GENERATING MEDICAL LOGIC MODULES

FOR CLINICAL TRIAL ELIGIBILITY

by

Craig Gerold Parker

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

November 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Craig Gerold Parker

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	David W. Embley, Chair
Date	Deryle W. Lonsdale
Date	William A. Barrett

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Craig Gerold Parker in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

David W. Embley
Chair, Graduate Committee

Accepted for the Department

Parris K. Egbert
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

GENERATING MEDICAL LOGIC MODULES

FOR CLINICAL TRIAL ELIGIBILITY

Craig Gerold Parker

Department of Computer Science

Master of Science

Clinical trials are important to the advancement of medical science. They provide the experimental and statistical basis needed to determine the benefit of diagnostic and therapeutic agents and procedures. The more patients enrolled in a clinical trial, the more confidence we can have in the trial's results. However, current practices for identifying eligible patients can be expensive and time-consuming. To assist in making identification of eligible patients more cost effective, we have developed a system for translating the eligibility criteria for clinical trials to an executable form. This system takes as input the eligibility criteria for a trial formatted as first order predicates. We then map these criteria against concepts in a target database. The mapped criteria are output as an Arden Syntax medical logic module using virtual medical record queries in the curly braces. The system was able to successfully process 85 out of 100 trials attempted. From these 85 tri-

als, the system idendified 1,545 eligibility criteria. From these criteria, we generate 520 virtual medical record queries, 253 of which were deemed useful in helping to determine eligibility.

ACKNOWLEDGMENTS

I thank my committee for their guidance and friendship. I thank Intermountain Health Care for broad access to resources and tolerance of this significant distraction from my daily duties. I thank my children for asking me every day if "it" was done yet. And I thank my wife for enduring more than a year's worth of "it'll be done in a couple of weeks."

Table of Contents

List of Tables

List of Figures

1 - Introduction

Clinical trials are important for medical research. They provide the experimental and statistical basis needed to determine the benefit of diagnostic and therapeutic agents and procedures. As a basic principle of statistics, the more people that can be enrolled in a clinical trial, the greater the confidence we can have in the results of the trial. However, it can be difficult to identify a significant number of patients who meet the criteria for participation. This is because trials often have very specific criteria for age, gender, state of a given disease, number and types of co-existing diseases, etc.

There are many ways to identify patients who are eligible for clinical trials. One commonly used method is for the clinicians who are participating in the trial to evaluate each patient they see in their clinic for eligibility. The advantages of this method include: (1) The workflow of the clinician is only minimally disturbed. (2) The clinician generally has an up-to-date picture of the patient's health conditions. (3) For any eligibility criteria that the clinician is unsure about, the patient is present for questioning or examination. The biggest disadvantage of this method is the fact that it only identifies patients who happen to have a clinic visit with a participating clinician immediately prior to or during the enrollment phase of the trial.

Another common method for identifying candidates is through advertisements distributed via television, radio, the internet, newspapers or magazines. These advertisements usually present a number of eligibility criteria and a method for contacting someone who can further evaluate their eligibility. The main advantage of this approach is that it can screen a large number of people, including people who would not have normally visited a clinician's office during the enrollment period. One of the obvious drawbacks of this method is the cost of advertising. Another is that the criteria must be presented in a manner understandable by individuals without medical training. This often means that many people who may meet the criteria presented in the advertisement will not be eligible for the trial when evaluated against the detailed trial criteria by a clinician. Finally this method usually requires a clinician to spend significant time evaluating potential trial enrollees. This is time that must be allocated outside of their normal clinic schedule and may present a significant impact on their practice.

A third method that is commonly used for identifying candidates is to review medical records looking for patients that may meet the eligibility criteria. As with advertising, this method can find individuals who are eligible, but may not have normally visited a clinic during the trial's enrollment. It also has the advantage that many of the details of the patient's medical status are available to the screener. In addition, the screener usually has some amount of clinical training. However, searching through medical records can be a laborious task, and the cost of hiring someone with medical training to do this can be significant. In addition, the information available may be out-of-date causing some eligible patients to be missed, and some ineligible patients to be evaluated further.

As more and more patient-specific medical data is stored in electronic medical records, a variation on this third approach is becoming increasingly feasible. Automated processes could be developed to sift through the available data and identify patients who are likely to be eligible for a given trial. For trials with simple eligibility criteria that correspond well with clinical observations that are commonly captured and recorded electronically, an automated system may be able to determine eligibility directly. In the more

