*Application of Information Technology* ■

# Domain-constrained Generation of Clinical Condition Sets to Help Test Computer-based Clinical Guidelines

PERRY L. MILLER, MD, PHD

**Abstract** The paper describes T/Gen, a prototype computer-based tool designed to help maintain the knowledge in a computer-based clinical practice guideline that provides patient-specific recommendations. T/Gen takes as input a set of clinical conditions to which a guideline must react, and allows the user to specify domain-specific constraints as to which combinations of conditions do not make sense or do not need to be exhaustively tested against one another. T/Gen automatically generates constrained sets of combinations of clinical conditions, each corresponding to a clinical case (or to several closely related clinical cases) that can be used to help test the computer-based guideline. The combinations can be used to test the guideline logic using T/Gen's built-in logic interpreter, or to generate a set of test cases for use in testing an operational guideline system. T/Gen has been developed and tested with five pilot guidelines, for two childhood immunization series, for influenza vaccination, for primary thyroid screening, and for embryo transplantation. The paper describes how T/Gen's approach is implemented for the five pilot guidelines, outlines the current status and future directions of the project, and discusses the design issues that arose in the course of carrying out the work.

■ **J Am Med Inform Assoc.** 2001;8:131–145.

The national emphasis on clinical practice guidelines will result in an increasing number of guidelines being placed into a computer-based form that allows the generation of patient-specific recommendations. The creation of such a computer-based guideline for a complex area of medicine can be a challenging and time-consuming process. The ongoing maintenance of the guideline is likely to be at least an equal challenge. The fundamental knowledge underlying a clinical domain may evolve rapidly. As a result, a clinical guideline will need to be updated on a regular basis, perhaps annually. The computer-based version of the guideline will also need to be updated. In addition, it will need to be thoroughly tested to ensure that it correctly reflects the nuances of the new recommendations.

It will be important to have computer-based tools that can assist in this knowledge-maintenance process. This paper describes preliminary work to develop and test one such tool, T/Gen (Test case Generator). T/Gen is written in the Lisp programming language and is currently linked to five pilot guidelines that have been expressed using a simplified version of GLIF (the Guideline Interchange Format), an evolving representation for expressing clinical guideline logic.[1] T/Gen can potentially operate with any guideline, not just guidelines written in GLIF.

As illustrated in Figure 1, T/Gen is designed to operate as follows.

- T/Gen takes as input a set of clinical conditions to which a guideline must react.

Affiliation of the author: Yale University School of Medicine, New Haven, Connecticut.

This work was supported in part by NIH grants R01 LM06682 and G08 LM05583 from the National Library of Medicine.

Correspondence and reprints: Perry L. Miller, MD, PhD, Center for Medical Informatics, Yale University School of Medicine, P.O. Box 208009, New Haven, CT 06520-8009; e-mail: <perry.miller@yale.edu>.

Received for publication: 4/12/00; accepted for publication: 9/12/00.

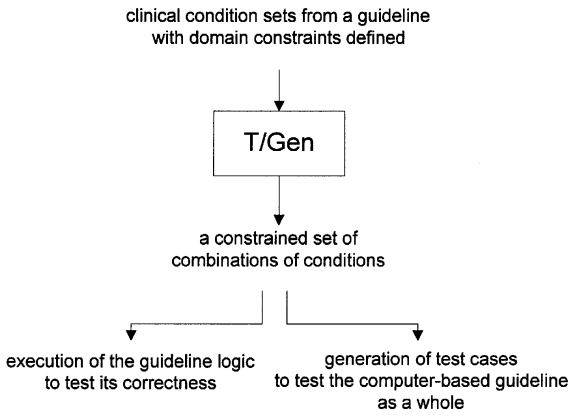


**Figure 1.** A schematic overview of T/Gen's operation.

- T/Gen allows the user to specify domain-specific constraints as to which combinations of clinical conditions do not make sense or do not need to be exhaustively tested against one another, as described later. (If such domain constraints are not used, a potentially large and unmanageable number of combinations of clinical conditions may be produced, even for a guideline of only moderate complexity.)
- T/Gen automatically generates constrained sets of combinations of clinical conditions. Each combination corresponds to a clinical case (or to several closely related clinical cases) that can be used to help test the guideline. The goal is to help ensure that the guideline responds appropriately to all meaningful combinations of clinical conditions.
- These sets of clinical conditions can be automatically run though T/Gen's Lisp implementation of the guideline's GLIF logic. This process helps test the accuracy, consistency, and completeness of that logic.
- The sets of clinical conditions can also be automatically converted into test cases. These test cases can be used to help test the entire system that implements the guideline, which typically includes the input and output interfaces and the execution engine in addition to the logic itself.

The goal is to provide a tool that can be used interactively in testing a new version of a guideline, a process that could involve iteratively imposing and relaxing several domain-specific constraints in a flexible fashion. T/Gen has been developed and tested with a set of five pilot guidelines:

- Two guidelines involve the Hib (*Haemophilus influenzae* type b) and DTP (diphtheria-tetanus-pertussis) vaccine series. These have been developed in the context of IMM/Serve,[2] a forecasting program for childhood immunization that is currently undergoing national dissemination. The development of tools to help maintain IMM/Serve's knowledge is an ongoing research project in the Yale Center for Medical Informatics.
- Three guidelines, for primary thyroid screening, embryo transplantation, and influenza vaccination, were encoded in GLIF by researchers in the Decision Systems Group at the Brigham and Women's Hospital in Boston, and their colleagues. This work was performed in part to assist in the process of specifying the GLIF language as part of the national InterMED Collaboratory.

The goals of this paper are 1) to describe how T/Gen's approach can be implemented and used in the context of the five pilot guidelines, 2) to outline the current status and future directions of the project, and 3) to discuss interesting design issues that arose during the course of the work.

## Background

The problem of validating and evaluating computer-based clinical guidelines involves a range of research issues.[3–5] The present project focuses on the automatic creation of test cases to help validate a computer-based guideline that produces patient-specific advice. Except for very constrained programs, the correct behavior of a computer program is inherently unprovable.[6] It is therefore necessary to develop strategies that help validate guideline programs within the constraints of what is possible and reasonable in a specific domain.

Two overall approaches to testing a computer program are a "white box" approach, in which the testing is based on the known logic of the program, and a "black box" approach, in which testing is driven by program specifications.[7] T/Gen takes the "white box" approach. A central issue in testing computer programs involves the degree of "coverage" of the program statements or the logical paths through the program, or both.[8] A second issue involves "boundary analysis," in which variables are given extreme values to ensure that correct behavior is still obtained.[9]

A number of formal approaches to validating a computer program with test cases have been used,[10] including mutation analysis,[11] state-based testing,[12] and data flow testing.[13] This work tends to focus on analyzing statistically how well a static set of test cases covers a program's logic. T/Gen has a somewhat different focus, since it allows the flexible gen-

eration of different sets of test cases to help ensure that each logical combination in a specified set has been exercised. The formal approaches frequently require very large numbers of test cases (tens of thousands or more), which would severely limit their utility for maintaining clinical guidelines.

