Algorithmic Detection of Boolean Logic Errors in Clinical Decision Support Statements

Adam Wright[1,2,3,4] Skye Aaron[2] Allison B. McCoy[1] Robert El-Kareh[5] Daniel Fort[6] Steven Z. Kassakian[7] Christopher A. Longhurst[5] Sameer Malhotra[8,9] Dustin S. McEvoy[4] Craig B. Monsen[10] Richard Schreiber[11] Asli O. Weitkamp[1] DuWayne L. Willett[12] Dean F. Sittig[13]

[1] Department of Biomedical Informatics, Vanderbilt University Medical Center, Nashville, Tennessee, United States
[2] Division of General Internal Medicine, Brigham and Women's Hospital, Boston, Massachusetts, United States
[3] Department of Biomedical Informatics, Harvard Medical School, Boston, Massachusetts, United States
[4] Partners eCare, Partners HealthCare System, Boston, Massachusetts, United States
[5] Department of Medicine, UC San Diego Health, University of California, San Diego, San Diego, California, United States
[6] Center for Outcomes and Health Services Research, Ochsner Health System, New Orleans, Louisiana, United States
[7] Department of Medical Informatics and Clinical Epidemiology, Oregon Health & Science University, Portland, Oregon, United States
[8] Department of Healthcare Policy and Research, Weill Cornell Medicine, New York, New York, United States
[9] Department of Internal Medicine, NewYork-Presbyterian Hospital, New York, New York, United States
[10] Center for Informatics, Atrius Health, Boston, Massachusetts, United States
[11] Physician Informatics and Department of Internal Medicine, Geisinger Holy Spirit, Camp Hill, Pennsylvania, United States
[12] Department of Internal Medicine, University of Texas Southwestern Medical Center, Dallas, Texas, United States
[13] School of Biomedical Informatics, University of Texas Health Science Center at Houston, Houston, Texas, United States

Address for correspondence Adam Wright, PhD, Department of Biomedical Informatics, Vanderbilt University Medical Center, 2525 West End, Nashville, TN 37203, United States (e-mail: adam.wright@vumc.org).

Appl Clin Inform 2021;12:182–189.

Abstract

Objective Clinical decision support (CDS) can contribute to quality and safety. Prior work has shown that errors in CDS systems are common and can lead to unintended consequences. Many CDS systems use Boolean logic, which can be difficult for CDS analysts to specify accurately. We set out to determine the prevalence of certain types of Boolean logic errors in CDS statements.

Methods Nine health care organizations extracted Boolean logic statements from their Epic electronic health record (EHR). We developed an open-source software tool, which implemented the Espresso logic minimization algorithm, to identify three classes of logic errors.

Results Participating organizations submitted 260,698 logic statements, of which 44,890 were minimized by Espresso. We found errors in 209 of them. Every

Keywords

- clinical decision support
- electronic health records and systems
- alerting
- decision support algorithm
- efficiency improvement

received
August 11, 2020
accepted after revision
December 16, 2020

DOI https://doi.org/10.1055/s-0041-1722918.
ISSN 1869-0327.

participating organization had at least two errors, and all organizations reported that they would act on the feedback.

Discussion An automated algorithm can readily detect specific categories of Boolean CDS logic errors. These errors represent a minority of CDS errors, but very likely require correction to avoid patient safety issues. This process found only a few errors at each site, but the problem appears to be widespread, affecting all participating organizations.

Conclusion Both CDS implementers and EHR vendors should consider implementing similar algorithms as part of the CDS authoring process to reduce the number of errors in their CDS interventions.

Background and Significance

When properly implemented, clinical decision support (CDS) systems contribute to improvements in quality and safety.[1–5] However, in recent work, our research team identified a large number of implementation errors and technical malfunctions in CDS systems.[6–10] Causes for these errors vary, and include improper definition of value sets, terminology changes, and system upgrades.[7,9,10]

Many CDS systems allow for the construction of rules from logical terms, linked through a Boolean logic statement.[11,12] ►Fig. 1 shows a representative example of a decision support rule in this form. This logic statement represents the core of the United States Centers for Disease Control and Prevention Advisory Committee on Immunization Practices recommendation on influenza vaccination,[13] and is similar to the form used in many electronic health record (EHR) systems. If desired, some terms could be defined further (for example, the second term could have a nested definition with its own logic statement, for example, "patient allergic to eggs OR patient has a fever"). The implementer also has the freedom to vary the construction. For example, in this case, the second term is phrased in the negative, but the term could be in the positive (patient has a contraindication to influenza vaccination), in which case it would be necessary to negate this term in the logic statement, as was done with the third statement.

