Selecting Test Cases from the Electronic Health Record for Software Testing of Knowledge-Based Clinical Decision Support Systems

Omar A. Usman, MD, MBA[1,2], Connie Oshiro, PhD[1], Justin G. Chambers[1], Samson W. Tu[2], MS, Susana Martins, MD, MSc[1], Amy Robinson, PharmD[1], Mary K. Goldstein, MD, MS[1,2]
[1]VA Palo Alto Health Care System, Palo Alto, CA; [2]Stanford University, Stanford, CA

Abstract

Software testing of knowledge-based clinical decision support systems is challenging, labor intensive, and expensive; yet, testing is necessary since clinical applications have heightened consequences. Thoughtful test case selection improves testing coverage while minimizing testing burden. ATHENA-CDS is a knowledge-based system that provides guideline-based recommendations for chronic medical conditions. Using the ATHENA-CDS diabetes knowledge-base, we demonstrate a generalizable approach for selecting test cases using rules / filters to create a set of paths that mimics the system's logic. Test cases are allocated to paths using a proportion heuristic. Using data from the electronic health record, we found 1,086 cases with glycemic control above target goals. We created a total of 48 filters and 50 unique system paths, which were used to allocate 200 test cases. We show that our method generates a comprehensive set of test cases that provides adequate coverage for the testing of a knowledge-based CDS.

Keywords: Clinical Decision Support, Knowledge-Based Systems, Expert Systems, Test Case Selection, Path Testing, Software Testing, System Testing, Offline Testing, Software Verification

Introduction

Knowledge-based systems are apt for modeling complex problems,[1] which makes them suitable for providing Clinical Decision Support (CDS). ATHENA-CDS is a knowledge-based system, developed jointly by the Department of Veterans Affairs (VA) and Stanford University, that provides guideline-based treatment recommendations for a range of chronic medical conditions (e.g. hypertension and diabetes).[2–6] The system takes data from the Electronic Health Record (EHR) and applies it to encoded clinical practice guidelines to make patient-specific recommendations. ATHENA-CDS aims to increase the quality, consistency, and personalization of medical care by increasing provider adherence to evidence-based medicine.[7] To achieve this goal, developers must ensure the quality and consistency of the system itself through rigorous software testing.

Software testing is challenging and costly;[5,8] yet, testing of CDS is necessary since clinical applications have heightened consequences and missed errors become increasingly expensive over time.[3,9,10] Additionally, physicians are often concerned about real or perceived mistakes in CDS recommendations.[11,12] Incorrect recommendations can lead to loss of confidence in the system.[13,14] Finally, medicine is ever-changing and new evidence often requires changes to the knowledge-base, another potential source for errors.[2] Testing addresses these issues by providing the means to identify pre-deployment errors while also gauging the accuracy of a system.[5,9]

The same complexity that makes knowledge-based systems flexible causes their testing to be difficult.[1,5] The ATHENA-CDS contains hundreds of knowledge frames,[5] each being a potential source for errors.[3,15] Testing of knowledge-based systems has a unique set of challenges that differ from testing of conventional software,[13,14,16] particularly when selecting test cases. Case testing is commonly used for the testing of knowledge-based systems.[16] Test cases play an important role in verifying the system's output against a reference standard; yet, the process for obtaining these cases is ill defined.[5,9,16] If test cases are poorly selected or too sparse, testing may fail to catch errors;[9] whereas, if test cases are too granular and superfluous, testing may be cost or labor prohibitive.[5,13]

Case testing also has specific requirements. Test cases are ideally drawn from a sample of real patients, cover a wide range of problems, locate errors in extreme or critical areas, and also cover commonly occurring cases.[1,9] De novo fabricated cases are problematic since they do not reflect real-world scenarios, are difficult to bound, contain developer bias, ignore the correlation between physiologic properties, and require substantial time and effort to create.[5,9] The widespread use of EHRs makes available large collections of data from real-world patients that can be used for testing.

We describe a generalizable approach for selecting test cases from the clinical data underlying the EHR that provides a breath of coverage, allows for testing at the boundary, uses real patient data, and avoids the creation of a complex search algorithm. Previous work by Tso et al. and Martins et al. used broad patient groups or clinical categories to stratify test cases.[4,5] In contrast, our approach stratifies test cases across specific system paths and then applies a *proportion heuristic* to allocate cases.[8,13,17] In brief, our method uses a set of production rules (Boolean filters) to create simplified paths (stacked if-then statements) that mimic the system's logic and partition the input space. We simplify path selection by using rule consolidation, tree pruning, and key scenarios. This method generates a comprehensive set of test cases that provides adequate coverage for the testing of a knowledge-based CDS.

Methods

CDS and Knowledge-Base

The ATHENA-CDS system was used as a proof-of-concept for our method; however, we intend the process described here to be generalizable to any knowledge based-system expressible as a collection of production rules and with access to real-world patient data such as from EHR systems. ATHENA-CDS includes modules for several chronic medical conditions including: hypertension, diabetes, heart failure, hyperlipidemia, and chronic kidney disease.[5] For this paper, we focused on the Type 2 Diabetes (T2DM) knowledge-base, which has recently been updated.