The present work builds on two previous projects at Yale (involving IMM/Test[14] and Commander[15]), which explored the use of constraints in the generation of clinical conditions to assist in knowledge verification in the domain of childhood immunization. T/Gen's approach is more broadly generalizable to non-immunization domains than is that of IMM/Test, and it is more flexible than that of Commander.

A number of recent projects in the field of clinical informatics have focused on the problems of maintaining clinical decision support knowledge, including describing 1) the nature and extent of changes over time,[16] 2) a database to support knowledge maintenance,[17] and 3) approaches and tools to facilitate knowledge maintenance.[18] Earlier work on the Oncocin rule checker involved analyzing the knowledge base itself for redundancy and inconsistency.[19]

In the field of software engineering, work has focused on the generation of test cases using domain knowledge expressed in the form of relations and constraints.[20–23] These are derived from a semantic model of the domain and therefore differ from T/Gen's approach of imposing constraints directly on the condition generation process itself.

## T/Gen: Constrained Generation of Combinations of Clinical Conditions

This section describes T/Gen's operation with the five pilot guidelines. It illustrates 1) how the guidelines are represented in GLIF, 2) how T/Gen accepts as input sets of clinical conditions with domain constraints, and 3) how T/Gen produces combinations of clinical conditions that can be used for logic verification and test case generation. It should be emphasized that the five pilot guidelines are examples used to explore T/Gen's approach. The precise clinical content of the guidelines is not the focus of this work.

### T/Gen and the Hib Guideline

Figure 2 shows a flowchart of a central component of the Hib vaccine guideline. This logic takes as input certain clinical information about a patient, and determines whether a Hib vaccine dose is due ("Hib2 due" means that dose 2 is due) and, if not, which dose (if any) should be scheduled next ("Hib2 next" means that dose 2 should be scheduled next). The guideline also specifies which vaccine brand should be used (Prpomp vs. HbOC). Each dose has an associated set of parameters, which include a minimum age and a minimum wait-interval from the previous dose in the series. Depending on certain clinical conditions (such as whether the child was over 12 months of age when Hib dose 1 was given), a different set of parameters may apply. Thus "Hib2" and "Hib2_final" refer to a different set of parameters to be used, in different circumstances, to determine when a Hib dose 2 is due or when it should be scheduled next. "Hib2 due" indicates that the Hib2 parameter set (which includes a specific minimum age and wait interval) is satisfied.

The Hib Guideline in Simplified GLIF

In the present project, we use a simplified, Lisp-based GLIF syntax to represent the logic of each guideline. We have taken this approach in part because many of the features of GLIF are not centrally relevant to the focus of this paper and in part because GLIF itself is still very much under development.

The GLIF version of the Hib guideline, shown below, encodes explicitly logic shown in Figure 2:

```
;; ———————— Hib dose 1 ————————

conditional_step hib1:
    condition: Hib_prior = 0
    destination: conditional_step hib1_c1
    otherwise: conditional_step hib2

conditional_step hib1_c1:
    condition: Hib1 due
    destination: conditional_step hib1_c2
    otherwise: action_step hib1_a3

conditional_step hib1_c2:
    condition: prpomp indicated
    destination: action_step hib1_a1
    otherwise: action_step hib1_a2

action_step hib1_a1:
    action: due Hib1 Prpomp
    next_step: end

action_step hib1_a2:
    action: due Hib1 HbOC
    next_step: end

action_step hib1_a3:
    action: next Hib1
    next_step: end
```

```
;; ———————— Hib dose 2 ————————
conditional_step hib2:
    condition: Hib_prior = 1
    destination: conditional_step hib2_c1
    otherwise: conditional_step hib3
```

For example, the GLIF objects shown here encode the logic that handles Hib dose 1. Each element represents one of the flowchart boxes and is either a conditional_step or an action_step. When a conditional_step is executed, its condition is evaluated and processing proceeds to one of the two specified next steps as appropriate. (In the examples given in this paper, we have made modest modifications to Lisp syntax to enhance readability.)

## Clinical Condition Generation for the Hib Guideline

The first step in using T/Gen is to define the set of clinical conditions, to which a guideline must react, that are to be tested. For the portion of the Hib guideline shown in Figure 2, there are ten sets of conditions (Figure 3).

The first condition can take on four possible values. The remaining conditions can each take on two possible values. As a result, there are $4 \times 2^9 = 2{,}048$ combinations of these conditions. Thus, a brute-force (or unconstrained) approach to generating conditions for this guidelines generates a large number of clinical conditions and would not be very practical for use in testing the logic.

On the other hand, many of the combinations of conditions generated by a brute-force approach are not clinically meaningful. For example:

- The question of whether Hib1 is due is relevant only when there are no prior Hib doses, i.e., when "Hib_prior = 0." A similar constraint applies to all the other doses.
- Whether the child was more than 12 months old at Hib dose 1 is relevant only when "Hib_prior = 1."
- Whether the child's current age is greater than 12 months is relevant only when "Hib_prior = 2."