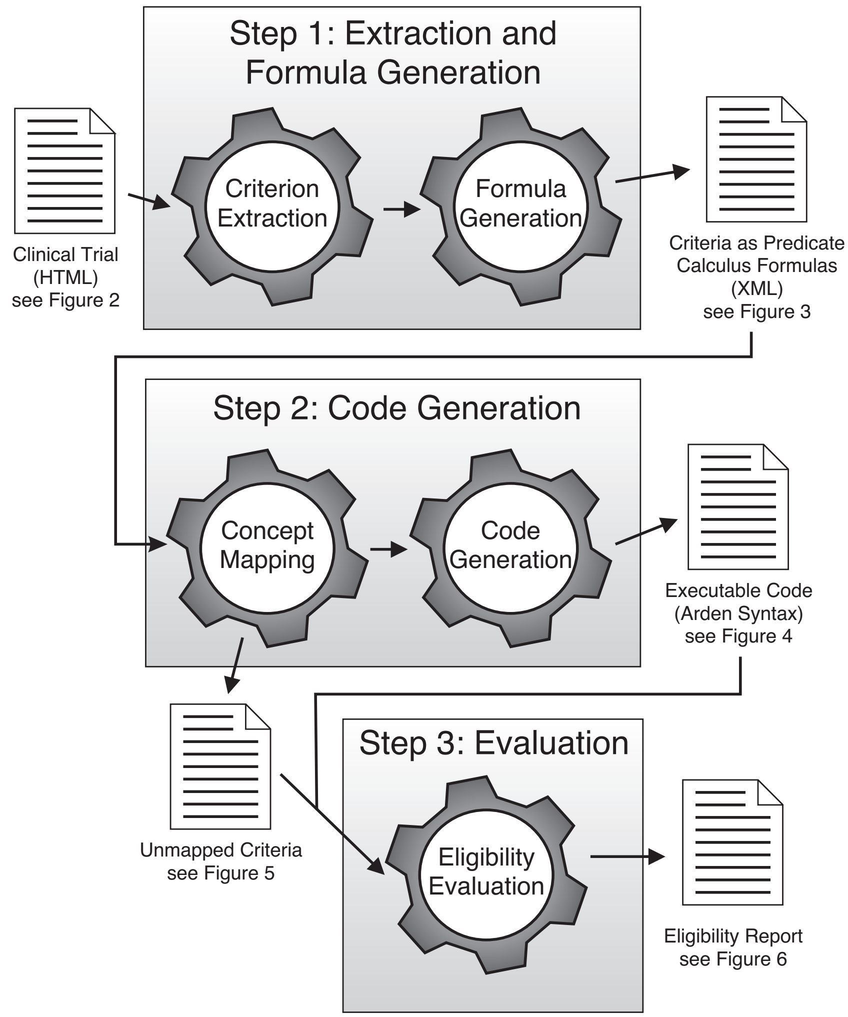

Figure 1 - Process for automatically evaluating clinical trial eligibility criteria.

common case where the criteria may be more complex and the corresponding clinical observations are not guaranteed to be available electronically, an automated system may still add valuable assistance by reducing the number of patients that would need to be evaluated manually.

We have developed an automated process for transforming natural language eligibility criteria into an executable form which can assist in identifying potential candidates for participation in a clinical trial. Figure 1 illustrates the process. We divide the process into three steps: Extraction and Formula Generation, Code Generation, and Evaluation.

Comparing Effects of 3 Sources of Garlic on Cholesterol Levels

Purpose:

The purpose of this study is to determine whether fresh garlic can positively affect cholesterol in adults with moderately high cholesterol levels. This study will also determine whether the same effects can be found for two main types of garlic supplements: a dried powdered garlic (designed to yield the same effect as fresh garlic) and an aged garlic extract preparation.

. . .

Eligibility:

Ages Eligible for Study: 30 Years - 65 Years
Genders Eligible for Study: Both

Inclusion Criteria:

- LDL-C 130-190 mg/dL
- BMI 19-30 kg/m2
- Weight stable for last 2 months
- Not actively on a weight loss plan
- Ethnicity representative of local population
- No plans to move from the area over the next 9 months

Exclusion Criteria:

- Pregnant, lactating, within 6 months postpartum, or planning to become pregnant in the next year
- Diabetes (type I or II) or history of gestational diabetes
- Heart disease
- Active neoplasms
- Renal or liver disease
- Hyperthyroidism or hypothyroidism
- Lipid lowering medications (known to affect lipid metabolism, platelet function, or antioxidant status)
- Blood pressure medications
- Excessive alcohol intake (self reported, more than 3 drinks/day)
- Currently under psychiatric care or severely clinically depressed

Location and Contact Information:

. . .

Figure 2 - A sample clinical trial.

```
<criteria trial=”http://www.clinicaltrials.gov/ct/show/NCT00056511”>
  . . .
  <criterion>
    <text>Inclusion Criteria</text>
    <text>LDL-C 130-190 mg/dL</text>
    <formula>
      ldl-c(N1) &amp; greater_than_or_equal(N1,N2) &amp;
      measurement(N2) &amp; magnitude(N2,130) &amp;
      units(N2,N3) &amp; mg/dl(N3) &amp;
      less_than_or_equal(N1,N4) &amp; measurement(N4) &amp;
      magnitude(N4,190) &amp; units(N4,N3)
    </formula>
  </criterion>
  . . .
  <criterion>
    <text>Exclusion Criteia</text>
    <text>Heart disease</text>
    <formula>heart_disease(N1)</formula>
  </criterion>
  . . .
  <criterion>
    <text>Exclusion Criteia</text>
    <text>No plans to move from the area over the next 9 months</text>
    <formula>Not Parsed</formula>
  </criterion>
  . . .
</criteria>
```

Figure 3 - Extracted eligibility criteria with predicate calculus formulas.

In Step 1, Extraction and Formula Generation, we extract eligibility criteria from a natural language description and transform them into first-order predicate calculus formulas. Figure 2 shows selected parts of a real clinical trial [Tust04], including the eligibility criteria section which is divided into sections for inclusion and exclusion criteria. (The complete trial appears in Appendix A.) The HTML source of a trial such as this is the input to Step 1. Figure 3 shows the output of Step 1 for three of the criteria in Figure 2. This example illustrates successful parsing of two of the criteria into predicate calcu-

```
maintenance:
  . . .
library:
  . . .
knowledge:
  type:
    data-driven;;
  data:
    . . .
    /* query for ldl-c */
    Criterion3 := READ {<VMRQuery> . . . </VMRQuery>};
    . . .
    /* query for heart disease */
    Criterion11 := READ {<VMRQuery> . . . </VMRQuery>};
    . . .
  logic:
    matches := 0;
    . . .
    if Criterion3 is present then matches := matches + 1;
    . . .
    if Criterion11 is present then matches := matches + 1;
    . . .
    write “Patient meets “ || matches || “ out of 18 criteria.”;
end;;
```

Figure 4 - Sample logic in the Arden Syntax [HCP+90] for determining eligibility.

lus formulas, as well as the output for a criterion that was not successfully parsed into a formula. The details of Step 1 are the subject of another thesis [Tus04] and are described only at a high level in this thesis.