Boolean logic, named after the mathematician George Boole (1815–1864), is a logical algebra which evaluates all statements, connected through logical operators like "and," "or," and "not," as true or false. A variety of fields use Boolean logic, including computer science, electrical engineering, logic, mathematics, and information retrieval, to create unambiguous and precisely specified statements of logic.[14] Boolean logic statements are inherently computable. Logic gates or microprocessors can evaluate these statements, so they are an attractive (and likely necessary) form of knowledge representation for computer-based CDS. However, humans frequently struggle with the task of accurately translating intended clinical rules into Boolean logic statements.[15–17]

Consistent with this experience, our own research suggests that errors in Boolean logic are a recurring cause of CDS malfunctions.[9] Although many of these errors have complex etiologies and are often difficult to identify, a subset of them involve logic statements that are objectively defective and which are algorithmically detectable. Three categories with example statements are:

- **Always True**
  Example: **1 or not 1**. This statement is a tautology. It is always true and does not depend on the value of term 1. *A hypothetical clinical example would be "Patient has diabetes or patient does not have diabetes."*
- **Always False**
  Example: **1 and not 1**. This statement is always false and represents a logic contradiction. *A hypothetical clinical example would be "Patient has diabetes and patient does not have diabetes."*
- **Absorption**
  Example: **1 or (1 and 2).** This statement contains a Boolean absorption and is thus logically redundant. It is true when term 1 is true and false when term 1 is false, and does not depend on the value of term 2.[18] *A hypothetical clinical example would be "Patient has diabetes or patient has both diabetes and hypertension," which really just identifies patients with diabetes, regardless of whether they have hypertension.*

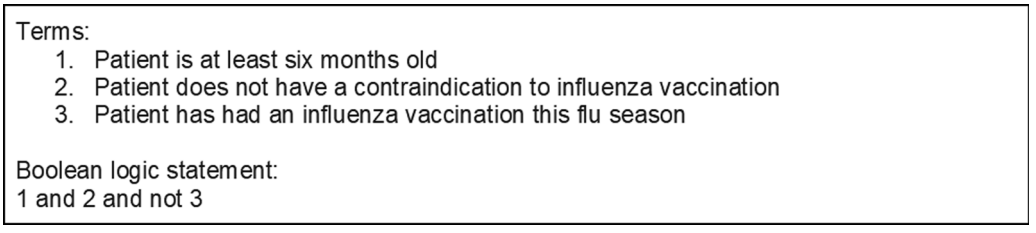

Fig. 1 Representative example of clinical decision support (CDS) logic for an influenza vaccination reminder.

Table 1 Participating sites

Site	Location	Number of logic statements extracted
Atrius Health	Boston, MA	23,169
Geisinger Health	Danville, PA	24,007
Ochsner Health System	New Orleans, LA	5,961
Oregon Health & Science University	Portland, OR	35,090
Partners HealthCare	Boston, MA	49,524
University of California San Diego	San Diego, CA	42,092
UT Southwestern Medical Center	Dallas, TX	27,447
Vanderbilt University Medical Center	Nashville, TN	29,233
Weill Cornell Medicine	New York, NY	24,175

In this article, we develop and evaluate logic minimization and evaluation software to identify these types of objective logic errors in real-world decision support knowledge bases from nine health care provider organizations across the United States.

Objectives

CDS can contribute to quality and safety. Prior work has shown that errors in CDS systems are common and can lead to unintended consequences. Many CDS systems use Boolean logic, which can be difficult for users to specify accurately. We set out to determine the prevalence of certain types of Boolean logic errors in CDS knowledge bases.

Methods

We identified nine members of the Clinical Informatics Research Collaborative (CIRCLE)[19] representing diverse healthcare provider organizations across the United States that used an EHR developed by Epic Systems (Verona, Wisconsin, United States), as shown in ►**Table 1**. We focused on this EHR because it explicitly stores logic statements separate from the control structure of the program, making them readily extractable. Some EHRs implement a domain-specific language for CDS which makes the Boolean logic statement implicit in the control structure of the program (and thus harder to extract). Others limit the depth of conjunction and disjunction to a single layer, limiting the complexity of logic which can be represented but also making it less likely for the author to create these types of defects.