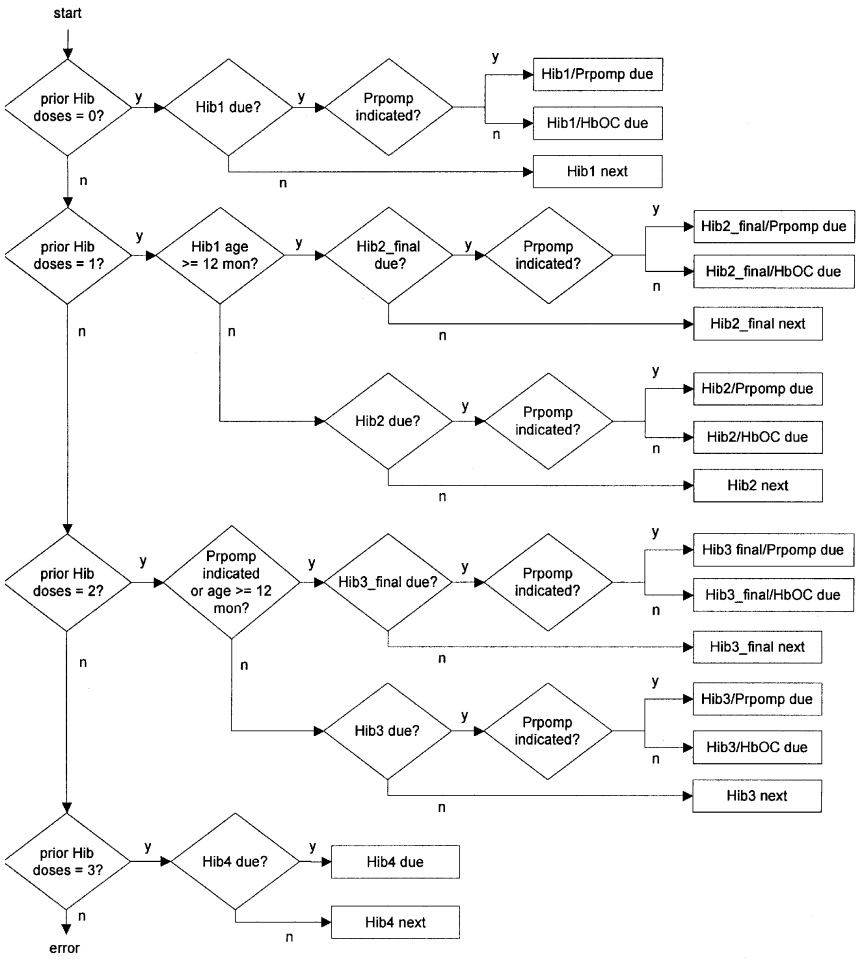


**Figure 2.** A flowchart of the Hib vaccine guideline.

1. [Hib_prior = 0], [Hib_prior = 1], [Hib_prior = 2], [Hib_prior = 3]
2. [Hib1 due], [Hib1 not due]
3. [Hib2 due], [Hib2 not due]
4. [Hib3 due], [Hib3 not due]
5. [Hib4 due], [Hib4 not due]
6. [Hib2_final due], [Hib2_final not due]
7. [Hib3_final due], [Hib3_final not due]
8. [Prpomp indicated], [Prpomp not indicated]
9. [Hib1_age >= 12m], [Hib1_age < 12m]
10. [age >= 12m], [age < 12m]

**Figure 3.** The ten clinical conditions sets for the Hib guideline shown in Figure 2.

- Whether the Prpomp brand is indicated is not relevant if "Hib_prior = 3."

These are examples of domain-specific constraints that can be used by T/Gen to limit dramatically the number of combinations of clinical conditions that it generates. These constraints can be expressed as shown in Figure 4.

Figure 4 lists the same sets of conditions as Figure 3 but includes constraints that limit the circumstances in which condition sets 2 through 10 will be included in the combinations of conditions that are produced. Each constraint listed in Figure 4 (to the right of the condition set to which it applies) indicates the circumstances in which the condition set is to be included in the combinations of conditions that are generated.

For example, condition set 2 ("Hib1 due," "Hib1 not due") will be included only if the condition "Hib_prior = 0" is included. Thus, if the condition "Hib_prior = 1" is included in a combination being generated, condition set 2 will be ignored, as will condition sets 4, 5, 7, and 10. These can be ignored because they are not relevant to the logic being tested when "Hib_prior = 1."

Using these constraints, T/Gen generates only 38 combinations of clinical conditions for Hib, clearly a much more manageable number to check than 2,048.

1. [Hib_prior = 0], [Hib_prior = 1], [Hib_prior = 2], [Hib_prior = 3]
2. [Hib1 due], [Hib1 not due] — only_if [Hib_prior = 0]
3. [Hib2 due], [Hib2 not due] — only_if [Hib_prior = 1]
4. [Hib3 due], [Hib3 not due] — only_if [Hib_prior = 2]
5. [Hib4 due], [Hib4 not due] — only_if [Hib_prior = 3]
6. [Hib2_final due], [Hib2_final not due] — only_if [Hib_prior = 1]
7. [Hib3_final due], [Hib3_final not due] — only_if [Hib_prior = 2]
8. [Prpomp indicated], [Prpomp not indicated] — not_if [Hib_prior = 3]
9. [Hib1_age >= 12m], [Hib1_age < 12m] — only_if [Hib_prior = 1]
10. [age >= 12m], [age < 12m] — only_if [Hib_prior = 2]

**Figure 4.** The ten clinical condition sets for Hib with domain-specific constraints defined.

1. [Hib_prior = 0], [Hib1 due], [Prpomp indicated]
2. [Hib_prior = 0], [Hib1 due], [Prpomp not indicated]
3. [Hib_prior = 0], [Hib1 not due], [Prpomp indicated]
4. [Hib_prior = 0], [Hib1 not due], [Prpomp not indicated]
5. [Hib_prior = 1], [Hib2 due], [Hib2_final due], [Prpomp indicated], [Hib1_age >= 12m]
6. [Hib_prior = 1], [Hib2 due], [Hib2_final due], [Prpomp indicated], [Hib1_age < 12m]
7. [Hib_prior = 1], [Hib2 due], [Hib2_final due], [Prpomp not indicated], [Hib1_age >= 12m]
8. [Hib_prior = 1], [Hib2 due], [Hib2_final due], [Prpomp not indicated], [Hib1_age < 12m]

• • •

20. [Hib_prior = 2], [Hib3 due], [Hib3_final due], [Prpomp indicated], [Age >= 12m]
21. [Hib_prior = 2], [Hib3 due], [Hib3_final due], [Prpomp indicated], [Age < 12m]

• • •

37. [Hib_prior = 3], [Hib4 due]
38. [Hib_prior = 3], [Hib4 not due]

**Figure 5** Representative examples of the 38 constrained combinations of conditions generated by T/Gen for the Hib guideline, using the constrained condition set shown in Figure 4.

In this way, T/Gen takes advantage of semantic constraints in the domain to generate a more efficient and clinically meaningful set of combinations. The combinations generated using these constraints include those shown in Figure 5. As can be seen, for each dose, only those clinical conditions that apply to the dose are included in the combinations generated.

### Additional "Post-generation" Constraints

The constraints described above are applied during the process of generating combinations. After these combinations have been generated, however, there is further potential to apply additional types of constraints. For example, for dose 2, the combinations in Figure 5 include all combinations of the condition set ("Hib2 due," "Hib2 not due") and the condition set ("Hib2_final due," "Hib2_final not due"). It is not necessary to include all these combinations. All combinations that contain the conditions of "Hib2 due" and "Hib2_final due" are redundant and can be eliminated, as can all combinations which contain "Hib2 not due" and "Hib2_final not due." Similar constraints can be imposed for dose 3.

These are called "post-generation" constraints, in the sense that they are applied after an initial set of combinations have been generated. Post-generation constraints eliminate certain combinations, either because they are clinically nonsensical or redundant.