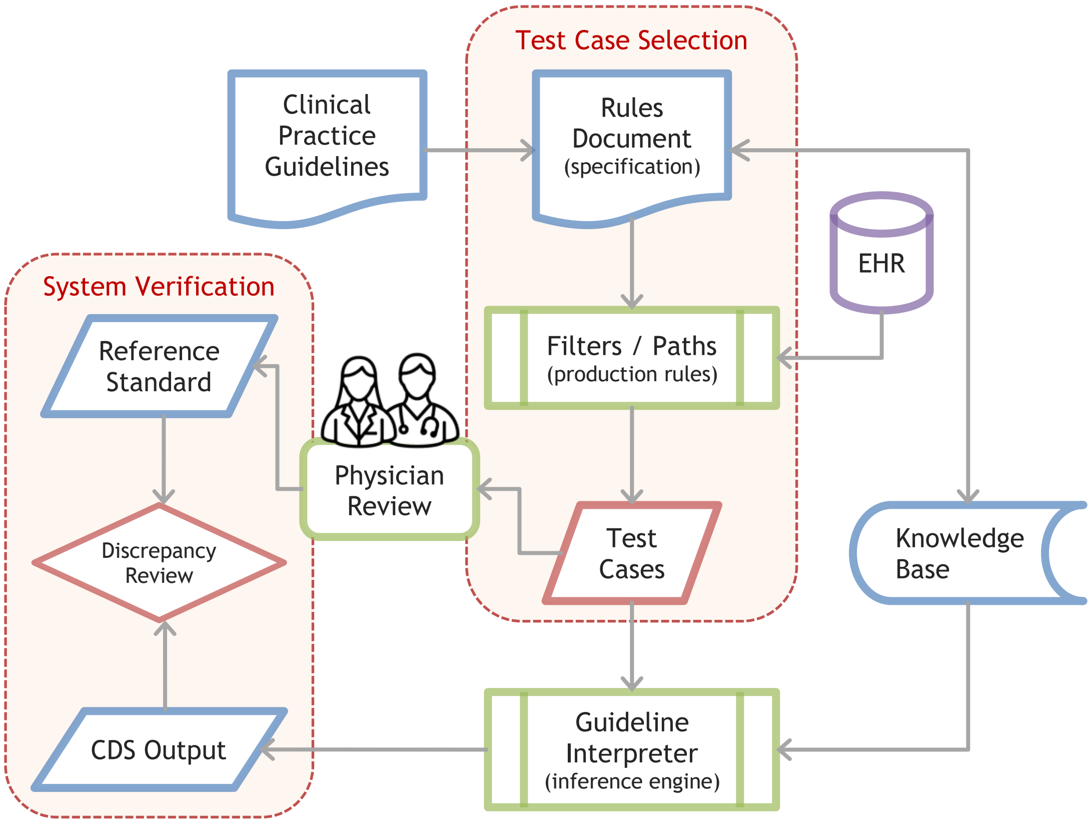
Test Case Selection
Clinical Practice Guidelines
Rules Document (specification)
EHR
System Verification
Filters / Paths (production rules)
Reference Standard
Physician Review
Test Cases
Knowledge Base
Discrepancy Review
Guideline Interpreter (inference engine)
CDS Output

Figure 1. Test Case Selection and System Verification Process

The process for creating test cases starts with clinical practice guidelines being converted into a Rules Document. The Rules Document is used to create filters and paths, which are applied to real patients from the EHR to create test cases. Test cases are then used for system verification.

Rules Document

As previously described,[4,5,7] clinical domain experts and knowledge engineers distill clinical practice guidelines, which were created for physician consumption, to a refined "Rules Document," which provides an unambiguous human-readable specification.[1,16] The full system logic is captured in the Rules Document through narrative, pseudocode, tables, and flow diagrams. The goals of the Rules Document are to record the clinical knowledge encoded in the system and to serve as a reference for system verification. Rather than use the actual knowledge-base, we derived the set of filters used in our method from the Rules Document (Figure 1), which was easier to convert into production rules and allowed us to remain agnostic to the knowledge representation (e.g. frames, semantic networks, etc.).[1]

Filters / Rules

Since exhaustive testing is impractical, previous studies describe a method that generates a representative set of test cases from system paths that provides adequate system coverage.[8,13,17] Similarly, we extracted a set of production rules (Boolean filters) from the Rules Document that allowed us to construct a set of paths (stacked if-then statements) that represents a simplified version of the system. For example, to find patient cases where saxagliptin (a diabetes medication) is relatively contraindicated (path J.2 in Table 1), we created two filters: (1) does the patient have heart failure and (2) has the patient ever had an eGFR (estimated Glomerular Filtration Rate) less than 45. When combined with other filters, we can construct complex patient scenarios. These filters were split into initial filters (Figure 2) and *Above Goal* filters (Figure 3). Patient records are stored and available in a relational database management system (Microsoft SQL Server, 2016). All filters were created using Structured Query Language (SQL).