We provided instructions to these organizations about how to extract logic statements from Best Practice Advisories (BPAs; Epic's name for rule-based alerts) and rules (a generic logic concept, used in BPAs and many other system functions) from their production environment.[20] BPAs, and their corresponding logic statements, are developed by CDS authors at each participating site or, in certain cases, by Epic directly. Users who access the Epic BPA authoring tools have Epic-provided training in building CDS and using Epic's tools.

We then developed an open source software tool (available at https://github.com/skyeaaron/LogicMinimizer) which uses the PyEDA[21] Python package and the Espresso heuristic logic minimization algorithm.[22,23] The Espresso algorithm was originally designed to take digital logic circuit designs that consist of interconnected sets of AND, OR, and NOT logic gates and reduces their complexity while keeping the same output. In this case, we apply Espresso to CDS logic statements, which, though implemented in software rather than hardware, can be thought of as equivalent to the ANDs, ORs, and NOTs of digital circuits. Given the generality of Boolean logic, the algorithm readily applies to logic statements from CDS to identify always true, always false, and absorption errors. ►**Fig. 2** shows an illustration of the algorithm. Consider the hypothetical example we gave above "1 OR (1 AND 2)" which corresponds to "patient has diabetes or has both diabetes and hypertension." When the Espresso algorithm operates, it first creates a truth table that represents the statement. It then attempts to minimize the truth table—in this case, observing that the output depends only on Term 1, and not on Term 2—the output of the algorithm is then a minimized logic statement—in this case "1 OR (1 AND 2)" is minimized, simply, to Term 1, which makes Term 2 an absorbed variable. The algorithm ran on a standard personal computer (PC) and took less than 1 hour per organization.

The program categorizes each statement into one of nine categories: "And/Or/Not," "At Least/At Most/Exactly," "Invalid Statement," "Always True," "Always False," "Absorption," "No Redundancy," "Timed Out," or "Non-Zero Exit Code."

The program removes statements the Espresso algorithm cannot process. There were three categories of these statements. First, as an alternative to more complicated Boolean logic statements, Epic allows for single operator logic statements like "and," "or," and "not" which logically conjoin or

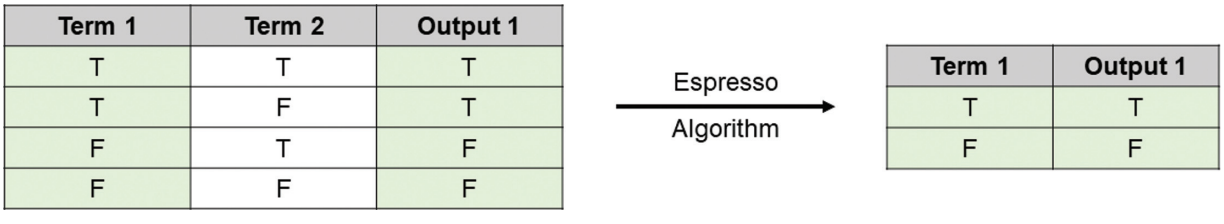

Fig. 2 Example Espresso minimization of the redundant Boolean logic statement: 1 OR (1 AND 2). The left side shows a truth table, which reveals that Output 1 depends only on Term 1—Term 2 is irrelevant to the output. Espresso minimizes this redundant statement to Term 1 only and produces a new truth table (shown on the right) which is equivalent to, but simpler than, the truth table on the left.

disjoin all terms in a statement. These statements cannot contain the Boolean logic errors we are studying, so we filtered them out. Second, Epic allows for quantified statements like "at least 3 terms from the list 1, 2, 3, 4, 5, 6, 7, or 8." This statement evaluates true when at least three of the first eight terms are true. These statements are not part of standard Boolean algebra, and the Espresso algorithm cannot process them, so we excluded them as well. Finally, some statements contained logic that is not possible to parse, such as "1 AND 2 AND AND 4 AND NOT 3" (the presence of "AND AND" makes it impossible to parse this statement). These unparseable statements require correction, even for Epic to properly use them. We excluded them before running the minimization. The Espresso minimization is run on all remaining statements.

