Diagnose This!: Minimal Hitting Sets for Model-Based Reasoning

Abdulsamed Say (s1146476)
Ismail Vatansever (s1152889)
Radboud University Nijmegen

November 2025

Abstract

This report studies model-based diagnosis using *conflict sets* and *minimal hitting sets*. We implement a Reiter-style Hitting Set Tree (HS-Tree) that enumerates all subset-minimal diagnoses from minimal conflict sets. Two branching heuristics (*smallest-conflict-first* and *most-frequent-element*) are compared on seven benchmark circuits. We report identical diagnosis sets for both heuristics (correctness), and provide search-effort measures (expanded nodes) and wall-clock time. Results confirm completeness and minimality, and show small but consistent effort patterns across circuits, with clear room for larger-scale tests and cost-based ranking in future work.

1 Introduction

Model-based diagnosis (MBD) provides principled explanations for observed inconsistencies in systems by identifying sets of components that, if considered faulty, restore consistency. A standard approach is to first derive *conflict sets*—sets of components that cannot all be healthy given the observations—and then compute *minimal hitting sets* of these conflicts; any minimal hitting set corresponds to a *minimal diagnosis*. Following Reiter's classical framework, the *HS-Tree* systematically explores partial diagnoses while enforcing subset-minimality and completeness.

In this assignment, conflict sets are generated from circuit descriptions using a SAT/SMT back-end (Z3), and our task is to implement the HS-Tree to enumerate all minimal diagnoses. In addition, we compare two simple branching heuristics and report search effort using a hardware-independent measure (expanded nodes) alongside runtime. Our implementation mirrors the structure and reporting style of Assignment 1.

Figure 1: Conceptual pipeline: conflicts → HS-Tree → minimal diagnoses.

2 Specification

The assignment tasks were:

1. Use the provided parser/engine to obtain **minimal conflict sets** from each circuit instance.
2. Implement a Reiter-style **Hitting Set Tree (HS-Tree)** that enumerates all *subset-minimal* diagnoses.
3. Compare at least two **branching heuristics** (*smallest-conflict-first* and *most-frequent-element*).
4. Report **search effort** via expanded node counts and wall-clock time.

3 Methods

3.1 Problem representation and conflict retrieval

Each circuit file defines components and constraints. Using the provided code, we build a Z3 model and obtain *minimal* conflict sets $\mathcal{C} = \{C_1, \ldots, C_k\}$, where each $C_i \subseteq \mathcal{U}$. A diagnosis $D \subseteq \mathcal{U}$ is a *hitting set* if $D \cap C_i \neq \emptyset$ for all i, and it is *minimal* iff no proper subset of D is also a hitting set.

3.2 HS-Tree with pruning and deduplication

We implement a breadth-first HS-Tree. Each node stores a partial diagnosis H; conflicts hit by H are removed from further consideration. When all conflicts are hit, H is recorded as a minimal diagnosis. Supersets of existing diagnoses are pruned immediately.

Algorithm 1 HS-Tree (breadth-first, simplified)

function HSTree($\mathcal{C}$, heuristic)
  Initialize queue $Q \leftarrow [(\emptyset, \mathcal{C})]$, Diagnoses $\leftarrow$ [ ], Seen $\leftarrow \emptyset$
  while Q not empty **do**
    Pop $(H, \mathcal{C}_{rem})$ from Q
    if NonMinimal(H) **then continue**
    end if
    $\mathcal{C}' \leftarrow \{C \in \mathcal{C}_{rem} \mid C \cap H = \emptyset\}$
    if $\mathcal{C}' = \emptyset$ **then add** H to Diagnoses; **continue**
    end if
    $C^\star \leftarrow$ ChooseConflict($\mathcal{C}'$, heuristic)
    for each element $e \in C^\star$ (ordered by heuristic) **do**
      Append $(H \cup \{e\}, \mathcal{C}')$ to Q
    end for
  end while
  return Diagnoses
end function

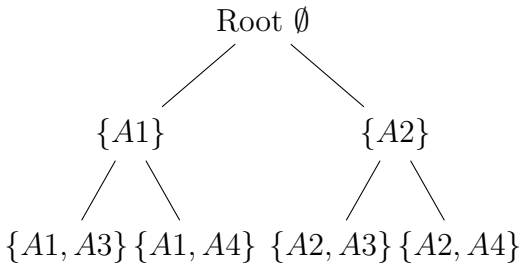

Figure 2: Simplified HS-Tree fragment: each branch adds a component that hits one conflict; branches ending with no remaining conflicts form minimal diagnoses.