Step 2, Code Generation, is the focus of this thesis. In this step, the system reads in parsed criteria and their predicate calculus formulas from Step 1 (see Figure 3). The system then attempts to map the criteria to concepts in an electronic medical record. For the criteria that are successfully mapped, the system outputs appropriate logic for computing whether or not a patient meets each criterion as Figure 4 illustrates. Since the

```
<MappingReport trial="http://www.clinicaltrials.gov/ct/show/NCT00056511">
 . . .
 <criterion>
   <text>Inclusion Criteia</text>
   <text>No plans to move from the area over the next 9 months</text>
   <criterionNotParsed/>
 </criterion>
 . . .
 <criterion>
   <text>Exclusion Criteria</text>
   <text>Active neoplasms</text>
   <formula>active(N1) &amp; neoplasms(N1)</formula>
   <criterionNotMapped/>
 </criterion>
 . . .
</MappingReport>
```

Figure 5 - Report of unmapped criteria.

system cannot always map all criteria, it also creates a document listing the unmapped criteria. Figure 5 shows an example of this output, illustrating both a criterion that was not parsed into a predicate-calculus formula in Step 1, as well as a criterion that was parsed in Step 1 but not mapped in Step 2.

In Step 3, Evaluation, the system evaluates the eligibility of a patient by executing the logic generated in Step 2 against that patient's electronic medical record. The system presents the result of this evaluation, along with a report of unmapped criteria to the user. Figure 6 shows an example of how this report may look. Based on the information presented by the system, the user can make an informed decision about whether to further evaluate the patient for enrollment in the clinical trial. Due to patient privacy issues, evaluation of patient data is beyond the scope of this thesis.

The system described in this thesis combines computer science and medicine to present a new solution to the problem of finding patients who are eligible to participate in

Eligiblity Report

Header	
Title of Trial	**Comparing Effects of 3 Sources of Garlic on Cholesterol Levels**
Patient Name	J. Doe
Medical Record #	1234567

Eligibility Summary	
Criteria met	**6**
Mapped criteria for which eligibility could not be determined	7
Criteria not mapped	5
Total criteria	**18**

Criterion Detail	
Criterion 1	
. . .	
Criterion 3	
Criterion	**LDL-C 130-190 mg/dL**
Mapped	Yes
Status	Patient meets this criterion
Criterion 4	
. . .	
Criterion 8	
Criterion	**No plans to move from the area over the next 9 months**
Mapped	No
Status	Unable to determine if patient meets this criterion
Criterion 9	
. . .	
Criterion 11	
Criterion	**Heart disease**
Mapped	Yes
Status	Unable to determine if patient meets this criterion
Criterion 12	
Criterion	**Active neoplasms**
Mapped	No
Status	Unable to determine if patient meets this criterion
Criterion 13	

Figure 6 - Sample eligibility report.

clinical trials. Many individuals and organizations have created a number of technologies and resources to bridge these disciplines. Chapter 2 presents background information on those technologies and resources that we use.

In Chapter 3 we describe the design of the system. The focus is on Step 2 of Figure 1, but Steps 1 and 3 will also be covered briefly. To evaluate the system, we applied it to a set of clinical trials. In Chapter 4 we describe our method of evaluation and the results. We also discuss the reasons behind the results and conclusions we draw from the results. This system represents a first approach to this problem. In Chapter 5 we look at ways the system could be enhanced in the future to provide better results.

2 - Background Information

In this chapter we discuss some resources and technologies that are commonly used in medical information systems and that we use in this project. In particular, we explain how these systems represent medical information in a computable form using a combination of medical vocabularies, data models, and languages for expressing medical logic.

2.1 - Coded Concepts

Medical information systems manage information that health care organizations need to care for patients, do administrative tasks, and meet regulatory requirements. These systems vary widely both in the breadth and the life cycle of information they handle. Narrowly focused systems may deal only with information related to a single disease or specialty of medicine. These systems may only be concerned with one or a few episodes of care. Such systems generally have fewer requirements for the representation of the information they manage. They need only what is sufficient for a specific task. A specialized clinical note application, for example, may only need to faithfully store and retrieve free text entered by the user. Such a system may need a few discrete data items such as a user identifier, a time stamp, and a note type, but beyond this, it may be sufficient to handle everything else as an unstructured text field.

On the other end of the spectrum are comprehensive electronic medical record systems. These systems strive to capture any information that may be clinically relevant

```
observation:
	type = “family history observation”
	value = “colon cancer”
	negation_indicator = “negated”
```

Figure 7 - Psuedocode data structure for the statetment, “the patient does not have a family history of colon cancer.”

and support a broad range of tasks longitudinally through time. This information may be used in many different ways including display back to the user, identification of clinical information to support billing, queries about the condition of a given patient, and population queries across patients. Therefore, these systems must represent information in more flexible and generalized ways.