In extremely rare cases, the minimization cannot be completed, either because the statement exceeds a user-specified timeout or because of some other error in the logic of the program. These statements are categorized as "Timed Out" (when the statement timed out) or "Non-Zero Exit Code" (when the program failed, but did not provide any detail about the failure mode except for a nonzero exit code). Based on discussions with the implementor of the Espresso algorithm, they appear to represent cases where the logic tree is not able to be properly mapped due to limitations of the algorithm, causing it to endlessly cycle and run out of memory. To ensure that we were not encountering a simple memory capacity limit, we reran these statements on a Linux cluster with 498 gigabytes of memory. The same statements that failed on the office PC also failed on the cluster, confirming that they are not analyzable within reasonable bounds (or likely at all, within the limits of the algorithm implementation).

Results

Participating organizations reported that it was very easy to extract the logic statements following the instructions provided—in most cases, the hardest step was finding an analyst with the proper security permissions—the extraction itself took only seconds to run, and the logic statements were then sent to our research coordinator.

►Fig. 3 shows the overall flow of the study. Sites submitted a total of 260,698 Boolean logic statements. Note that 215,680 were and/or/not statements, which cannot contain the types of redundancies we are studying. Eighty-three contained quantifiers so were not purely Boolean, and 9 contained errors that made them unparseable. Of the remaining 44,926 a total of 36 failed to minimize due to the algorithmic limitation described above. An example of a statement that could not be minimized is:

(1 OR 2 OR 3 OR 4 OR 5 OR 6 OR 7 OR 8 OR 9 OR 10 OR 11 OR 12 OR 13 OR 14 OR 15 OR 16 OR 17) AND (18 OR 19 OR 20 OR 21 OR 22 OR 23 OR 24 OR 25 OR 26 OR 27 OR 28 OR 29 OR 30 OR 31 OR 32 OR 33 OR 34) AND ((18 AND 35 AND 36 AND 37 AND 38) OR (19 AND 39 AND 40 AND 41 AND 42) OR (20 AND