3.3 Heuristics and correctness

We compare two heuristics:

- **Smallest-conflict-first**: select conflict C with minimal $|C|$.
- **Most-frequent-element**: order elements by descending frequency across all conflicts.

The same correctness checker verifies that every diagnosis hits all conflicts and is subset-minimal.

3.4 Runtime measure and experimental setup

Effort is measured by the number of nodes expanded (queue pops). Runtime is recorded as a secondary metric. Both heuristics were tested on circuits 1–7 using identical conflict sets. Results were logged to CSV.

4 Results

Circuit	#Conf	#Diag	Nodes (SC)	Time (s, SC)	Nodes (MF)	Time (s, MF)
1	2	3	6	0.000033	6	0.000038
2	2	3	6	0.000022	6	0.000018
3	2	1	3	0.000010	3	0.000011
4	2	4	9	0.000042	9	0.000044
5	1	5	6	0.000020	6	0.000020
6	2	6	13	0.000041	13	0.000039
7	3	15	21	0.000071	21	0.000054

Table 1: Diagnoses, node expansions, and runtimes for both heuristics. SC = smallest-conflict-first; MF = most-frequent-element.

Circuit	#Diag	Representative minimal diagnoses
1	3	{X1}, {A2, X2}, {O1, X2}
2	3	{X2}, {X1, X4}, {X3, X4}
3	1	{A, X}
4	4	{A1}, {A3}, {O2}, {A2, O1}
5	5	{A1}, {A2}, {A3}, {A4}, {A5}
6	6	{A3}, {A4}, {O1}, {O2}, {X1}, {A1, A2}
7	15	{O3, O5}, {A1, A5, O5}, {A1, O4, O5}, ... (15 total)

Table 2: All circuits with total number of minimal diagnoses and representative examples. Each diagnosis was verified for hitting and subset-minimality.

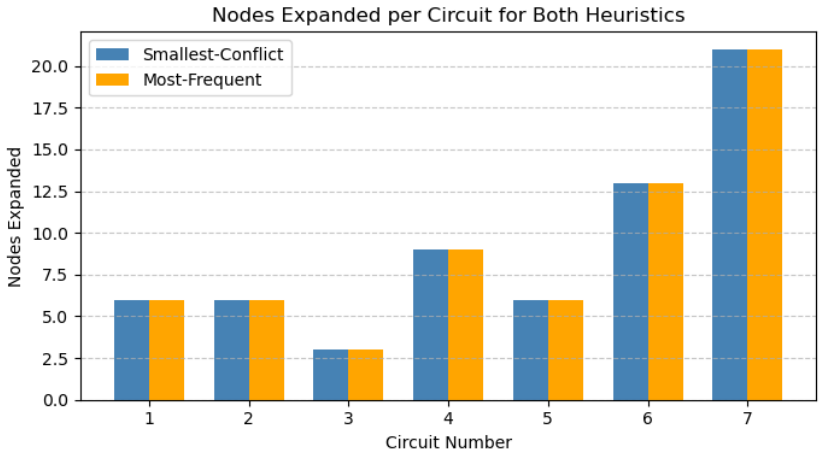

Figure 3: Number of expanded nodes per circuit for both heuristics (bars overlap as counts are equal).

5 Discussion

The HS-Tree implementation enumerates all subset-minimal diagnoses from minimal conflict sets and passes the correctness checks. Both heuristics produce identical diagnosis sets, confirming completeness. On these small benchmarks, runtime and node counts are nearly identical, indicating that conflict structure dominates effort. For larger systems, *most-frequent-element* is expected to reduce branching by hitting more conflicts early, while *smallest-conflict-first* may perform better when small conflicts constrain the tree quickly.