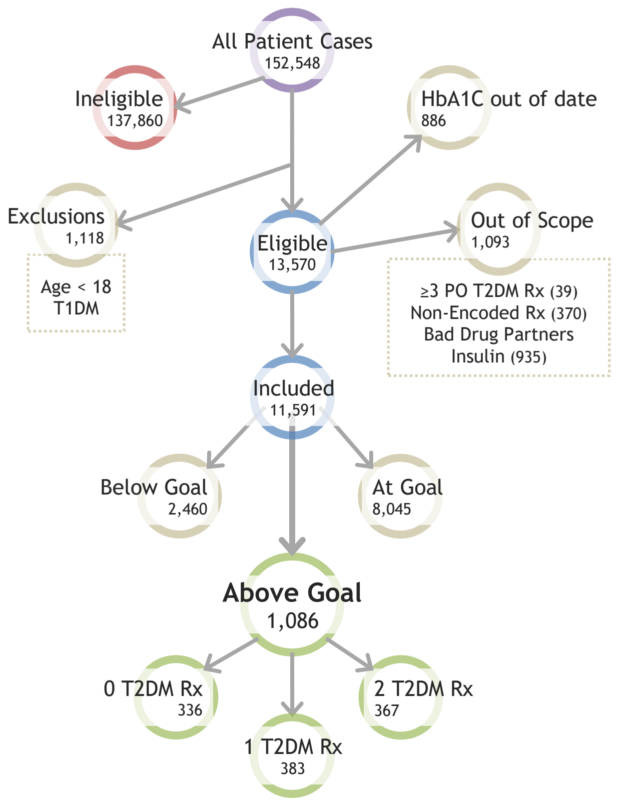

Figure 2. Initial Path Tree Diagram

Patient cases follow a straightforward path to arrive at the Above Goal category, in which the system will give treatment recommendations. First, all cases are filtered for eligibility criteria, exclusions, being out of scope, or not having a recent laboratory marker for glycemic control. Then, cases are evaluated for glycemic control goals. A large number of included patient cases are at goal, leaving 1,086 cases in the Above Goal category. T2DM = Type 2 Diabetes, T1DM = Type 1 Diabetes, Rx = Drugs.

Initial Filters

We mapped the initial path of the diabetes knowledge-base as depicted in Figure 2. Physicians receive recommendations only if the patient case meets eligibility criteria, does not have any exclusions, is not “out of scope,” has a recent Hemoglobin A1c (HbA1C), and the HbA1C is above target diabetes goals. Eligibility criteria include cases with a T2DM diagnosis code or active glycemic control medications. Exclusion criteria include cases under 18

[1] The ATHENA-CDS is uses frames for knowledge representation.

years of age or those with Type 1 Diabetes. Out of scope cases have either (1) more than three active prescriptions of oral hypoglycemic medications, (1) insulin as an active prescription, (3) any “non-encoded drug” (medications that were determined to be out of scope), or (4) any “bad drug partner” (other drugs that may cause adverse interactions). For patient cases that do not have a HbA1c or Glycosylated Hemoglobin (GH) in the last year, the system will recommend obtaining these tests and provide no further recommendations. For the remaining cases, there are three scenarios: the patient’s data indicate the case is at goal, below goal, or above goal. Being at goal is defined as $6 \leq$ HbA1C ≤ 9 or $9 \leq$ GH ≤ 11; therefore, cases are above goal if they have a HbA1C > 9 or GH > 11.

Above Goal Filters

If a patient’s data places the case in the Above Goal category, the system will give treatment recommendations. Based on the clinical practice guidelines, the VA formulary, and bounding decisions, there are only five drugs available for recommendation (encoded drugs): metformin, glipizide, pioglitazone, saxagliptin, and empagliflozin.

Each of these drugs can fall into one of the following nine endpoints recommendations based on the specific patient data (Figure 3): absolute contraindication, relative contraindication (an intermediate endpoint that can coexist with other endpoints), do not start controllable criteria, do not start uncontrollable criteria, start drug, do not increase dose controllable criteria, do not increase dose uncontrollable criteria, and increase dose. Additionally, a drug can be at maximum dose, intermediate dose, or not started. If there is an absolute contraindication, the drug will not be recommended, and if there is an active prescription for a drug with an absolute contraindication, a recommendation will be made to stop the drug and a substitution will be suggested. Drugs with relative contraindications may still be recommended because sometimes the benefit of the medication will outweigh the risk of the relative contraindication; however, the relative contraindication will be displayed and the drug will be placed lower in the list of recommendations. Relative contraindication is an intermediate endpoint instead of a terminal endpoint since it does not have an impact on later recommendations. If the drug does not have an active prescription, the system can recommend starting the medication or the recommendation can be blocked by controllable criteria (e.g. missing or outdated laboratory values) or blocked by uncontrollable criteria (e.g. laboratory values that are above or below a cutoff). If the drug is at an intermediate dose, the system can recommend increasing the medication or the recommendation can be blocked for dose increase by controllable criteria or blocked for dose increase by uncontrollable criteria.