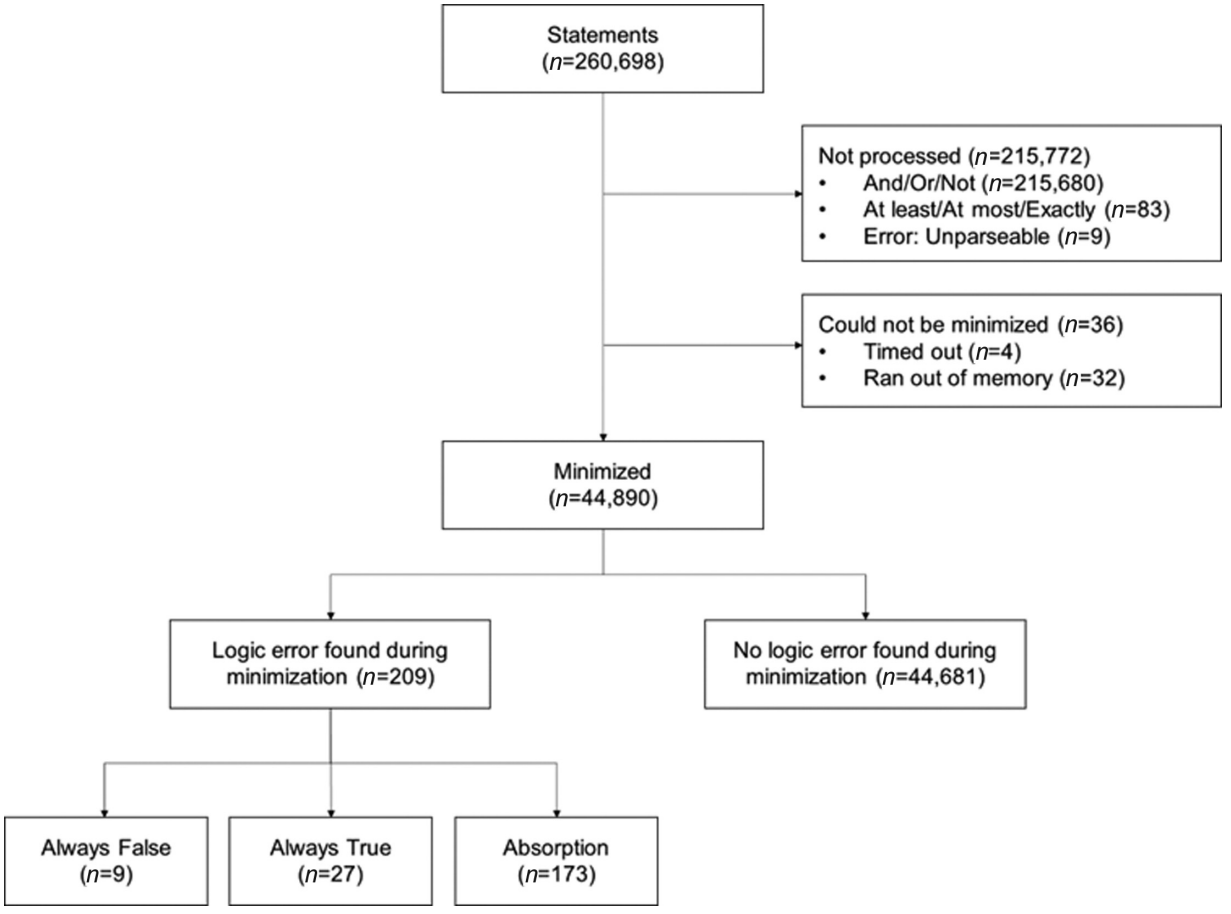

Fig. 3 Processing flow diagram showing types of errors.

Table 2 Logic statement reductions showing examples of errors

	Logic statement	Clinical meaning
Example 1		
Original	1 OR (1 AND 2)	(Patient has no ideal body weight on file) OR (Patient has no ideal body weight on file AND Patient has ideal body weight on file)
Reduced	1	Patient has no ideal body weight on file (result depends only on element 1; the value of element 2 is irrelevant)
Example 2		
Original	1 OR (1 AND 2 AND 3)	(Ambulatory referral order to care management) OR (Ambulatory referral order to care management AND Referral order scheduling status is not scheduled, expired, appointment canceled, unable to schedule AND Referral order does not have a Misc Flag)
Reduced	1	Ambulatory referral order to care management (result depends only on element 1; the values of elements 2 or 3 are irrelevant)
Example 3		
Original	1 AND (NOT 1)	The clinical encounter is an inpatient encounter AND the clinical encounter is not an inpatient encounter
Reduced	FALSE	Always false (result does not depend on the value of its only input, 1)

43 AND 44 AND 45 AND 46) OR (21 AND 47 AND 48 AND 49 AND 50) OR (22 AND 51 AND 52 AND 53 AND 54) OR (23 AND 55 AND 56 AND 57 AND 58) OR (24 AND 59 AND 60 AND 61 AND 62) OR (25 AND 63 AND 64 AND 65 AND 66) OR (26 AND 67 AND 68 AND 69 AND 70) OR (27 AND 71 AND 72 AND 73 AND 74) OR (28 AND 75 AND 76 AND 77 AND 78) OR (29 AND 79 AND 80 AND 81 AND 82) OR (30 AND 83 AND 84 AND 85 AND 86) OR (31 AND 87 AND 88 AND 89 AND 90) OR (32 AND 91 AND 92 AND 93 AND 94) OR (33 AND 95 AND 96 AND 97 AND 98) OR (34 AND 99 AND 100 AND 101 AND 102))

Of the 44,890 logic statements successfully minimized by Espresso, there were 209 instances of errors, representing 0.08% of all statements submitted and 0.5% of minimized logic statements. Nine statements were always false, 27 statements were always true, and 173 statements had an absorption. ►Table 2 shows three examples of errors that were identified.

Every participating site had at least two Boolean logic errors found during logic minimization, and the number of errors ranged from 2 to 51. We sent a report identifying the logical errors following our analysis to each site. All of the sites stated that the report was accurate and useful, and that they planned to take action based on the report. None of the sites reported prior awareness of any of the errors found by the system.

Discussion

This study is the first to describe an algorithm for identifying an important class of CDS logic errors. Though uncommon, detectable errors were widespread, occurring at each study site. These logic errors may result in incorrect alerts displaying to users (particularly likely for logic statements which are always true), or not displaying when they should (likely when logic statements are always false). Redundant logic statements are also a marker of potential errors. In redundant statements, not all the included clinical elements affect the result of the evaluation—if the logic statement developer intended the unused elements to affect the result, it is likely that the redundant statement does not reflect his or her intent. These types of errors could lead CDS to fire in cases where it should not, or not fire in cases where it should, which could lead users to make an error (or omission) that could lead to patient harm[9,24] and contribute to alert fatigue.[25,26]

