13th International Conference on Applied Informatics
Eszterházy Károly Catholic University
Eger, Hungary, February 19–21, 2026

Formal Verification of Questionnaire Logic Using SMT Solvers

Peter Saghelyi[a], **Peter Berecky**[a]

[a]ELTE Faculty of Informatics, Budapest, Hungary
peter.saghelyi@inf.elte.hu

Abstract

Modern questionnaire systems, particularly those enhanced with artificial intelligence, can generate surveys with conditional logic of unprecedented complexity. While traditional manually-designed surveys remain manageable for human review, AI-generated questionnaires present challenges including exponential path proliferation, hidden logical contradictions, and circular dependencies that exceed human comprehension. Despite decades of active research on questionnaire structure and validation [1–3, 6, 7], prior approaches have focused on documentation, visualization, and path enumeration—none providing a direct method for formally proving questionnaire correctness. This work presents a formal, declarative framework for questionnaire specification and automated verification using SMT solvers, offering mathematical guarantees that testing-based approaches cannot provide.

1. Formal Questionnaire Model

We formalize questionnaires using preconditions and postconditions as logical formulas, inspired by Hoare Logic [4]. This declarative paradigm treats questionnaires as collections of logical relationships where flow emerges dynamically based on respondent answers.

Definition 1. A **questionnaire** $\mathcal{G}$ is a tuple $(\mathcal{I}, \mathbf{S}, \mathcal{D}, \mathcal{P}, \mathcal{Q}, I_{\text{start}})$ where:

- $\mathcal{I} = \{I_1, \ldots, I_n\}$ is a finite set of items (a.k.a. questions)
- $\mathbf{S} = (S_1, \ldots, S_n)$ is a vector of outcome variables representing the actual responses

- $\mathcal{D} = (D_1, \ldots, D_n)$ specifies domain constraints for each S_i
- $\mathcal{P} = (P_1, \ldots, P_n)$ where P_i is the precondition (Boolean formula determining item visibility)
- $\mathcal{Q} = (Q_1, \ldots, Q_n)$ where Q_i is the postcondition (constraint on valid responses)
- $I_{\text{start}} \in \mathcal{I}$ is the designated starting item

Let $B := \bigwedge_{i=1}^{n} D_i(S_i)$ denote the base constraint. We classify precondition reachability as: **ALWAYS** if $\text{UNSAT}(B \wedge \neg P_i)$, **NEVER** if $\text{UNSAT}(B \wedge P_i)$, and **CONDITIONAL** otherwise. Postconditions are classified relative to their preconditions as **TAUTOLOGICAL**, **CONSTRAINING**, or **INFEASIBLE**.

2. Comprehensive Validation

To prove that a questionnaire is correctly designed and well-formatted, we examine four critical properties that capture common design errors.

Isolated Questions. An item is isolated when its precondition can never be satisfied under any combination of valid answers. This occurs when the logical conditions for displaying a question are internally contradictory or conflict with domain constraints. For example, a question requiring both $age < 18$ and $years\_employed > 20$ can never be reached. We detect isolation by checking whether $B \wedge P_i$ is satisfiable, where B represents domain constraints and P_i the precondition.

Dead Ends. Even when per-item analysis shows a question as reachable (i.e. not isolated), accumulated constraints from predecessor items may render it unreachable in practice. Consider a questionnaire where the first question requires $income \geq 50000$ to proceed, while a later question targets respondents with $income < 30000$. Each condition is individually satisfiable, but no respondent can satisfy both. We detect dead ends through path-based validation, checking reachability under accumulated postconditions from all predecessor items in dependency order.

Circular Dependencies. When item A's precondition references item B's outcome, and B's precondition references A's outcome, neither can be evaluated first. We construct a dependency graph where edges represent data flow between items through outcome variables and intermediate computations. Kahn's algorithm for topological sorting detects cycles: if edges remain after the algorithm terminates, those edges form circular dependencies that make the questionnaire impossible to evaluate.

Contradictory Logical Constraints. Postconditions from multiple items may collectively create unsatisfiable requirements. We define the global satisfiability formula $\mathcal{F} := B \wedge \bigwedge_{i=1}^{n}(P_i \Rightarrow Q_i)$ and verify that at least one valid completion exists. If $\mathcal{F}$ is unsatisfiable, the questionnaire contains conflicting constraints that prevent any respondent from completing it successfully.

3. The QML Language and SMT-Based Verification

We introduce the Questionnaire Markup Language (QML), a YAML-based specification language where preconditions and postconditions are expressed as Python boolean expressions, and items may contain imperative code blocks for intermediate computations. Our compiler transforms these specifications into Z3 SMT constraints through Single Static Assignment (SSA) transformation, conditional branching transformation, and bounded loop unrolling.

What distinguishes this approach is the unified analysis of both declarative conditions and imperative code blocks. Variables introduced in code blocks flow through the dependency graph alongside outcome variables, enabling the SMT solver to reason about the complete questionnaire logic. Despite questionnaire consistency checking being NP-complete [8], modern SMT solvers handle real-world questionnaires effectively, providing mathematical guarantees that testing-based approaches cannot offer [5].

References

[1] D. Elliott: *The Application of Graph Theory to the Development and Testing of Survey Instruments*, International Journal of Market Research 54.4 (2012), pp. 545–570, doi: 10.2501/IJMR-54-4-545-570.

[2] J. T. Fagan, B. V. Greenberg: *Using Graph Theory to Analyze Skip Patterns in Questionnaires*, in: Proceedings of the Section on Survey Research Methods, American Statistical Association, 1988, pp. 193–198.

[3] I. P. Fellegi, D. Holt: *A Systematic Approach to Automatic Edit and Imputation*, Journal of the American Statistical Association 71.353 (1976), pp. 17–35, doi: 10.1080/01621459.1976.10481472.

[4] C. A. R. Hoare: *An Axiomatic Basis for Computer Programming*, Communications of the ACM 12.10 (1969), pp. 576–580, doi: 10.1145/363235.363259.

[5] L. de Moura, N. Bjørner: *Z3: An Efficient SMT Solver*, in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), vol. 4963, Lecture Notes in Computer Science, Springer, 2008, pp. 337–340, doi: 10.1007/978-3-540-78800-3_24.

[6] I. Schiopu-Kratina, P. Lavallée, L.-P. Rivest: *Survey Questionnaires and Graphs*, Methodology: European Journal of Research Methods for the Behavioral and Social Sciences 11.4 (2015), pp. 128–138, doi: 10.1027/1614-2241/a000100.

[7] L. Willenborg: *Computational Aspects of Survey Data Processing*, PhD thesis, University of Amsterdam, 1988.

[8] L. Willenborg: *Testing and Checking Questionnaires*, Journal of Official Statistics 11.3 (1995), pp. 299–317.