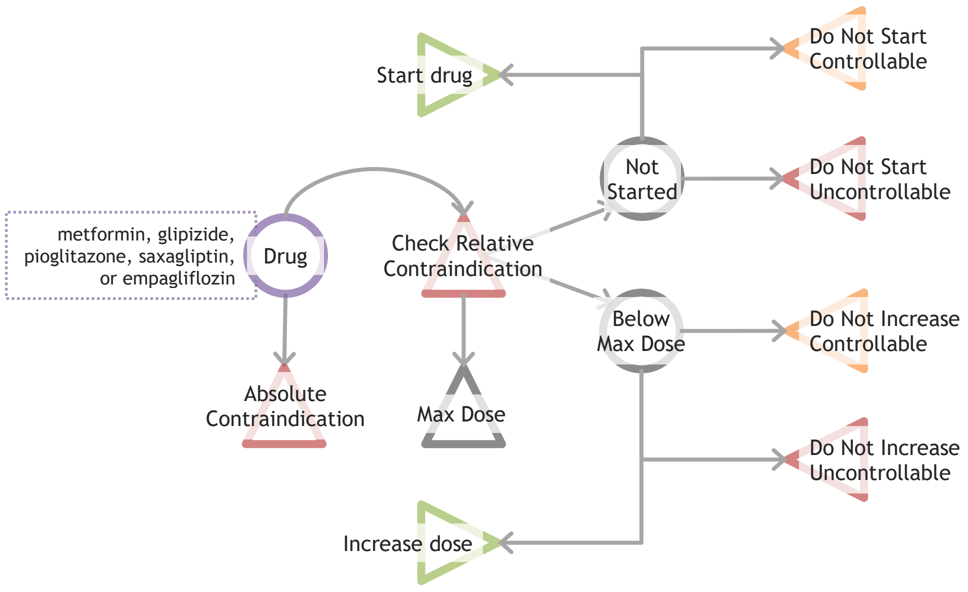
Start drug
Do Not Start Controllable
Not Started
Do Not Start Uncontrollable
metformin, glipizide, pioglitazone, saxagliptin, or empagliflozin
Drug
Check Relative Contraindication
Below Max Dose
Do Not Increase Controllable
Absolute Contraindication
Max Dose
Do Not Increase Uncontrollable
Increase dose

Figure 3. Drug Scenarios and Endpoints

Triangles indicate endpoints (relative contraindication is an intermediate endpoint). Drugs are first checked for absolute contraindications. If there are no absolute contraindications, then drugs are checked for relative contraindications and dosage. Relative contraindications affect recommendation order but do not influence start drug or increase dose recommendations. If the drug is at maximum dose, other medications are considered. If there is no active prescription or if the drug is at an intermediate dose, the system checks multiple fulfillment criteria to make a final recommendation.

Drugs fall into the various endpoints shown in Figure 3 based on multiple fulfillment criteria, too many to list in space constraints of this paper. Many of these filters are based on laboratory values or patient specific characteristics. For example, for metformin, if the patient has no absolute contraindications to metformin, has no active prescriptions for metformin, but has an eGFR between 30 and 45 in the past month, then the drug is placed in the category "do not start uncontrollable" (path G.4 in Table 1).

Paths

Using the stacked filters combined with the various drug endpoints, we constructed several system paths as shown in Table 1. These paths provide wide coverage of the problem space in a compact form.[17] Initial paths were relatively simple and their logic was exhaustively coded. However, with Above Goal paths, because of the presence of multiple early branch points and cyclical rules, it was impractical to exhaustively code system logic using if-then statements.[17] We used three simplifications to constrict the number of paths: rule consolidation, tree pruning, and selective cases. For all three methods we used both clinical and informatics domain expertise to determine which simplifications were high yield and unlikely to omit important distinctions.

Rule consolidation entails combining rules where placing a branch would cause unnecessary complexity and the simplification would not substantially decrease the test coverage. For example, instead of checking separately for HbA1c and GH goals, we combined these laboratory checks into one rule. Tree pruning entails not traversing down the full path to a hypothetical leaf node when the added granularity would require substantial effort without providing additional test coverage. Hura and Srikanthan used a similar process to condense a set of nodes into a single node.[17] We used a combination of domain expertise and a *proportion heuristic*, as described later, to determine where and when to prune. For example, we did not systematically evaluate two-drug recommendations (e.g. glipizide and empagliflozin) since (1) the drug recommendation programming logic is independent and if an error existed then it would likely be found in the one-drug case and (2) some drug combinations are exceedingly rare in the real-world data (e.g. there was only one above goal case with both glipizide and empagliflozin). Finally, selective cases are key scenarios determined by a domain expert that must be tested. Starting with specific key scenarios, we conducted retrograde analysis to determine which filters were needed to select a representative test case. For example, one key scenario that is often encountered in the real world is a patient with prescriptions for both metformin at maximum dose and glipizide at maximum dose (path L.5 in Table 1).

Scenarios

There are three base scenarios for Above Goal paths: (1) the patient has no active T2DM medications, (2) the patient has one active T2DM medication, or (3) the patient has two active T2DM medications. In scenario one, when the patient has no active prescriptions, metformin is recommended as the first line agent unless there is an absolute contraindication. For scenario two, where appropriate, we evaluated drugs for maximum dose, do not increase controllable criteria, and do not increase uncontrollable criteria. For scenario three, we only selected specific key scenarios to test (paths L.1-6 in Table 1) due to the large number of potential scenarios and the low yield for infrequent scenarios. As stated previously, exhaustive testing is impractical; the use of key scenarios allows for expert-based and thoughtful extension of test coverage.