We developed a portable, effective software tool to identify these classes of logic errors. The results are easily interpretable and actionable. Healthcare organizations using Boolean logic for implementing CDS alerts should run these algorithms to detect potential errors. EHR developers that support Boolean logic should consider implementing these, or similar, algorithms within their knowledge authoring environments as "linters" (i.e., analysis tools that can identify likely or possible programming errors) to help logic developers identify potential errors before deploying the logic in the clinical setting.[27] These algorithms could also be implemented in EHR-agnostic authoring tools, such as CDS Connect.[28]

It is intriguing that all sites had demonstrable errors, yet none were aware of them. In our previous work on CDS anomalies, we learned that clinicians may not be aware of such errors[6] and showed that many types of build and conceptualization errors were common.[9] We should note all of the sites had robust CDS development processes, performed by experienced clinicians, informaticians, and programmers who were also responsible for conducting rigorous testing prior to release. None of these errors were identified. It may be beneficial to include logic analysis periodically and prospectively using such tools to assess CDS accuracy rather than wait for someone to detect a problem or worse, a patient to be harmed.

Limitations

Our study has several limitations. First, we limited the study to organizations using Epic Systems' EHRs. Since most EHRs use some form of Boolean logic for CDS, it should be possible to use the same algorithm and program we developed to

analyze logic from any EHR. The program requires logic statements in a general format like "1 and 2 or 3," so it is not specific to Epic. However, one would need to build queries to extract the logic statements from other EHRs. For EHRs that explicitly code their Boolean rule logic, these queries are likely to be simple. However, some EHRs use a domain-specific language, where the control flow of the program first needs analysis to then model it as a Boolean logic statement—this is more complex, but algorithms for such transformations do exist.[29–31] Second, our program only detects a subclass of logic errors, though that subclass consists of statements that are clearly and unambiguously erroneous. Our experience suggests that other types of logic errors are also prevalent. For example, an alert that searched for patients with "most recent Body Mass Index (BMI) < 25" AND "most recent BMI > 30" would also be unsatisfiable but would not be detected by our algorithm. The class of Boolean logic statement errors detectable through our software represents a relatively small fraction of all CDS logic errors. However, these errors can be readily detected through simple algorithmic checks, using open source software, and almost always deserve further analysis and correction, so we believe that flagging them is highly useful.

Third, we were not able to evaluate the effect of these logic errors on patient safety or healthcare quality. To simplify the rule collection procedure and only obtain the minimum necessary information to conduct the study, each site submitted the logic statements used in their CDS, but not the details of the corresponding CDS interventions themselves, nor did we collect data on alert firing rates or patient impact. Finally, we did not collect data on sites' preexisting testing or review processed for Boolean logic statements, so we were not able to assess how our method corresponded to existing processes.

Future work should look at other EHRs and add additional heuristics and methods for detecting potential logic errors, testing logic statements, and facilitating the development of accurate logic statements to detect and prevent other categories of logic errors, with this form of Boolean verification as one of several checks. Other types of logic errors may be detectable through additional automated means (for example, an alert that specifies a condition like "patient is under 18 and is over 65"), while others may be difficult to detect without human domain knowledge ("patient is pregnant and has prostatitis").

Conclusion

We implemented an algorithm capable of identifying three specific types of logic errors in CDS statements. An evaluation of logic statements from nine health care organizations found errors in statements submitted by all participants. Both CDS implementers and EHR vendors should consider implementing similar algorithms to reduce the number of errors in their CDS interventions. Proactively detecting and fixing errors, even minor ones, helps make the EHR more reliable and trustworthy—if clinicians find errors, they distrust and lose faith in computer-generated advice or recommendations.[32] To maintain their trust and respect, CDS implementers must strive to create artifacts that are free from errors. Perfectly functioning CDS places us one step closer to providing the safest and highest quality patient care possible.

Clinical Relevance Statement

Decision support, when working well, can improve quality and safety. This article describes a technique for finding a particular class of decision support issues where logic is misspecified. It is important to correct these logic errors.

Multiple-Choice Questions

1. Does the logic statement:
   1 AND NOT 1
   Contain a Boolean logic error of the type discussed in this paper?
   a. No.
   b. Yes, it is always true.
   c. Yes, it is always false.
   d. Yes, it has another type of absorption.

   Correct Answer: No statement can be both true and false (this is called the "law of the excluded middle" in philosophy). Thus, this logic statement is a contradiction, and will always evaluate as false, so an alert that used it would never fire. This is one of the types of errors that our algorithm detects.