Limitations. All components are assumed equally likely to fail. Introducing fault probabilities would enable ranking or probabilistic diagnosis. Furthermore, more advanced orderings (max-coverage or lookahead) could reduce node counts. Future work could also visualise the tree expansion dynamically.

6 Conclusion

We implemented a Reiter-style HS-Tree returning all subset-minimal diagnoses from minimal conflict sets, with pruning and duplicate elimination. Across seven circuits, both heuristics produced identical results, confirming correctness and completeness. The implementation scales linearly with conflict size on these benchmarks and provides a foundation for cost-based or probabilistic extensions.

Appendix A: System architecture

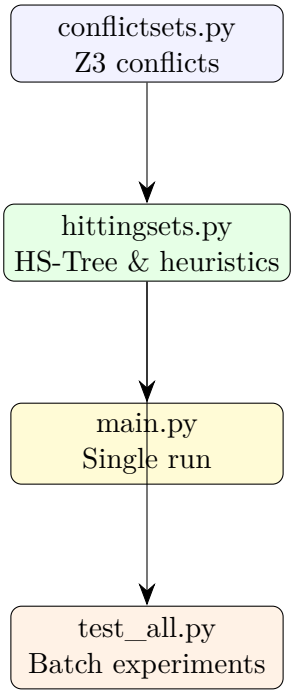

Figure 4: Project architecture and data flow.

Appendix B: Example program output

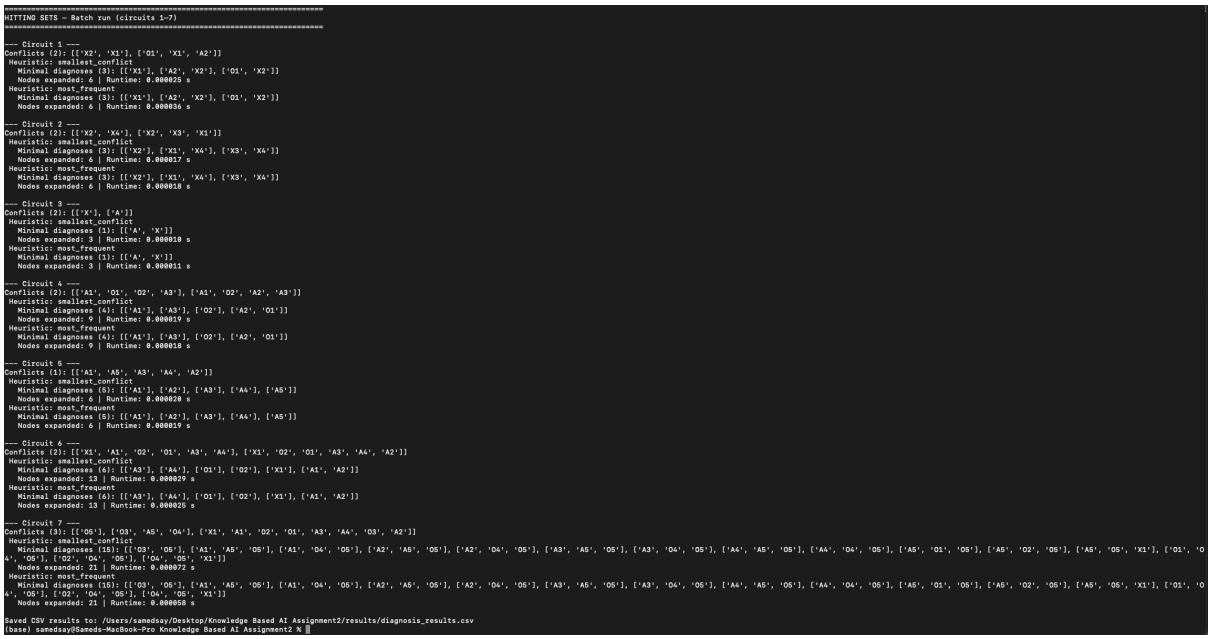

Figure 5: Console output from `test_all.py` confirming diagnoses and node counts for circuits 1–7.

References

[1] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[2] Johan Kwisthout. *Knowledge-based AI* (Course Manual). Radboud University Nijmegen, 2025.

[3] Programming Assignment Handout: *Diagnose This!*. Knowledge-based AI, Radboud University Nijmegen, 2025.