For the Hib guideline, the following four post-generation constraints can be used:

1. [Hib2 due] and [Hib2_final due]
2. [Hib2 not due] and [Hib2_final not due]
3. [Hib3 due] and [Hib3_final due]
4. [Hib3 not due] and [Hib3_final not due]

If a combination contains any of these pairs of conditions, it is eliminated. Applying these post-generation constraints further reduces the number of combinations produced to 22.

### An Additional Constraint

There is still an opportunity for one more constraint that involves the first four combinations, which deal with Hib dose 1. If the condition "Hib1 not due" is true, then Prpomp is irrelevant. This constraint can be included by adding the clause [Hib1 not due] to the constraint on condition 8 (see Figure 4), so that it reads:

8. [prpomp indicated], [prpomp not indicated]

— not_if [Hib_prior = 3] or [Hib1 not due]

When this additional clause is added, the first four conditions in Figure 5 are reduced to the following three combinations:

1. [Hib_prior = 0], [Hib1 due], [Prpomp indicated]
2. [Hib_prior = 0], [Hib1 due], [Prpomp not indicated]
3. [Hib_prior = 0], [Hib1 not due]

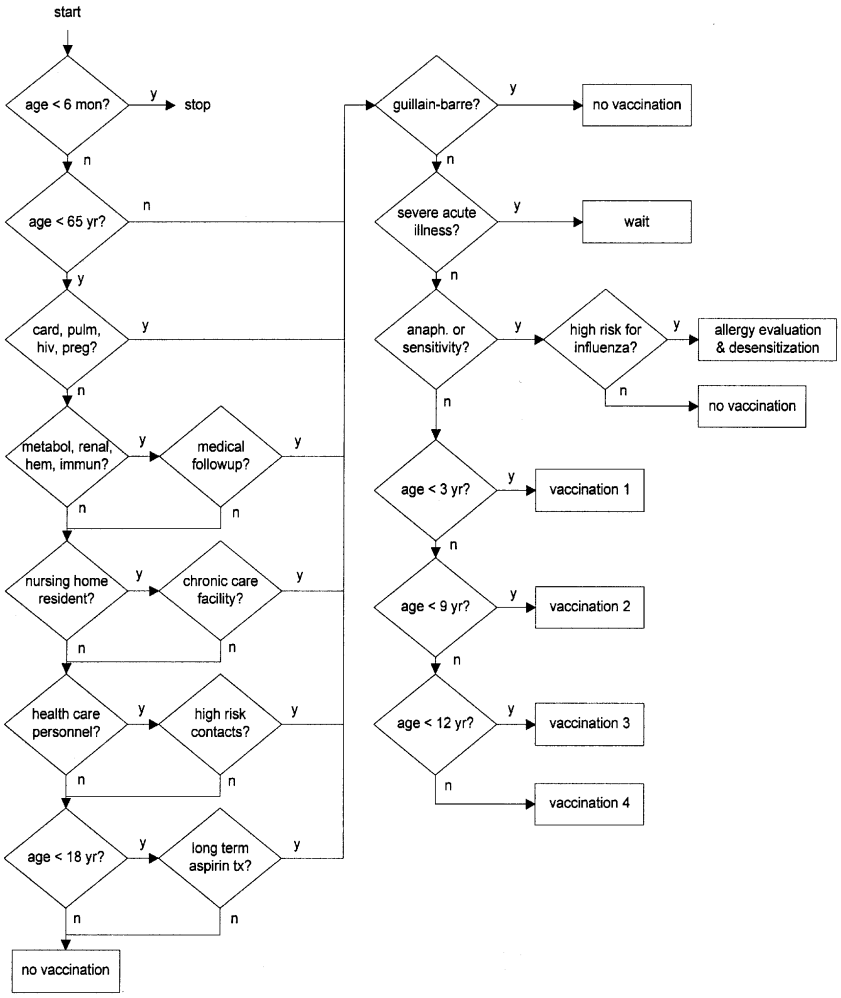


**Figure 6** A flowchart of the Flu guideline.

This reduces the total number of combinations produced for the Hib guideline to 21.

### How Many Constraints Are Enough?

The Hib guideline illustrates that there is considerable flexibility in imposing constraints to restrict the set of combinations of conditions that T/Gen generates. A number of questions arise. For example, it is not clear that using more constraints is always better. For the purposes to testing the logic, it may not be desirable to restrict the combinations only to those that make clinical sense. It may be useful to test the logic with a number of nonsensical cases.

Also, after a fair number of constraints have been imposed, some combinations may still be redundant. There is a tradeoff between the complexity of the constraints and the number of combinations generated. At some point it may be easier to look at a few extra cases rather than work very hard to eliminate as many redundant combinations as possible. We discuss these issues further later in the paper.

### T/Gen and the DTP Guideline

The DTP guideline has a structure similar to that of the Hib guideline. For DTP, an unconstrained use of T/Gen generates 5,120 combinations, which can be reduced to 22 combinations using constraints in a fashion similar to that described for Hib.

### T/Gen and the Flu Guideline

Figure 6 shows the flowchart of the pilot influenza (Flu) guideline, which is based on the 1995 version of recommendations by the CDC[23] and which has a very different structure from that of Hib and DTP.

In the Flu guideline, there are two phases to the logic. First, one set of decisions (on the left in Figure 6) considers whether the patient is a potential candidate for Flu vaccination. If so, the decisions on the right (phase 2) check for contraindications or special circumstance, and then determine which specific vaccination recommendation applies.

1. [age >= 6m], [age < 6m]
2. [age >= 65y], [age < 65y]
3. [card pulm HIV or pregnant], [not: card pulm HIV or pregnant]
4. [metabolic renal hem or immunosup], [not: metabolic renal hem or immunosup]
5. [medical follow-up], [not: medical follow-up]
6. [nursing home resident], [not: nursing home resident]
7. [chronic care facility], [not: chronic care facility]
8. [health care personnel], [not: health care personnel]
9. [high risk contacts], [not: high risk contacts]
10. [age < 18y], [age >= 18y]
11. [long term aspirin tx], [not: long term aspirin tx]
12. [guillain-barre post vacc], [not: guillain-barre post vacc]
13. [severe acute illness], [not: severe acute illness]
14. [vacc anaphylaxis or egg hypersens], [not: vacc anaphylaxis or egg hypersens]
15. [high influenza risk], [not: high influenza risk]
16. [age < 3y], [3y <= age < 9y], [9y <= age < 12y], [age >= 12y]