2. Consider the logic statement:
   1 AND (1 OR 2)
   where term 1 represents "patients with diabetes" and term 2 represents "patients with heart failure." The statement contains a Boolean absorption which would be flagged by the algorithm. In this case, term "2" is absorbed, so the alert will fire whenever term 1 is true and does not consider term 2 at all. If the alert is intended to fire only for patients who have both diabetes and heart failure, what would happen if the error was not corrected?
   a. The alert would fire incorrectly for patients with diabetes who do not also have heart failure.
   b. The alert would fire incorrectly for patients with heart failure who do not also have diabetes.
   c. The alert would fire for all patients.
   d. The alert would never fire.

   Correct Answer: The rule author added two criteria to the alert logic—one that looks to determine whether the patient has heart failure, and one that checks to see if the patient has diabetes. It is reasonable to infer that the builder intended the alert to check both of these—either to find patients with both diabetes and heart failure, or perhaps one or the other of diabetes or heart failure. Regardless of their intent, the logic statement 1 AND (1 OR 2) is true for patients who have diabetes, and false for

patients who do not have diabetes. Heart failure is irrelevant to the logic evaluation. The logic statement can be reduced from "1 AND (1 OR 2)" to "1." Since the CDS author likely intended to include "heart failure" somewhere in the logic, there is a good chance that this represents a logic error that needs to be fixed. This type of error is detected by our algorithm.

Protection of Human and Animal Subjects

Human and/or animal subjects were not included in this project.

Authors' Contributions

A.W. wrote the manuscript and conceived the study design. S.A. wrote the logic minimization program. D.F. acted as a tester for the logic minimization program. All authors provided data. All authors provided critical revisions for important intellectual content.

Funding

Research reported in this publication was supported by the National Library of Medicine of the National Institutes of Health under Award Number R01LM011966. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Conflict of Interest

R.E.-K. reports grants from National Library of Medicine, grants from Gordon and Betty Moore Foundation during the conduct of the study; A.B.M. reports grants from National Library of Medicine, grants from Agency for Healthcare Research & Quality during the conduct of the study; D.S.M., C.B.M., and S.A. each report grants from National Library of Medicine during the conduct of the study; D.F.S. reports grants from National Library of Medicine during the conduct of the study; personal fees from Informatics-Review LLC, outside the submitted work; A.W. reports grants from National Library of Medicine during the conduct of the study; D.F., S.Z.K., C.A.L., S. M., R.S., A.O.W., and D.L.W., each have nothing to disclose.

Acknowledgment

We would like to thank Christopher Drake, the developer of PyEDA, who provided guidance about memory errors in large Boolean statements.