Proportion Heuristic

Researchers have used different strategies to determine the number of cases per category/path: Gonzalez et al. used a priority heuristic (i.e. high, medium, low) per path,[13] Tso et al. distributed cases evenly among 12 broad patient groups,[5] and Martins et al. distributed evenly among 5 clinical categories.[4] Unlike these studies, we first partition the input space into paths, as described above, and then use an objective measure of path importance, which we call the *proportion heuristic*. The proportion heuristic is the number of cases found in the path compared to cases with a similar parent node. For example, of all eligible patients there are 370 cases who are out of scope because of non-encoded drugs (path C.2) and 886 cases who have no HbA1C or GH in the past year (path D.1), therefore we will select the number of test cases for each path with the same ratio of 886:370.

Number of Test Cases

We set the desired total number of test cases to 200, which would typically be based on resource and data constraints. As shown in Figure 4, we arbitrarily set test cases to comprise of 80% cases that are Above Goal (n=160), 5% cases that are ineligible (n=10), and 15% remaining cases (i.e. exclusions, out-of-scope, etc.) (n=30). To increase testing entropy and aid in catching errors of omission, we arbitrarily select 20% of the Above Goal cases at random (n=32) and the remaining 80% using filters (n=128). Within the filtered groups, the cases are allocated using the proportion heuristic. Since applying the heuristic may result in decimal values, we apply a ceiling function to obtain the final whole number. To generate a set of test cases for each path, we draw *n* cases at random from patients in that specific path, where *n* is the final number of cases determined by our proportion heuristic.

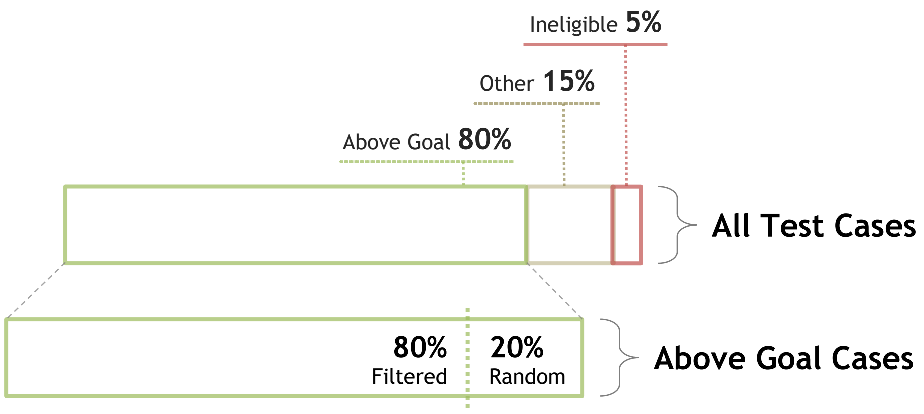

Figure 4. Proportion of Test Cases

We arbitrarily set the proportion of test cases for the following categories: 80% of the test cases will comprise of Above Goal cases, 20% other cases (exclusions, out of scope, etc.), and 5% ineligible cases. Of the Above Goal cases, 80% will be derived from filters/paths as described in our method and 20% will be selected at random from the database.

Results

Initial Path Number of Patient Cases

Figure 2 shows the initial path and associated number of patient cases at each step. The database contained 152,548 patients. A large number of patient cases (n=137,860) were ineligible as they did not have a T2DM diagnosis or have any T2DM medications. There were 1,118 exclusions, 886 cases with no HbA1C or GH in the last year, 1,093 out of scope cases including 935 on insulin, 370 with non-encoded drugs, and 39 on more than three oral hypoglycemic diabetes medications. After these initial filters, there were 11,591 included cases, of which 8,045 were at goal, 2,460 were below goal, and 1,086 were above goal.

Above Goal Number of Patient Cases

Of the cases that were in the Above Goal category, 336 had no encoded diabetes medications, 383 had one medication, and 367 had two medications. For cases that had at least one medication, 625 had prescriptions for metformin, 416 had prescriptions for glipizide, 91 had prescriptions for saxagliptin, 46 had prescriptions for pioglitazone, and 3 had prescriptions for empagliflozin.

Filters and Paths

We created a total of 48 filters, including 11 initial filters and 37 Above Goal filters. For the Above Goal filters, 18 were for laboratories, 12 were drug specific (e.g. dosage, adverse drug reactions), 6 were for diagnoses, and 1 was for gender. Using these filters, we found 50 unique system paths as shown in Table 1. The number of test cases selected per path ranged from 0 to 15 with a median of 2.