To enable these diverse uses of clinical information, such systems collect and store information in a highly structured form. It takes significant effort to design and maintain this type of structured data, but the benefit from the resulting flexibility is great. Consider, for example, the statement, “The patient does not have a family history of colon cancer.” If this statement is stored as a text string, it is useful for displaying back to a human at some point in the future. However, for an automated process to use the information, the statement would need to be enhanced by some mechanism such as natural language processing. While natural language processing can be useful in medicine (indeed this project makes use of it), its reliability is not sufficient for many medical uses.

It is much easier for automated processes to use information that is captured and recorded in a structured format. The example above could be represented by a data structure with a field for the type of observation (“family history observation”), the value of the observation (“colon cancer”), and a negation indicator (“negated”) as Figure 7 illustrates. While this approach is more computable, it requires more effort in defining the data structures and the data capture methods associated with the data structures. In addition, structured data entry requires more effort on the part of the user to think about the structure of the information and enter it appropriately. In medicine the benefit of structured data entry frequently outweighs the burden and medical application developers are increasingly designing their systems around structured data.

The use of coded medical vocabularies greatly facilitates this approach. Coded vocabularies consist of a set of concepts, each of which has a unique identifier or code. The code “254837009“ in the SNOMED-CT[SC98] coded vocabulary, for example, represents the concept “breast cancer.” Often the coded vocabularies organize concepts into logical generalization/specialization hierarchies. For example, concepts for “penicillin” and “erythromycin” are specializations of the concept for “antibiotic.” Frequently, the coded vocabularies also provide other information about each concept such as synonyms, definitions, and relationships with other concepts. A concept in a coded vocabulary is called a *coded concept*. In this thesis we will represent coded concepts in the form, <code | code system | text>. For example, we will refer to the concept for breast cancer in SNOMED-CT as <254837009 | SNOMED-CT | breast cancer>.

Coded concepts facilitate a consistent representation for medical information. This makes it easier to share information between different systems while maintaining meaning. Coded concepts are convenient for automated medical applications because they are less prone to lexical errors such as misspellings or one phrase having more than one meaning depending on its context. For example, the word “fundus” may be associated with a portion of the eye, the stomach, or the uterus, depending on the context in which it is used. For each of these uses, the coded vocabulary would define a distinct concept with its own identifier. In SNOMED-CT the concepts are: <65784005 | SNOMED-CT | fundus of eye>, <414003 | SNOMED-CT | fundus of stomach>, and <27485007 | SNOMED-CT | fundus of uterus>.

```
diagnosis:
        has-required-field:
                name = “type”
                type = coded concept
        has-optional-field:
                name = “subject”
                type = coded concept
```

Figure 8 - Psuedocode definition for a detailed clinical model for a diagnosis.

2.2 - Detailed Clinical Models

While coded vocabularies provide much of the raw material needed to describe clinical information, they are not sufficient alone. If we want to state that a patient has a diagnoisis of breast cancer, we could store the concept, <254837009 | SNOMED-CT | breast cancer>, in her electronic medical record. If we wanted to state that the patient had a family history of breast cancer, we could store the concept, <275862002 | SNOMED-CT | family history of breast cancer>, in her record. Suppose now that we wanted to store the fact that it was the patient’s sister that had breast cancer. Currently SNOMED-CT does not have a concept for this. Although a coded vocabulary like SNOMED-CT could add concepts like this, it is not a practical solution. It would require the maintainers of the vocabulary to create concepts for most combinations of disease and family members.

A solution to this problem is to use detailed clinical models. A detailed clinical model is a data model that defines relationships between coded concepts or other data values to describe information of clinical interest. For example, a detailed clinical model may define a diagnosis as something that has a type and a subject as in Figure 8. This defintion states that a “diagnosis” has two fields. The first field is named “type” and contains a value that is a coded concept. This field is required. The second field is named “subject,” meaning the subject of the diagnosis, or who has this diagnosis. The value of

```
diagnosis:
        type: <254837009 | SNOMED-CT | breast cancer>

diagnosis:
        type: <254837009 | SNOMED-CT | breast cancer>
        subject: <27733009 | SNOMED-CT | sister>
```

Figure 9 - Psuedocode instances of detailed clinical models.

this field is also a coded concept. This field is optional; if it is not present, the subject of the diagnosis is assumed to be the patient. Figure 9 shows two data instances. The first one asserts that the patient has a diagnois of breast cancer, and the second one asserts that the patient's sister has a diagnosis of breast cancer. By defining and using detailed clinical models, we are able to combine coded concepts into meaninful expressions. This allows us to efficiently describe clinical information. We make extensive use of both coded concepts and detailed clinical models in the Concept Mapping process shown in Step 2 in Figure 1.

2.3 - Intermountain Health Care's Electronic Medical Record

The target electronic medical record for this project is Intermountain Health Care's Clinical Data Repository (CDR)[CDR]. Intermountain Health Care (IHC) is a regional, nonprofit, integrated health system based in Salt Lake City, UT. The CDR is the result of a joint development effort between IHC and 3M Health Information Systems. The CDR is a robust electronic medical record system which makes extensive use of coded vocabularies and detailed clinical models.

The detailed clinical models used by the CDR are defined using Abstract Syntax Notation One (ASN.1)[HRS+98]. ASN.1 is an ISO standard for describing electronic messages[ASN]. As its name implies, ASN.1 provides a syntax for describing messages

```
LabResult ::= SET {
        labTestname          [0] CodedConcept,
        labTestValue         [1] REAL,
        unitsOfMeasure       [2] CodedConcept }

CodedConcept ::= SET {
        code            [0] VisibleString (SIZE (1..20)),
        codeSystem  [1] VisibleString (SIZE (1..255)),
        text             [2] VisibleString (SIZE (1..255)) }
```

Figure 10 - A simple ASN.1 definition for a laboratory result.

that is abstract from any specific encoding. However, in addition to the abstract specification, ASN.1 also defines multiple specifications for specific encodings, including binary and XML encodings. Thanks to its flexibility and efficiency, ASN.1 is used in many different areas ranging from telecommunications to genome databases.