References

1 Kaushal R, Shojania KG, Bates DW. Effects of computerized physician order entry and clinical decision support systems on medication safety: a systematic review. Arch Intern Med 2003; 163(12):1409–1416
2 Garg AX, Adhikari NK, McDonald H, et al. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review. JAMA 2005;293(10): 1223–1238
3 Mishuris RG, Linder JA, Bates DW, Bitton A. Using electronic health record clinical decision support is associated with improved quality of care. Am J Manag Care 2014;20(10):e445–e452
4 Nuckols TK, Smith-Spangler C, Morton SC, et al. The effectiveness of computerized order entry at reducing preventable adverse drug events and medication errors in hospital settings: a systematic review and meta-analysis. Syst Rev 2014;3:56
5 Melton BL. Systematic review of medical informatics-supported medication decision making. Biomed Inform Insights 2017; 9:1178222617697975
6 Wright A, Ash JS, Aaron S, et al. Best practices for preventing malfunctions in rule-based clinical decision support alerts and reminders: results of a Delphi study. Int J Med Inform 2018; 118:78–85
7 Wright A, Wright AP, Aaron S, Sittig DF. Smashing the strict hierarchy: three cases of clinical decision support malfunctions involving carvedilol. J Am Med Inform Assoc 2018;25(11):1552–1555
8 Wright A, Hickman TT, McEvoy D, et al. Methods for detecting malfunctions in clinical decision support systems. Stud Health Technol Inform 2017;245:1385
9 Wright A, Ai A, Ash J, et al. Clinical decision support alert malfunctions: analysis and empirically derived taxonomy. J Am Med Inform Assoc 2018;25(05):496–506
10 Wright A, Hickman TT, McEvoy D, et al. Analysis of clinical decision support system malfunctions: a case series and survey. J Am Med Inform Assoc 2016;23(06):1068–1076
11 McCoy AB, Wright A, Sittig DF. Cross-vendor evaluation of key user-defined clinical decision support capabilities: a scenario-based assessment of certified electronic health records with guidelines for future development. J Am Med Inform Assoc 2015;22(05):1081–1088
12 Kong G, Xu D-L, Yang J-B. Clinical decision support systems: a review on knowledge representation and inference under uncertainties. Intl J Comput Intel Syst 2008;1(02):159–167
13 Grohskopf LA, Sokolow LZ, Broder KR, Walter EB, Fry AM, Jernigan DB. Prevention and control of seasonal influenza with vaccines: recommendations of the Advisory Committee on Immunization Practices-United States, 2018-19 Influenza Season. MMWR Recomm Rep 2018;67(03):1–20
14 Whitesitt JE. Boolean Algebra and Its Applications. Reading, MA: Addison-Wesley Publishing Company, Inc.; 1961
15 Herman GL, Loui MC, Kaczmarczyk L, Zilles C. Describing the what and why of students' difficulties in Boolean logic. ACM Trans Comput Edu 2012;12(01):1–28
16 Pane J, Myers B, eds. Tabular and Textual Methods for Selecting Objects from a Group. IEEE International Symposium on Visual Languages; Seattle, WAIEEE2000
17 Almstrum VL. Investigating student difficulties with mathematical logic. In: Hinchey MG, Dean CN, eds. Teaching and Learning Formal Methods. San Diego, CA: Academic Press; 1996:131–160
18 Grätzer G. Lattice Theory: First Concepts and Distributive Lattices. San Francisco: W. H. Freeman; 1971
19 McCoy AB, Wright A, Sittig DF. Multi-institutional, large-scale, international applied clinical informatics research through the Clinical Informatics Research Collaborative (CIRCLE). Stud Health Technol Inform 2019;264:1730–1731
20 Wright A, Aaron S, Sittig DF. Testing electronic health records in the "production" environment: an essential step in the journey to a safe and effective health care system. J Am Med Inform Assoc 2017;24(01):188–192
21 Drake C. Python Electronic Design Automation. Accessed May 30, 2019 at: https://github.com/cjdrake/pyeda
22 Brayton RK, Sangiovanni-Vincentelli AL, McMullen CT, Hachtel GD. Logic Minimization Algorithms for VLSI Synthesis. Norwell, MA: Kluwer Academic Publishers; 1984
23 Rudell RL, Sangiovanni-Vincentelli AL. Multiple-valued minimization for PLA optimization. IEEE Trans Comput Aided Des Integrated Circ Syst 1987;6(05):727–750
24 Stone EG. Unintended adverse consequences of a clinical decision support system: two cases. J Am Med Inform Assoc 2018;25(05): 564–567

25 Chaparro JD, Hussain C, Lee JA, Hehmeyer J, Nguyen M, Hoffman J. Reducing interruptive alert burden using quality improvement methodology. Appl Clin Inform 2020;11(01):46–58
26 McGreevey JD III, Mallozzi CP, Perkins RM, Shelov E, Schreiber R. Reducing alert burden in electronic health records: state of the art recommendations from four health systems. Appl Clin Inform 2020;11(01):1–12
27 Johnson SC. Lint, a C Program Checker. Computer Science Technical Reports 1978
28 Lomotan EA, Meadows G, Michaels M, Michel JJ, Miller K. To Share is Human! Advancing evidence into practice through a national repository of interoperable clinical decision support. Appl Clin Inform 2020;11(01):112–121
29 Bergeretti J-F, Carre BA. Information-flow and data-flow analysis of while-programs. ACM Trans Prog Lang Syst 1985;7(01):37–61 (TOPLAS)
30 Li D, Endle CM, Murthy S, et al. Modeling and executing electronic health records driven phenotyping algorithms using the NQF Quality Data Model and JBoss® Drools Engine. AMIA Annu Symp Proc 2012;2012:532–541
31 Allen FE, Corporation I, eds. Control Flow Analysis. Proceedings of a Symposium on Compiler Optimization; 1970Urbana-Champaign, IL
32 Van Dort BA, Zheng WY, Baysari MT. Prescriber perceptions of medication-related computerized decision support systems in hospitals: a synthesis of qualitative research. Int J Med Inform 2019;129:285–295