Table 1. Paths, Number of Patient Cases, and Number of Test Cases

	Path Name	Patient Cases	Test Cases
A.1	Ineligible	138,018	10
B.1	Excluded	1,118	3
C.1	Out of scope, ≥3 oral T2DM medications	39	1
C.2	Out of scope, non-encoded medications	370	1
C.3	Out of scope, active prescription for insulin	935	2
D.1	No HbA1c or GH in past year	886	2
E.1	Below goal for glycemic control	2,460	5
E.2	At goal for glycemic control	8,045	15

F.1	Above goal for glycemic control	1,086	
F.2	Above goal, no T2DM medications	336	
F.3	Above goal, one T2DM medication	383	
F.4	Above goal, two T2DM medications	367	
G.1	Above goal, metformin absolute contraindication	103	2
G.2	Above goal, metformin relative contraindication	509	9
G.3	Above goal, no metformin, metformin do not start controllable	326	6
G.4	Above goal, no metformin, metformin do not start uncontrollable	160	3
G.5	Above goal, no metformin, start metformin	107	2
G.6	Above goal, on metformin, maximum dose	47	1
G.7	Above goal, on metformin, do not increase controllable	281	5
G.8	Above goal, on metformin, do not increase uncontrollable	52	1
G.9	Above goal, on metformin, increase metformin	220	4
H.1	Above goal, glipizide absolute contraindication	4	1
H.2	Above goal, glipizide relative contraindication	67	2
H.3	Above goal, no glipizide, start glipizide	669	11
H.4	Above goal, on glipizide, maximum dose	122	2
H.5	Above goal, on glipizide, increase glipizide	241	4
I.1	Above goal, pioglitazone absolute contraindication	24	1
I.2	Above goal, pioglitazone relative contraindication	61	1
I.3	Above goal, no pioglitazone, pioglitazone do not start controllable	579	10
I.4	Above goal, no pioglitazone, pioglitazone do not start uncontrollable	178	3
I.5	Above goal, no pioglitazone, start pioglitazone	374	6
I.6	Above goal, on pioglitazone, maximum dose	45	1
I.7	Above goal, on pioglitazone, do not increase controllable	14	1
I.8	Above goal, on pioglitazone, do not increase uncontrollable	2	1
I.9	Above goal, on pioglitazone, increase pioglitazone	8	1
J.1	Above goal, saxagliptin absolute contraindication	41	1
J.2	Above goal, saxagliptin relative contraindication	282	5
J.3	Above goal, no saxagliptin, saxagliptin do not start controllable	574	10
J.4	Above goal, no saxagliptin, start saxagliptin	382	7
J.5	Above goal, on saxagliptin, maximum dose	63	2
J.6	Above goal, on saxagliptin, do not increase controllable	0	0
J.7	Above goal, on saxagliptin, increase saxagliptin	15	1
K.1	Above goal, empagliflozin absolute contraindication	236	4
K.2	Above goal, no empagliflozin, empagliflozin do not start controllable	481	8
K.3	Above goal, no empagliflozin, start empagliflozin	367	6
K.4	Above goal, on empagliflozin, maximum dose	0	0
K.5	Above goal, on empagliflozin, do not increase controllable	0	0
K.6	Above goal, on empagliflozin, increase empagliflozin	2	1
L.1	Above goal, on metformin and glipizide, metformin contraindicated, glipizide maximum dose	1	1
L.2	Above goal, on metformin and glipizide, metformin do not increase controllable, glipizide maximum dose	49	1
L.3	Above goal, on metformin and glipizide, metformin do not increase uncontrollable, glipizide maximum dose	7	1
L.4	Above goal, on metformin and glipizide, increase metformin, glipizide maximum dose	22	1
L.5	Above goal, on metformin and glipizide, metformin maximum dose, glipizide maximum dose	14	1
L.6	Above goal, on metformin and saxagliptin, metformin contraindicated, saxagliptin maximum dose	2	1
G.1	Above goal, random selection		32
		Total:	**200**

Discussion

Using 48 filters to create 50 simplified system paths we were able to select a set of test cases that provides a breadth of coverage while also testing critical areas of the ATHENA-CDS knowledge-based system. This method improves upon previous work[4,5] by stratifying cases across a larger number of divisions, finding a set of paths that are representative of the underlying system, and allocating cases using an objective heuristic. Our method is generalizable to other knowledge-based systems that are expressible as production rules and have access to real-world patient data.

Testing Coverage

Coverage measures test comprehensiveness or how well the test reflects the input space.[9] Coverage is lowered by sparsity in the data (e.g. the case doesn't exist in the EHR), inadequate partitioning of the input space (e.g. the case was never defined),[10] and resource constraints (e.g. cost, labor, time prohibitions). If we are only testing for *errors of commission* (knowledge that is present), then only a few cases per path are required; however, if we are also attempting to catch *errors of omission* (knowledge that is absent), then random testing is required.[10,13] Additionally, since we are using rule consolidation and tree pruning for simplification, there is increased need for random testing.