The best analogies for understanding what ASN.1 is and how it works are nested *structs* in the C programming language and XML. All three are tools for defining nested data structures where each field in the structure can have a name and a type. All three tools have distinct concepts for definitions and instances. The biggest difference between ASN.1 and the others is that the defintions specify an abstract model independent of a given representational technology. This means that while instances of C *structs* are always regions of memory and instances of XML are always text documents, instances of ASN.1 can be represented in many different forms depending on the chosen encoding rules. Since all of the encodings are representationally complete with respect to the abstract model, they are interchangable. Figure 10 gives an example of an ASN.1 definition for a simple detailed clinical model. This figure also illustrates that the type of each item in an ASN.1 definition may be a primitive (e.g. REAL) or it may be the result of another definition (e.g. a CodedConcept).

All coded concepts in the CDR are drawn from IHC's Healthcare Data Dictionary (HDD)[RHHW94], another technology jointly developed by IHC and 3M. The HDD is effectively a large coded vocabulary (over 800,000 concepts with over 4 million synonyms) containing both locally defined concepts and concepts from other coded vocabularies. The names of all the detailed clinical models used in the CDR and the fields they contain are defined as concepts in the HDD. The result is that all data stored in the CDR can be viewed as name-value pairs. The name portion of the pairs are always coded concepts. The values can be either primitive data types or other detailed clinical models. For example, given the following coded concepts:

```
<1155 | HDD | Lab Observation>
<552 | HDD | Lab Test Name>
<10220 | HDD | Serum Sodium>
<90753 | HDD | Lab Test Value>
<1110 | HDD | Units of Measure>
<1729 | HDD | mEq/L>
```

a lab result for a serum sodium could be represented as:

```
{ Lab Observation : {
     {Lab Test Name : Serum Sodium},
     {Lab Test Value : 140},
     {Units of Measure : mEq/L} } }
```

or by replacing the text names with the appropriate codes we simply have:

```
{1155 : {{552 : 10220},{90753 : '140'},{1110 : 1729}}}.
```

Note that this is not an actual ASN.1 encoding. IHC uses the ASN.1 Basic Encoding Rules (BER) to encode its clinical data. BER is a binary encoding. The example above is an approximate textual representation for the binary encoding. This encoding can be viewed as a series of name-value pairs. The values immediately preceding the

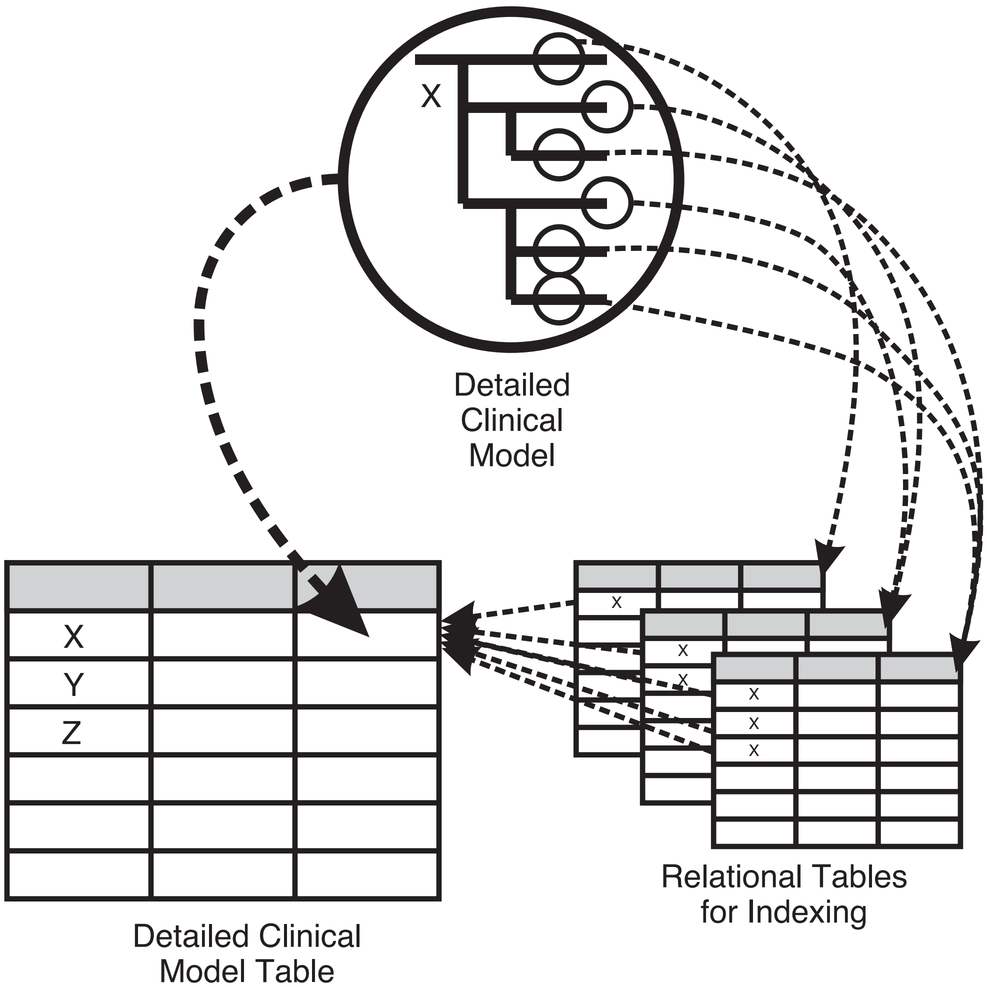

Figure 11 - How detailed clinical models are stored in IHC's CDR.

colons are concepts from the HDD and are the "name" portion of the name-value pairs. Whatever comes after the colon is the "value" portion. Notice that the value can be a coded concept from the HDD, a primitive data type such as a numeric value or a string, or a composite of other name-value pairs.