**Figure 7** **(left)** The 16 clinical conditions sets for the Flu guideline shown in Figure 6.

**Figure 8** **(below)** Condition sets for testing Phase 1 of the Flu guideline, as described in the text.

1. [age >= 6m]
2. [age < 65y]
3. [card pulm HIV or pregnant], [not: card pulm HIV or pregnant]
4. [metabolic renal hem or immunosup], [not: metabolic renal hem or immunosup] — second_only_if [card pulm HIV or pregnant]
5. [medical follow-up], [not: medical follow-up] — only_if [metabolic renal hem or immunosup]
6. [nursing home resident], [not: nursing home resident] — second_only_if [card pulm HIV or pregnant] or [metabolic renal hem or immunosup]
7. [chronic care facility], [not: chronic care facility] — only_if [nursing home resident]
8. [health care personnel], [not: health care personnel] — second_only_if [card pulm HIV or pregnant] or [metabolic renal hem or immunosup] or [nursing home resident]
9. [high risk contacts], [not: high risk contacts] — only_if [health care personnel]
10. [age < 18y], [age >= 18y] — second_only_if [card pulm HIV or pregnant] or [metabolic renal hem or immunosup] or [nursing home resident] or [health care personnel]
11. [long term aspirin tx] [not: long term aspirin tx] — only_if [age < 18y]
12. [not: guillain-barre post vacc]
13. [not: severe acute illness]
14. [not: vacc anaphylaxis or egg hypersens]
16. [age >= 12y]

In constructing this flowchart, we have abbreviated the description of most of the conditions. The details of each condition are not important for the purposes of this paper. In addition, certain conditions in this flowchart combine several clinical conditions in a single question. For example, the abbreviated condition "metabolic renal hem or immunosup" stands for "chronic metabolic disease or renal dysfunction or hemoglobinopathy or immunosuppression." This approach (combining several clinical conditions in a single T/Gen condition) is a useful strategy. If several clinical conditions can be combined into a single yes–no question, then T/Gen's operation will be more efficient, since many fewer combinations of conditions will need to be generated.

The condition sets for the Flu guideline are outlined in Figure 7. This guideline involves 15 condition sets with two values each and one condition set with four values. A brute-force, unconstrained approach would therefore generate $4 \times 2^{15} = 131{,}072$ combinations of conditions. Clearly, this is a completely unmanageable number of combinations for practical use in verification of the guideline. The question therefore arises whether domain constraints can enable us to limit these combinations to a manageable number. The flowchart in Figure 6 suggests several possible strategies.

- First, condition set 1 ("age >= 6m," "age < 6m") is not very interesting, since if the patient is under 6 months of age, the guideline simply shuts down. This condition does not need to be tested against every combination of the other clinical conditions.
- Also, as mentioned above, the guideline logic divides quite naturally into two phases—a) the logic that checks the patient's appropriateness for vaccination and b) the logic that applies only to patients for whom vaccination is potentially appropriate. The logic in these two phases might most efficiently be tested independently.
- In the first phase (on the left side of Figure 6), there are several steps in which an initial question is asked (e.g., "health care personnel?"), after which a second question (e.g., "high risk contacts?") determines whether vaccination is potentially appropriate. If so, control moves to the second phase of the guideline logic. It is clearly unnecessary to test all possible combinations of these pairs of values against all the other pairs multiple times.

In response to these considerations, we divide T/Gen's processing of this guideline into two parts, each testing one phase of the logic.

Phase 1

Figure 8 shows condition sets designed to test phase 1 of the logic. Here we have fixed the value of condition sets 1 and 2 (to be "age >= 6m" and "age < 65y," respectively), so that phase 1 will be executed. In addition, we have fixed the values of conditions 12 through 16, so that when control passes to phase 2, only a single path will be taken (since this logic is being tested separately). In addition, as shown in Figure 8, we have defined a set of constraints that have the result that the various pairs of conditions (as discussed above) are generated one at a time in a focused, non-repetitive fashion.

Four pairs of condition sets have the same general logical structure—that is, if the first is true, the second is tested to see if vaccination is appropriate. These pairs are condition sets 4 and 5, 6 and 7, 8 and 9, and 10 and 11. We will use condition sets 4 and 5 to discuss these constraints.

Condition set 4 is defined and constrained as follows:

4. [metabolic renal hem or immunosup],
   [not: metabolic renal hem or immunosup]
   — second_only_if [card pulm HIV or pregnant]

The effect of this constraint is that the second condition ("not: metabolic renal hem or immunosup") is the only member of this condition set included in a combination being generated if the condition "card pulm HIV or pregnant" is also included. This constraint prevents the proliferation of many combinations containing the first condition of condition set 4, "metabolic renal hem or immunosup."

Condition set 5 is defined and constrained as follows:

5. [medical follow-up], [not: medical follow-up]
   — only_if [metabolic renal hem or immunosup]

The effect of this constraint is that this condition set is ignored (and not included in any combination generated) unless its paired condition set (condition set 4 above) has the value that makes this condition set relevant. Using these constraints, the total number of combinations generated for phase 1 is reduced to 10.

Phase 2

Figure 9 shows a condition set designed to test phase 2 of the logic. Here we have fixed the value of conditions 1, 2, and 3 (to be "age >= 6m," "age < 65y," and "card pulm HIV or pregnant," respectively), so that analysis will proceed directly to the phase 2 logic. We then list the different combinations of conditions found in phase 2. Using T/Gen

1. [age >= 6m]
2. [age < 65y]
3. [card pulm HIV or pregnant]
12. [guillain-barre post vacc], [not: guillain-barre post vacc]
13. [severe acute illness], [not: severe acute illness] — second_only_if [guillain-barre post vacc]
14. [vacc anaphylaxis or egg hypersens], [not: vacc anaphylaxis or egg hypersens] — second_only_if [guillain-barre post vacc] or [severe acute illness]
15. [high influenza risk], [not: high influenza risk] — only_if [metabolic renal hem or immunosup]
16. [age < 3y], [3y <= age < 9y], [9y <= age < 12y], [age >= 12y]

**Figure 9** Condition sets for testing phase 2 of the Flu guideline, including constraints, as described in the text.

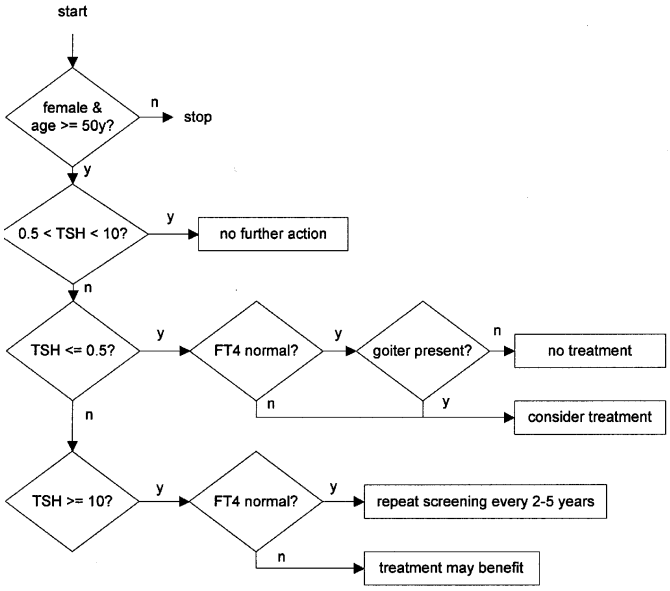


**Figure 10** A flowchart of the primary thyroid screening guideline.

1. [female]
2. [age >= 50y]
3. [TSH <= 0.5], [0.5 < TSH < 4.5], [TSH >= 10]
4. [FT4 test normal], [FT4 test not normal] — not_if [0.5 < TSH < 4.5]
5. [goiter present], [goiter not present] — not_if [0.5 < TSH < 4.5] or [TSH >= 10] or [FT4 test not normal]