Exhaustive software testing is often impractical[13] and would require all variations of data inputs including but not limited to maximum, minimum, missing values, and sampling at both sides of all cutpoints. Therefore, CDS developers need methods that, while not providing complete coverage of all possible cases, provide reasonable coverage of key cases and produce acceptable system performance.[9,17] Our method is not intended to, and will not, achieve complete test coverage; it intends to maximize test coverage given the constraints of testing by creating a set of simplified paths that mimic the system's logic and simplifying path selection by using rule consolidation, tree pruning, and key scenarios. Using the proportion heuristic, we give our best attempt at reasonable distribution of test cases across the CDS. The method described in this paper is an advance over prior methods.[4,5]

System Validation and Verification

This paper guides researchers in obtaining a set of test cases, which is an early—yet critical—step in the multi-step evaluation process (Figure 1). After selection, test cases are evaluated by domain experts for intended output in order to create a reference standard.[4,5,9,14,15] To conduct system *verification*,[9,13] the test cases are evaluated against the software specification. To conduct system *validation*, the cases are evaluated against a real-world "gold standard."[4,7,9] Since our method relies on creating paths from the software specification it is more appropriate for system verification.

These methods for system validation and verification fall under the black box category as discrepancies detect that an error has occurred, but not where the error occurs.[16] In addition to system testing, test cases can be used for white box methods (where internal structures are tested) such as unit testing and control flow testing or other black box methods such as regression testing and Turing tests.[9,10,16] Errors found in later steps may serve as an evaluation of the adequacy of this initial step of testing.[11,12]

Determination of the Gold Standard

The reference standard is not the same as the "gold standard" since expert opinion is often equivocal or incorrect due to fatigue, inattention, errors in the specification, disagreement, or misunderstandings.[4,13,14] Some cases may be incorrectly evaluated by both the expert and the CDS (i.e. both give the same wrong recommendation) causing the gold standard to be incorrectly labeled. Using multiple experts in the review process can mitigate this problem and improve the accuracy of determining gold standard cases.[14] Once discrepancies are reconciled, cases may be labeled as gold standard and used for future testing.[4]

System Accuracy

Discrepancies found in testing are helpful in correcting software errors;[5] after the discrepancies are reconciled and the software has been updated, the remaining software incongruence with the reference standard indicates the accuracy of the system. Expert systems are not expected to have perfect performance since they are abstractions of reality and require a set boundary.[1,4,9] In the context of CDS, this entails settling for an acceptable level of performance by establishing a boundary that captures all of the desired clinical scenarios.[1,4,7] Systems such as ATHENA-CDS are designed to support licensed health professionals rather than to take actions directly; health professionals using the system are aware of potential limitations of data and programming.

Failure to select test cases at this boundary, which often require considerable skill, results in decreased test coverage and may incorrectly inflate system performance. For example, if most of the errors occur at the boundary, selecting a high proportion of standard cases for testing will overestimate of the system's accuracy.[9] On the other hand, selecting disproportionately more edge-cases, as described in this paper, will underestimate the system's accuracy. However, the goal of testing is to elicit discrepancies and find errors to improve accuracy, not measure accuracy itself;[4,14] if the goal were to measure accuracy, stratified sampling based on real-world weightings would be more appropriate.[9,14]

Case Sparsity and Synthesis

Sparsity is a function of data volume, data variety, and the joint probability of the edge case. Large, unselected, real-world clinical datasets, as used in this study, usually have features that are normally distributed.[5] Because of this, researchers may fail to find certain edge cases and will have to either fabricate test cases de novo, which is problematic for clinical data, or, more preferably, synthesize test cases from a prototype.[5] One strategy is for researchers to synthesize cases by starting with the empty leaf node and traverse up the tree to an ancestor node that can serve as a prototype and then altering this prototype to fit the edge case. In our study, we "lost" approximately 8,000 cases for patients whose glycemic control met the goal; these cases could easily be converted to Above Goal cases while retaining the verisimilitude of the rest of the patient data. Additionally, there were three paths that did not result in any test cases (paths J.6, K.4, and K.5). Synthesizing test cases is an important topic for future research.

Expansion or Customization of the Knowledge-Base

New evidence, updated clinical practice guidelines, or local customization require changes to the knowledge-base, updates to the Rules Document, and conduction of regression testing.[2,4,11] Using our testing approach, changes to the Rules Document are easily translated to new filters and paths that can be used for regression testing.

Limitations

The method described here may not apply or be ideal for all situations; indeed, there are many approaches for system testing and test case selection of knowledge-based systems and researchers should select a strategy that is tailored for their application.[10,16] For example, our method heavily relies on access to patient records, yet these may be not available or obtainable. Even with access to real patient records, if the data are sparse then researchers may not be able to generate critical mass of edge cases. In addition, if the knowledge of a system is not easily representable by simple if-then rules, then our method would be difficult to apply.

Our method relies on the existing knowledge contained in the knowledge-base and Rules Document; therefore, it favors catching *errors of commission* and the method may fail at catching all *errors of omission*, despite including random cases to identify these errors.[13] Finding errors of omission is extremely difficult since the missing knowledge is hard to predict and requires extensive random testing or sensitivity analysis.[5,9] Even with random testing, if errors are not uniformly distributed across the input space they may be missed in fault prone areas.[10] Researchers have shown that post-fielding surveillance, akin to post-marketing surveillance for adverse drug events, is effective at detecting these hard to catch errors.[6,11,12]

Conclusion

Testing of knowledge-based systems is complex, labor intensive, and expensive. Thoughtful test case selection improves testing coverage while minimizing testing burden.[5] In this paper, we describe a generalizable method for selecting test cases that uses a set of production rules (filters) to create simplified paths. Test cases are allocated to paths using a proportion heuristic. Using the ATHENA-CDS diabetes knowledge-base as a proof-of-concept, we generated a comprehensive set of test cases that provides adequate test coverage. With the increasing use and sophistication of CDS, rigorous testing of these systems is important.