The CDR is made up of a database and a set of services that operate on the database. For the most part, data in the database is only accessed through the services. The services perform a couple of functions. First, they provide a common access mechanism to ensure consistent security, auditing, and error handling. Equally important is the way the services handle detailed clinical models. To applications built on the services, the CDR behaves more like an object-oriented database than a relational database. The applications pass instances of detailed clinical models to the services and get other instances of detailed clinical models back. Internally, the data is actually stored in a relational database, but this fact is almost completely hidden from any application.

Although the underlying database is relational, the data is not stored in a traditionally normalized relational manner. Instead the CDR has one table where the services store each instance of a detailed clinical model, formatted as an ASN.1 BER string. Every row in this table has a binary field that holds a BER string. Other fields in each row provide information for indexing purposes such as a patient identifier. In addition, the services shred the BER strings into another small set of tables. These tables are used for indexing purposes. In effect, all of the data in the CDR is stored twice, once in the BER string and once in the relational tables, as Figure 11 illustrates. This allows the services to do fast indexed searches in the relational tables to identify the detailed clinical models of interest. They can then read back the entire instance with a single row read instead of the large number of joins it would take to reconstitute the models if they were stored only in a normalized relational format. This is advantageous because applications commonly need the entire detailed clinical models rather than just the information present in a single row of a relational table.

2.4 - Arden Syntax

One of the outputs of Step 2 in Figure 1 is executable logic in Arden Syntax. Arden Syntax [HCP+90] was developed in 1990 as a language for encoding medical knowledge. It was developed in an attempt to address the need to share medical knowledge between hospitals and other medical institutions. Arden Syntax is currently maintained by the Health Level Seven (HL7) Arden Syntax Special Interest Group and is an ANSI standard. Many vendors of electronic medical records have implemented Arden compilers in their systems.

Arden Syntax is written in units called medical logic modules (MLMs). Each MLM contains the logic necessary for making one medical decision. Portions of an MLM are shown in Figure 4. An Arden Syntax MLM is made up of categories and slots. The three categories in Arden Syntax are the maintenance category, the library category, and the knowledge category. Each category contains a list of slots. The slots in the mainenance category contain information related to knowledge base maintenance and change control. The maintenance category does not contain any clinical information. The slots in the library category describe the sources of information used in creating the MLM, keywords, and related information.

The knowledge category of an MLM is where the clinical logic is represented. The most significant slots in this category are the data slot and the logic slot. The data slot contains mappings of symbols used in an MLM to data in the target electronic medical record. The logic slot, as its name implies, contains the logic that operates on the data. Figure 12 shows an outline of the structure of an MLM.

While Arden Syntax is the best option currently available for sharing medical logic across institutions, it suffers from what is known as the “curly braces problem.” Arden Syntax does not specify a notation for referencing data elements in the target electronic medical record (EMR). Rather, such references are written in a form that is understood by the native EMR and placed inside curly braces (e.g. the curly braces may contain a SQL statement specific to a given EMR). This means that while the logic of the module should be portable from one EMR to the next, the references to the data in the EMR are not portable. One proposed solution to the “curly braces problem” is to use and abstraction call the virtual medical record (VMR) .

Maintenance

- Title
- MLM name
- Arden Syntax version
- Version
- Institution
- Author
- Specialist
- Date
- Validation

Library

- Purpose
- Explanation
- Keywords
- Citations
- Links

Knowledge

- Type
- Data
- Priority
- Evoke
- Logic
- Action
- Urgency

Figure 12 - An outline of the categories and slots that make up an Arden Syntax MLM.

2.5 - The Virtual Medical Record

A VMR is an abstraction of a data model for a medical record [PRC+04]. It is intended that decision logic can be written against a VMR and then distributed to any number of healthcare organizations, each possibly using a different EMR. Each EMR would have a mapping to the VMR and would therefore be able to translate VMR logic into native queries. In the MLM in Figure 4, curly braces follow the “READ” keyword. In this case the curly braces contain an abbreviated snippet of XML representing a VMR query. The specification of a standard VMR is a current effort of the Clinical Decision Support Technical Committee of HL7.

The VMR that we use in this project is based on some early work from HL7. This VMR consists of a small set of classes that describe clinically relevant information. These classes include Observation, SubstanceAdministration, and Encounter. Each class has a number of attributes. For example the Observation class has a “code” attribute that specifies the type of the observation, a “value” attribute, and other attributes for capturing information such as the timing and status of the observation. In this project we limit our VMR queries to queries on the code and value attributes of the Observation class. This small subset of the VMR captures a large majority of the information needed to determine clinical trial eligibility.

3 - System Design

The focus of this thesis is the process of transforming first order predicate formulas describing clinical trial eligibility criteria into an executable form. This corresponds to Step 2 of Figure 1, Executable Generation. As we describe the design of the system in this chapter, we touch briefly on Step 1 of Figure 1, Extraction and Formula Generation, as it provides the input for Step 2. Similarly we briefly describe Step 3 of Figure 1, Evaluation, because it illustrates a strategy for using the output of Step 2. However, the bulk of this chapter is devoted to describing our focus, Step 2.