**Figure 11** The condition sets for the primary thyroid screening guideline with constraints defined.

| T/Gen Generated Condition Set | Recommendation |
|---|---|
| 1. [female], [age >= 50y], [0.5 < TSH < 4.5] | [no further action] |
| 2. [female], [age >= 50y], [TSH <= 0.5], [FT4 test normal], [goiter present] | [consider treatment] |
| 3. [female], [age >= 50y], [TSH <= 0.5], [FT4 test normal], [goiter not present] | [no treatment] |
| 4. [female], [age >= 50y], [TSH <= 0.5], [FT4 test not normal] | [consider treatment] |
| 5. [female], [age >= 50y], [TSH >= 10], [FT4 test normal] | [repeat screening q 2-5 years] |
| 6. [female], [age >= 50y], [TSH >= 10], [FT4 test not normal] | [treatment may benefit] |

**Figure 12** Combinations generated for the PTS guideline with the constraints defined in Figure 11, also indicating the recommendations generated for each combination by processing the conditions through T/Gen's GLIF version of the guideline logic.

with these conditions with no constraints generates 64 combinations. If the constraints shown in Figure 9 are used, only 10 combinations are generated

Thus, the Flu guideline as a whole can be constrained to generate 20 combinations covering both guideline phases, 10 for each phase.

### T/Gen and the Primary Thyroid Screening Guideline

Figure 10 is a flowchart of the pilot guideline for primary thyroid screening (PTS).[24,25] This involves five sets of conditions, as shown in Figure 11. If the patient is not female or is not at least 50 years old, however, the logic terminates, so these conditions are fixed in the condition sets. The remaining three conditions, with no constraints, generate 12 combinations of conditions. The constraints reduce the number of combinations to six (Figure 12).

This guideline raises an interesting issue. Twelve combinations of conditions is not an unmanageable number. It is therefore not essential that these be reduced. By using the constraints, one reduces the combinations to those that are clinically meaningful, in the sense that they involve all combinations of conditions for which the guideline is designed to offer recommendations. For purposes of testing the logic, however, it may be useful to test a broader set of combinations, just to make sure that all are handled appropriately. In other words, it may not always be desirable to fully exploit the power of T/Gen's constraints.

### T/Gen and the Embryo Guideline

T/Gen was also used with a pilot guideline that makes recommendations as to the appropriate number of embryos to transfer when performing in vitro fertilization, an internal guideline developed at the Brigham and Women's Hospital. The use of T/Gen with this guideline is described in more detail in an appendix to this paper.* The embryo guideline illustrates a constraint that allows a multi-valued variable to assume only two values (one of its permissible values and the negation of that value) in specified conditions. Using two such "negate_if" constraints, T/Gen can reduce 24 combinations of conditions to 16 for the embryo guideline.

| Guideline | Combinations (unconstrained) | Combinations (constrained) |
|---|---|---|
| Hib | 2,048 | 21 |
| DTP | 5,120 | 22 |
| Flu | 131,072 | 20 |
| Thyroid | 12 | 6 |
| Embryo | 24 | 16 |

**Figure 13** For each of the five pilot guidelines, this figure summarizes the total number of combinations of conditions generated by T/Gen, with and without constraints.

## A Summary of the T/Gen Approach

Figure 13 provides a summary of T/Gen's ability to constrain the combinations of clinical conditions generated for the five pilot guidelines. This section summarizes the various issues introduced above in the context of a discussion of the pilot guidelines.

*The appendix appears online, as supplemental material to this article, at www. jamia.org.

### Structuring the Condition Sets

In preparing to use T/Gen, the first step is to prepare a list of conditions sets that are to be used. To accommodate the structure of a particular guideline, several possible strategies can be employed in this task:

- *Combining conditions.* If several clinical conditions can be combined into a single yes–no predicate, this should be done.
- *Separating phases.* If a complex guideline can be broken into components or phases that can be tested independently, this is a potentially powerful technique.
- *Holding certain conditions fixed.* If certain conditions are not used robustly in the guideline, they can be held fixed to a single value, thereby simplifying T/Gen's processing.

### Defining Constraints

Once the condition sets are defined, a variety of different constraints can be applied:

- Certain constraints result in a condition set being conditionally ignored (e.g., the "only_if" and "not_if" constraints).
- Other constraints result in a condition set being conditionally held to a single fixed value (e.g., the "second_only_if" constraint).
- Other constraints result in a condition set being conditionally reduced in scope to one value and its negation (e.g., the "negate_if" constraints).
- Post-generation constraints can be used to eliminate certain combinations after they have been generated.

As we continue to gain experience using T/Gen with a range of guidelines in different clinical domains, we anticipate that this spectrum of strategies and constraints may expand.

## Using T/Gen to Help Test Logical Correctness

The combinations of conditions that are produced by T/Gen are expressed as sets of logical predicates, i.e., expressions that each evaluate to either true or false. Examples include"Age < 12m?" and "TSH >= 10?" In contrast, a test case needs to contain actual patient values, for example "Age = 10m" or "TSH = 11.4."

It is not necessary, however, to convert T/Gen's logical expressions into precise patient values just to check the logic of the guideline. A list of combinations produced by T/Gen can be passed directly to T/Gen's Lisp-based GLIF interpreter and run through the GLIF version of the guideline to see what recommendations are produced for each combination.

As an example, Figure 12 shows the results of running the combinations of conditions produced for the PTS guideline through that guideline. In this figure, each combination is followed by the guideline recommendation. This recommendation was produced by executing the logic of the guideline, consulting the specific combination of conditions at each decision point to determine which step to take next. Directly executing the guideline logic in this fashion can assist in assessing the correctness of the guideline in several ways. (This testing might also discover logical errors in the execution engine.)

- The recommendations generated for each combination can be checked for accuracy.
- Combinations of conditions for which no recommendation is generated can be identified. If these correspond to clinically nonsensical combinations, additional constraints may be added to eliminate them, if desired. Alternatively, if the omission is an error, the guideline logic can be modified to handle the combination correctly.
- Any combinations that erroneously generate multiple recommendations can also be identified.

