fault is inactive, we can prove P : this is the minimal explanation of infeasibility with respect to C.

In this case, the IVCs generated are:
$IVC_1 = \{\neg f1\}$, sensor 1 fault (stuck at low) = false,
$IVC_2 = \{\neg f2\}$, sensor 2 fault (stuck at low) = false,
$IVC_3 = \{\neg f3\}$, sensor 3 fault (stuck at low) = false.

MUSes are equivalent to the IVCs.
$MUS_1 = \{\neg f1\}$,
$MUS_2 = \{\neg f2\}$,
$MUS_3 = \{\neg f3\}$.

Now we look at Maximal Satisfiable Subsets (MSS). MSSs are the sets for which we have the maximal number of elements that will prove our constraint system. If we add anything to these sets, it becomes UNSAT. Since our original constraint system is of the form $C = \{\neg f1, \neg f2, \neg f3, \neg P\}$, we have:
MUS: P is SAT $\iff$ $\neg P$ is UNSAT
MSS: P is UNSAT $\iff$ $\neg P$ is SAT.

The Minimal Correction Sets (MCSs) are the complement of MSS relative to constraint system C. The MCSs describes the infeasibility of the system and works as a "correction" set to the problem. The MCSs describe the constraints that when removed from the constraint system, provide a satisfiable system. Furthermore, any strict subset of the MCSs, when removed from the constraint system, will provide an unsatisfiable system. It seems that we do not need to actually find the $MSSs$ in order to get their complement because of what is called a *hitting set*. Intuitively, a minimal hitting set

has the minimal number of elements in it such that every set in that collection has something in common with the set its "hitting." Every MCS is a minimal hitting set of its MUSes.

For us in this example, it is: MCS $= \{\neg f1, \neg f2, \neg f3\}$. This is the hitting set because if we take the intersection of every MUS with the MCS, it is nonempty and it is the minimal such set for which this is true. When the constraints on these model elements are removed, we get a constraint system that is satisfiable with regard to $\neg P$. Thus, this is the minimal set of elements for which the top level property occurs. What this means from the models perspective is that all three faults together cause the top level property to fail. Thus, when all three sensors have a fault which causes them to report low temp, we do not shut down when we should. Intuitively, it makes sense that this is the Minimal Cut Set because it is the minimal description of why the top level property fails.

1.4 Replacement of a Contract with its MinCutSets for Max N Faults Analysis

At the leaf level, only faults are contained in IVCs (and consequently in MCSs). Thus we store these cut sets in a lookup table for quick access throughout the algorithm. For this algorithm, we assume we are in an intermediate level of the analysis. Given a fault hypothesis of max N faults, we disregard any cut sets over cardinality N and collect the rest.

Assuming we have used the hitting set to generate MCS from the IVC, this is where the algorithm begins.

Let $MCS = F \cup G$ for faults $f \in F$ and contracts $g \in G$. If $|G| = 0$, then add MCS to contract lookup table (it is already a MinCutSet).

Assume $|G| = \alpha > 0$.

Let $Cut(g_j) = cut_1(g_j), ..., cut_\beta(g_j)$ be all minimal cut sets for a contract $g_j \in G$ where $|Cut(g_j)| = \beta_j$.

Algorithm 1: Replacement

```
List(MCS) = MCS : initialize list of MCSs that contain contracts ;
Cut(g_j) : all minimal cut sets with cardinality less than or equal to N of the
  contract g_j ;
0 < j ≤ α ;
0 < i ≤ γ ;
for all MCS_i ∈ List(MCS) do
    Remove MCS_i from List(MCS);
    for all g_j ∈ MCS_i do
        Remove g_j from MCS_i ;
        for all cut_k(g_j) ∈ Cut(g_j) do
            Add cut_k(g_j) to MCS_i ;
            if |MCS_i| ≤ N then
                if ∃g_j ∈ MCS_i then
                    Add MCS_i to List(MCS)
                else
                    Add MCS_i to contract look up table (done)
```

The number of replacements R that are made in this algorithm are given as the combination of minimal cut sets of the contracts within the MCS. The validity of this statement follows directly from the general multiplicative combinatorial principle. Therefore, the number of replacements R is given by:

$$R = \prod_{j=1}^{\alpha} |Cut(g_j)|$$

This is equivalent to the number of minimal cut sets generated which follows easily from the combinatorial calculation. It is also important to note that the cardinality of $List(MCS)$ is bounded. Every new MCS that is generated that still contains contracts is added to $List(MCS)$. Thus every contract up until the penultimate contract contributes to the cardinality of $List(MCS)$. The bound is given by:

$$|List(MCS)| \leq \prod_{j=1}^{\alpha-1} |Cut(g_j)|$$

Theorem 1. *The elimination of any $|MCS_i| \leq N$ will not eliminate any $|MinCutSet| \leq N$.*

Proof. If $!\exists g_j \in MCS_i$, then $MCS_i = MinCutSet$ by proof previously given. Thus, this cut set is not required for consideration in the analysis since $|MinCutSet| > N$.

If $\exists g_j \in MCS_i$, then further replacement will need to be made. Thus in the next phase of the algorithm, we will have MCS_{i+1}. In this case, the size can only get larger and $|MCS_i| \leq |MCS_{i+1}|$ and we have not eliminated a cut set that must be

considered for the analysis.

A few thoughts...

- All elements in $List(MCS)$ are "ORed" together.
- All elements in MCS_i are "ANDed" together.
- When no more contracts exist in MCS_i, place that list in the data table and NOT back into $List(MCS)$.
- Need to create a quick way to check for contract existence in an MCS_i. Either a smart method to call or a good data structure (probably both).

References

1. E. Ghassabani, A. Gacek, and M. W. Whalen. Efficient generation of inductive validity cores for safety properties. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 314–325. ACM, 2016.
2. E. Ghassabani, M. Whalen, and A. Gacek. Efficient generation of all minimal inductive validity cores. In *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design*, pages 31–38. FMCAD Inc, 2017.
3. M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva. Fast, flexible mus enumeration. *Constraints*, 21(2):223–250, 2016.
4. R. Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.