3.1 - Overview of Extraction and Formula Generation

Step 1 of Figure 1 shows a two-part process[Tus04] for taking a web page describing a clinical trail, extracting the eligibility criteria, and transforming them into a set of first order predicate formulas. The first part, Criterion Extraction, takes a web page describing a clinical trial as input. For this thesis we used clinical trials from ClinicalTrials.gov, an internet site sponsored by the National Institutes of Health and the National Library of Medicine[CT]. We created a Python script that reads the web page describing a trial and extracts the eligibility criteria as well as available context information. The context information consists of items such as whether a criterion is an inclusion criterion or an exclusion criterion. The output of this part is an XML document containing the criteria and context information. Adapting the system to work with trials from other sources would involve modifying the Python script to understand the format of the new source.

The second part of Step 1, Formula Generation, takes the XML document with the extracted criteria as an input. This process parses each criterion using a link grammar parser[ST91]. From this it then creates a first order predicate calculus formula representing each criterion as Figure 3 illustrates. This process relies paritally on recognizable sentence or phrase structure. Since the authors of clinical trials sometimes use telegraphic or ungrammatical phrasing, and since the link grammar parser we are using in this work is not familiar with many medical terms and syntactic constructs, the system is not able to correctly parse some criteria into predicate formulas.

Figure 3 shows an example of the output of the Extraction and Predicate Generation step. The root element of this XML document is labeled “criteria”. This element contains a “trial” attribute whose value is the URL of the clinical trial. The “criteria” element contains a sequence of “criterion” elements. Each of these “criterion” elements contains a sequence of “text” elements followed by a “formula” element. The “text” elements contain text that the system extracts from the trial document. The last “text” element in a sequence contains the eligibility criterion of interest. The preceding “text” elements contain available context information.

The “formula” element contains the predicate calculus formula generated in the second part of Step 1. In this formula, ampersand symbols representing the ‘and’ of terms in the formula, are encoded as “&” as dictated by the rules for encoding XML. Most of the labels for the terms in the formulas are lower case forms of words or phrases from the original eligibility criterion. Where labels consist of more than one word, the words are joined by an underscore character. Some of the labels do not come directly from words in the original criterion, but rather are generated by the Formula Generation process to represent implied semantics. For example, when the structure of a criterion is

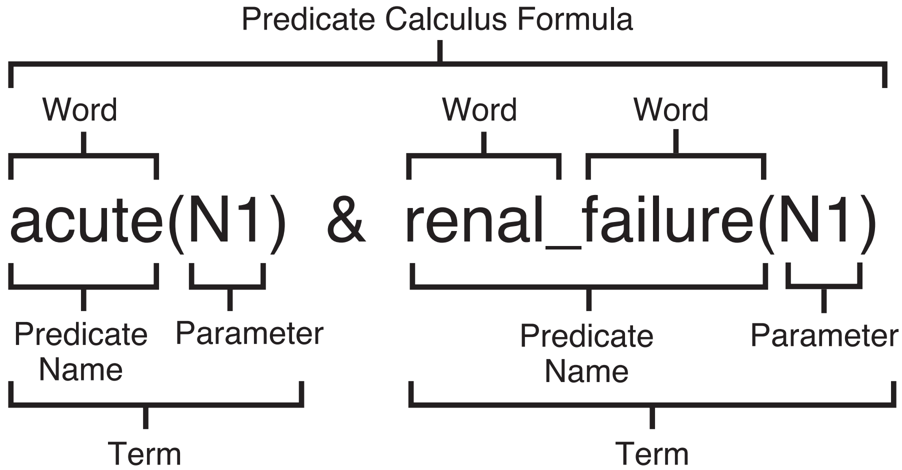

Figure 13 - Parts of a predicate calculus formula.

consistent with what the Formula Generation process expects for a numeric measurement, the system creates terms with labels such as “measurement”, “magnitude”, and “units” to make explicit the structure of the measurement.

The parameters of the predicate terms are most often variables. The naming convention for these variables is an upper case letter followed by a number. In addition, some of the parameters may be string or numeric values. If a parameter is a string it begins with a lower case letter. Numbers are represented without modification as parameters. Figure 13 diagrams the parts that make up a predicate formula.

Part of the processing done by the link grammar parser leverages part of speech information about the words being processed. We found this information to be useful in the concept mapping process described below. We therefore made the system able to include this information in the XML file output from Step 1. When this information is included, a string indicating the part of speech is prepended to the label of a term with a separating pound-sign symbol. For example, the criterion “active neoplasms” would be represented as ADJ#active(N1) & N#neoplasms(N1).

3.2 - Concept Mapping

The process outlined in Step 2 of Figure 1 takes the XML file described above as input. It attempts to map each criterion to concepts and data structures in the target electronic medical record. For each criterion that is successfully mapped we generate executable code for determining if a patient meets the criterion. We discuss the concept mapping portion of this process in this section and discuss the code generation portion in the next section.

As discussed in Chapter 2, IHC's CDR (Clinical Data Repository) stores clinical data as instances of detailed clinical models that can be viewed as a series of nested name-value pairs. Recall also that all of the pair names are coded concepts, as are some of the pair values. Since all of these coded concepts are in the HDD (Healthcare Data Dictionary), the mapping task consists largely of trying to match words and phrases from the eligibility criteria to concepts in the HDD that represent either names or values in detailed clinical models.