## Using T/Gen to Generate Test Cases

The combinations of clinical conditions produced by T/Gen can also form the basis for constructing a set of test cases. To do this, a set of patient values needs to be selected that correspond to the conditions listed in T/Gen's output. For certain domains, the construction of such test cases involves a straightforward mapping of conditions to values. For other domains, the process of constructing test cases is much more complex.

### The Primary Thyroid Screening Guideline: Simple Knowledge-mediated Mapping of Conditions to Test Cases

The primary thyroid screening guideline is an example of a very straightforward mapping between the conditions produced by T/Gen and the clinical values required in a test case. Figure 14 shows a set of test cases that was automatically produced from the PTS combinations shown in Figure 12. This test case generator chooses an arbitrary value that corresponds to each condition, as seen in Figure 14.

Even though the creation of these test cases is straightforward, there is still an opportunity to apply different strategies. For example, it is possible to construct several cases for a given combination, so that the range limits for each condition's values can be tested. For example, if the condition were "0.5 < TSH < 10," two cases might be created, one with TSH = 0.6 and one with TSH = 9.9. For the condition "TSH <= 0.5," a case with TSH = 0.4 and a case with with TSH = −0.1 might be created (which would test the lower limit of the range to see whether an error message was appropriately generated).

### The Hib Guideline: Complex Knowledge-mediated Mapping of Conditions to Test Cases

The Hib guideline is an example in which the mapping between T/Gen's conditions and a corresponding test case is not straightforward. The problem in this domain is that a previous Hib vaccination history (containing specific dates for each previous Hib vaccination) needs to be constructed. This cannot be done by means of a simple mapping from T/Gen's combination of conditions. Considerable domain knowledge is required. In addition, for this domain, it is potentially useful to construct several test cases for each combination of conditions.

Figure 15 shows a representative set of test cases produced for a specific T/Gen combination of generated conditions: "Hib3 not due," "Hib3_final due," "Prpomp indicated," and "Age < 12m." The process

PTSG: case 1
Sex: female
Age: 55
TSH: 3.2

PTSG: case 2
Sex: female
Age: 55
TSH: 0.3
FT4: normal
Goiter present

PTSG: case 3
Sex: female
Age: 55
TSH: 0.3
FT4: normal
Goiter not present

PTSG: case 4
Sex: female
Age: 55
TSH: 0.3
FT4: abnormal

PTSG: case 5
Sex: female
Age: 55
TSH: 12.4
FT4: normal

PTSG: case 6
Sex: female
Age: 55
TSH: 12.4
FT4: abnormal

**Figure 14** Sample test cases generated from the T/Gen combinations seen in Figure 11 for the PTS guideline.

Case 34: Hib dose 3, condition set 7, dose is due
# Conditions: Hib3 not due; Hib3_final due
# Conditions: Prpomp indicated; Age < 12m
Date Used for Forecast: 3/1/2000
Date of birth: 3/1/1999
Hib: 12/4/1999, 1/1/2000

Case 35: Hib dose 3, condition set 7,
too early—under minimum age
# Conditions: Hib3 not due; Hib3_final due
# Conditions: Prpomp indicated; Age < 12m
Date Used for Forecast: 3/1/2000
Date of birth: 3/2/1999
Hib: 12/4/1999, 1/1/2000

Case 36: Hib dose 3, condition set 7,
too soon after previous dose
# Conditions: Hib3 not due; Hib3_final due
# Conditions: Prpomp indicated; Age < 12m
Date Used for Forecast: 3/1/2000
Date of birth: 2/19/1999
Hib: 12/5/1999, 1/2/2000

Case 37: Hib dose 3, condition set 7,
long enough after previous dose
# Conditions: Hib3 not due; Hib3_final due
# Conditions: Prpomp indicated; Age < 12m
Date Used for Forecast: 3/1/2000
Date of birth: 2/19/1999
Hib: 12/4/1999, 1/1/2000

Case 38: Hib dose 3, condition set 7,
series complete
# Conditions: Hib3 not due; Hib3_final due
# Conditions: Prpomp indicated; Age < 12m
Date Used for Forecast: 3/2/2000
Date of birth: 3/1/1999
Hib: 12/4/1999, 1/1/2000, 3/1/2000

**Figure 15** Sample test cases generated from a T/Gen combination for the Hib guideline.

by which these test cases are constructed has been described in a previous paper.[14] The cases shown in Figure 15 test several facets of IMM/Serve's functionality. Specifically, the cases test whether the guideline responds correctly 1) when Hib3_final dose is due based on minimum age, 2) when the dose is not yet due based on minimum age, 3) when the dose is due based on the wait-interval from the previous dose, 4) when the dose is not yet due based on the wait-interval, and 5) when the series is complete.

## Current Status and Future Directions

The current implementation of T/Gen is a prototype system developed to explore the overall approach. We have explored these ideas in the context of five pilot clinical guidelines. We expect to explore other types of constraints by looking at different clinical domains. For example, constraints that deal with temporal logic would be particularly important for certain guidelines. In addition, we are actively using T/Gen to help maintain IMM/Serve's knowledge.

Looking to the future, we would like to develop T/Gen as a flexible tool that can be used by the clinical guideline community.

- We plan to create a Web interface that allows the user (a knowledge engineer) to define sets of conditions to which a clinical guideline must react, together with constraints defined on those conditions.
- Ideally, this Web-based system should also be able to extract a set of conditions from, for example, a GLIF-based clinical guideline and present those conditions to the user to work with.
- Using the Web interface, the user could work with the various conditions and constraints interactively, including or excluding condition sets and activating or deactivating different constraints.
- The condition sets generated could be automatically passed to a Web version of the guideline written in GLIF or in some other language or could be downloaded to the user for further analysis (e.g., conversion into test cases).
- The user should be able to store the various condition sets and constraints at the Web site. In this way, as the guideline logic evolves over time, the user could periodically return to T/Gen and update the conditions and constraints as needed, in an ongoing process of knowledge maintenance.

In this way, the Web version of T/Gen could be integrated into the knowledge maintenance process for many clinical guidelines being developed by different groups for different domains of clinical medicine. The current prototype version of T/Gen is a step in this direction.

## Discussion: Lessons Learned

This section discusses several of the lessons we have learned in the process of developing T/Gen and using it with the five pilot guidelines.

### The Need for a Flexible Interactive Tool to Assist in the Art of Guideline Maintenance

The design of T/Gen was informed by the philosophy that knowledge maintenance is an art, not a science. Our goal is to provide a set of methodologies and tools, but the knowledge engineer must use these creatively and interactively. Indeed, as mentioned in the Background section, the correctness of a program of even moderate complexity is inherently unprovable. It would, therefore, be impossible to develop a knowledge maintenance tool that can serve as a gold standard for testing a computer-based clinical guideline in any absolute sense.