Acknowledgments

This work was supported by the VA Health Services Research and Development (HSR&D) Quality Enhancement Research Initiative (QUERI) project: Optimizing Appropriate Use of Medications for Veterans, grant number IP1 HX002261-01 (PIs: Heidenreich, Goldstein). Dr. Usman's work was supported by the VA Office of Academic Affiliations and the VA HSR&D program through an Advanced Fellowship in Medical Informatics. We would like to thank Dr. Michael Ashcraft for his input as a subject matter expert and Jan Urbano for his help in editing the manuscript. Views expressed are those of the authors and not necessarily those of the Department of Veterans Affairs or other affiliated institutions.

References

1. Plant RT. Rigorous approach to the development of knowledge-based systems. Knowl-Based Syst. 1991 Dec 1;4(4):186–96.

2. Goldstein MK, Hoffman BB, Coleman RW, Musen MA, Tu SW, Advani A, et al. Implementing clinical practice guidelines while taking account of changing evidence: ATHENA DSS, an easily modifiable decision-support system for managing hypertension in primary care. Proc AMIA Symp. 2000;300–4.

3. Goldstein MK, Hoffman BB, Coleman RW, Tu SW, Shankar RD, O'Connor M, et al. Patient safety in guideline-based decision support for hypertension management: ATHENA DSS. Proc AMIA Symp. 2001;214–8.

4. Martins SB, Lai S, Tu S, Shankar R, Hastings SN, Hoffman BB, et al. Offline testing of the ATHENA Hypertension decision support system knowledge base to improve the accuracy of recommendations. AMIA Annu Symp Proc. 2006;539–43.

5. Tso GJ, Yuen K, Martins S, Tu SW, Ashcraft M, Heidenreich P, et al. Test Case Selection in Pre-Deployment Testing of Complex Clinical Decision Support Systems. AMIA Summits Transl Sci Proc. 2016;2016:240–9.

6. Goldstein MK, Coleman RW, Tu SW, Shankar RD, O'Connor MJ, Musen MA, et al. Translating Research into Practice: Organizational Issues in Implementing Automated Decision Support for Hypertension in Three Medical Centers. J Am Med Inform Assoc JAMIA. 2004;11(5):368–76.

7. Trafton JA, Martins SB, Michel MC, Wang D, Tu SW, Clark DJ, et al. Designing an automated clinical decision support system to match clinical practice guidelines for opioid therapy for chronic pain. Implement Sci IS. 2010 Apr 12;5:26.

8. Diaz E, Tuya J, Blanco R. A modular tool for automated coverage in software testing. In: Eleventh Annual International Workshop on Software Technology and Engineering Practice. 2003. p. 241–6.

9. O'Keefe RM, Balci O, Smith EP. Validating Expert System Performance. IEEE Expert. 1987 Jan;2(4):81–90.

10. Kirani SH, Zualkernan IA, Tsai W-T. Evaluation of Expert System Testing Methods. Commun ACM. 1994 Nov;37(11):71–81.

11. Chan AS, Martins SB, Coleman RW, Bosworth HB, Oddone EZ, Shlipak MG, et al. Post-fielding Surveillance of a Guideline-based Decision Support System. In: Henriksen K, Battles JB, Marks ES, Lewin DI, editors. Advances in Patient Safety: From Research to Implementation (Volume 1: Research Findings) [Internet]. Rockville (MD): Agency for Healthcare Research and Quality (US); 2005 [cited 2017 Aug 23]. (Advances in Patient Safety). Available from: http://www.ncbi.nlm.nih.gov/books/NBK20476/

12. Lin ND, Martins SB, Chan AS, Coleman RW, Bosworth HB, Oddone EZ, et al. Identifying Barriers to Hypertension Guideline Adherence Using Clinician Feedback at the Point of Care. AMIA Annu Symp Proc. 2006;2006:494–8.

13. Gonzalez AJ, Gupta UG, Chianese RB. Performance evaluation of a large diagnostic expert system using a heuristic test case generator. Eng Appl Artif Intell. 1996 Jun 1;9(3):275–84.

14. O'Leary DE. Validation of Expert Systems- with Applications to Auditing and Accounting Expert Systems*. Decis Sci. 1987 Jul 1;18(3):468–86.

15. Fox J, Bury J. A quality and safety framework for point-of-care clinical guidelines. Proc AMIA Symp. 2000;245–9.

16. Batarseh FA, Gonzalez AJ. Validation of knowledge-based systems: a reassessment of the field. Artif Intell Rev. 2015 Apr 1;43(4):485–500.

17. Hura GS, Srikanthan T. A graph-theoretic approach to expert-system testing. Eng Appl Artif Intell. 1995 Oct 1;8(5):539–47.