We want to develop a set of strategies that can be used in a flexible fashion. We want the user to be able to vary sets of conditions and different constraints interactively. This process should have several benefits. It will help in testing and verifying the knowledge. It will also help the knowledge engineer better understand the nuances of the logic. This is particularly important in the context of knowledge maintenance, when the knowledge engineer has to return to the knowledge periodically (perhaps once a year) and in the meantime is likely to have forgotten many of its idiosyncrasies and intricacies.

### The Value of Cross-checking Two Forms of the Same Knowledge

One valuable feature of T/Gen's approach is that it provides the knowledge engineer with two forms of the same knowledge (the computer-based guideline logic and the constrained condition sets) that can be studied and refined independently and in concert. By manipulating these two forms of the guideline knowledge, the knowledge engineer is able to check one against the another.

It is difficult, in working with only one form of the knowledge in isolation, to appreciate its features and interconnections in a comprehensive fashion. We believe that having two forms of the knowledge that can be cross-checked provides the knowledge engineer with a much more robust vehicle for working with the knowledge in a powerful way.

### Tradeoffs Between the Number of Constraints and the Efficiency of Testing

As discussed previously, T/Gen offers considerable latitude for tradeoffs between the number of constraints defined and the efficiency of the testing

process. A small number of selected constraints can result in a dramatic reduction of combinations generated. Subsequent definition of constraints, however, may involve diminishing returns. Also, as discussed previously, for the purposes of discovering errors in the logic, it may not be desirable to maximally constrain the set of conditions generated.

The most reasonable approach may also depend on how the cases are to be checked. If the results of running each case through the logic need to be analyzed manually, then it will be important to limit the combinations produced. Sometimes, however, a new version of the guideline engine is tested, and the results of running it on test cases can be compared with the results obtained with the previous engine, in a fully automated fashion. In this situation, it could be useful to leave T/Gen fairly unconstrained and run thousands of test cases through the logic, using automated comparison to detect any errors in the new engine compared with the old.

In summary, T/Gen is a tool that can be used to help in the process of testing a computer-based clinical guideline. As we gain experience applying it with different guidelines, we will gain a more complete appreciation of how it is best used. The present pilot project is a first step in this direction.

The author thanks Aziz Boxwala, MD, PhD, and Robert A. Greenes, MD, PhD, of the Decision Systems Group at the Brigham and Women's Hospital, Boston, Massachusetts, and the national InterMED Collaboratory, for providing three of the pilot GLIF guidelines used in this work.

*References*

1. Ohno-Machado L, Gennari JH, Murphy SN, et al. The guideline interchange format: a model for representing guidelines. J Am Med Inform Assoc. 1998;5:357–72.
2. Miller PL, Frawley SJ, Sayward FG, Yasnoff WA, Duncan L, Fleming D. Combining tabular, rule-based and procedural knowledge in computer-based guidelines for childhood immunization. Comput Biomed Res. 1997;30:211–31.
3. Friedman CP, Wyatt JC. Evaluation Methods in Medical Informatics. New York: Springer-Verlag, 1997.
4. Adelman L. Evaluating Decision Support and Expert Systems. New York: Wiley, 1992.
5. Miller PL, Sittig DF. The evaluation of clinical decision support systems: what is necessary vs. what is interesting. Med Inform. 1990;15:185–90.
6. Perlis AJ, DeMillo RA, Lipton RJ. Social processes and proofs of theorems and programs. Conference Record of the 4th ACM Symposium on Principles of Programming Languages. New York: ACM Press, 1997:206–14.
7. Beizer B. Software Testing Techniques. New York: Van Nostrand Reinhold, 1990.
8. Veevers A, Marshall AC. A relationship between software coverage metrics and reliability. J Software Testing, Verification, and Reliability. 1994;4(1):3–8.
9. Marick B. The Craft of Software Testing. Upper Saddle River, NJ: Prentice Hall, 1995.
10. Roper M. Software Testing. London, UK: McGraw-Hill, 1994.
11. Howden W. Weak mutation testing and completeness of test sets. IEEE Trans Software Eng. 1982;SE8(4):371–9.
12. Turner CD, Robson DJ. The state-based testing of object-oriented programs. Proceedings of the Conference on Software Maintenance. Los Alamitos, Calif: IEEE Computer Society Press, 1993:302–10.
13. Frankel P, Weyuker E. An applicable family of data flow testing criteria. IEEE Trans Software Eng. 1988;14(10):1483–98.
14. Miller PL, Frawley SJ, Brandt C, Sayward FG. Tools for immunization guideline knowledge maintenance II: automated Web-based generation of user-customized test cases. Comput Biomed Res. 1998;31:190–208.
15. Miller DW, Frawley SJ, Miller PL. Using semantic constraints to help verify the completeness of a computer-based clinical guideline for childhood immunization. Comput Methods Programs Biomed. 1999;58:267–80.
16. Jenders RA, Huang H, Hripcsak G, Clayton PD. Evolution of a knowledge base for a clinical decision support system encoded in the Arden Syntax. Proc AMIA Annu Symp. 1998:558–62.
17. Kuperman GJ, Fiskio JM, Karson A. A process to maintain the quality of a computerized knowledge base. Proc AMIA Annu Symp. 1999:87–91.
18. Geissbuhler A, Miller RA. Distributing knowledge maintenance for clinical decision-support systems: The "knowledge library" model. Proc AMIA Annu Symp. 1999:770–4.
19. Suwa M, Scott AC, Shortliffe EH. An approach to verifying completeness and consistency in a rule-based system. AI Magazine. 1982;3:16–21.
20. Gotlieb A, Botella B, Rueher M. Automatic test data generation using constraint solving techniques. Proceedings of the International Symposium on Software Testing and Analysis '98. New York: ACM Press, 1998:53–62.
21. Chang J, Richardson DJ. Structural specification-based testing with ADL. Proceedings of the International Symposium on Software Testing and Analysis '96. New York: ACM Press, 1996:62–70.
22. Peters D, Parnas DL. Generating a test oracle from program documentation. Proceedings of the International Symposium on Software Testing and Analysis '94. New York: ACM Press, 1994:58–65.
23. Centers for Disease Control and Prevention. Prevention and control of influenza: recommendations of the Advisory Committee on Immunization Practices (ACIP). MMWR Morb Mortal Wkly Rep. 1995;44(RR3):1–22.
24. Clinical Guideline, Part 1: Screening for Thyroid Disease. American College of Physicians. Ann Intern Med. 1998;129:141–3.
25. Helfand M, Redfern CC. Clinical Guideline, Part 2: Screening for Thyroid Disease: An Update. American College of Physicians. Ann Intern Med. 1998;129:144–58.