The HDD contains many concepts that are irrelevant for our purposes. To enhance performance and to make the system more portable, we created a database with the subset of HDD content consisting of concepts that are either names or values in the detailed clinical models that are stored in the CDR. The content of the HDD is stored in a normalized relational fashion and we kept the same relational structure in our working subset. This way our system could easily use the live HDD or our subset of the HDD by merely changing a configuration parameter. In addition, we created an abstraction of the HDD for our system with a TerminologyServer interface that defines a set of methods for making vocabulary related queries. We then created an implementation of this interface against the HDD.

The Concept Mapping portion of the system iterates through each criterion, and attempts to map it to coded concepts from the HDD used in the CDR's detailed clinical

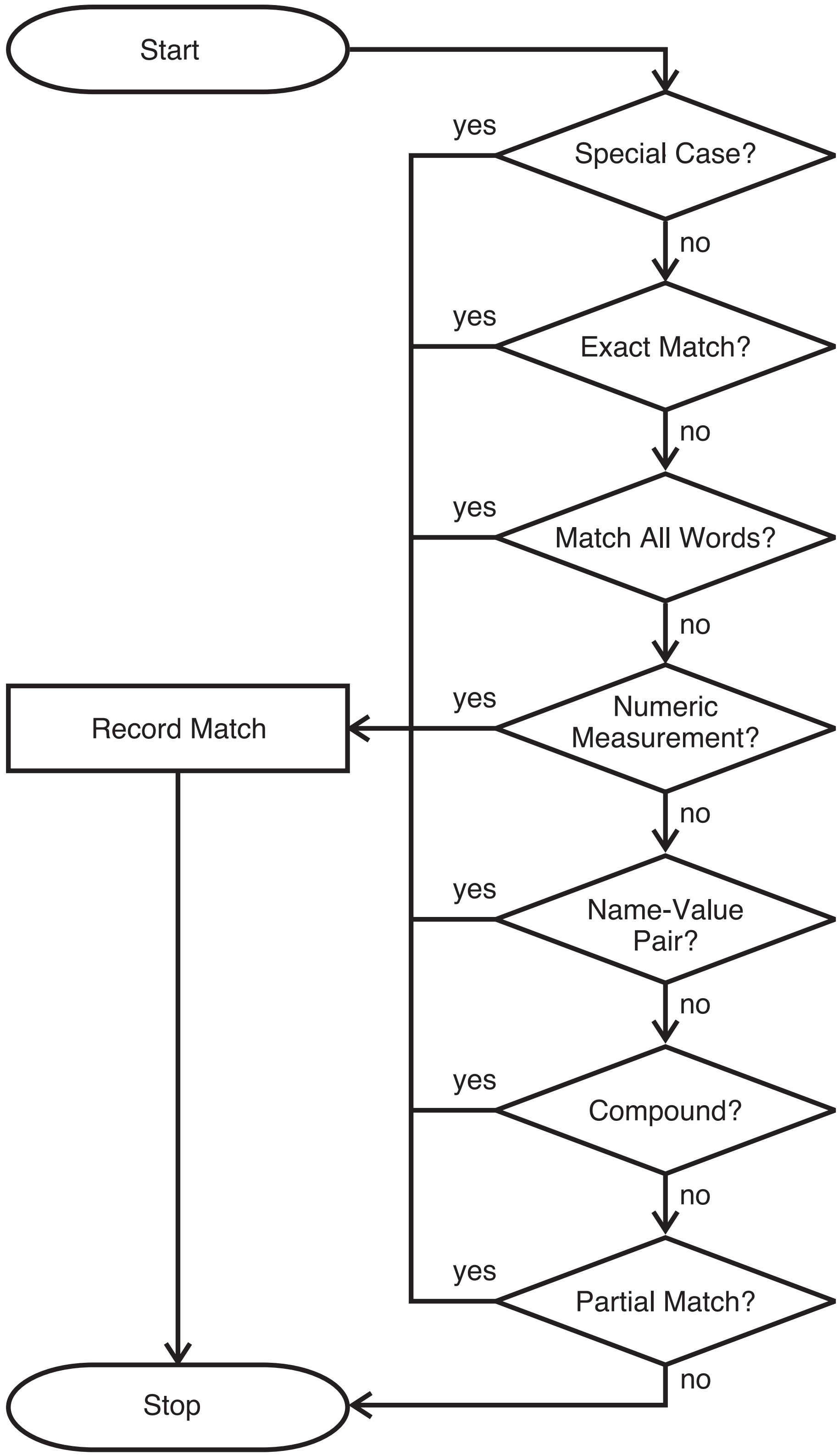

Figure 14 - Flow chart of matching process.

models. The system uses multiple matching strategies to generate these mappings. These strategies are executed sequentially, and once a match is found, subsequent matches are not sought. Figure 14 is a flow diagram outlining the matching process. Each of the seven decision points in the diagram represents a matching strategy, which we describe in detail below.

As with the TerminologyServer interface, the concept mapping application is made up of a generic interface that can use specific implementations. For this thesis we created an implementation of the Mapper interface that is specific to IHC's database and uses the IHC specific terminiology server. To apply this system to a different electronic medical record would require creating the appropriate implementations of these interfaces. The system chooses which implementation to use by reading a Java properties file on invocation.

The first step in the matching process is to look for and handle special cases. ClinicalTrials.gov represents information about the eligible ages and genders for a trial differently from other eligibility criteria. Age and gender information is in a consistent location and format across all trials. As a matter of efficiency, and to build a mechanism for special case handling into our system, we chose to handle age and gender as special cases. To do this we created specific methods to look for and interpret these criteria. These methods use string comparisons and regular expression matching to determine if a criterion is one of the special cases.

The second matching technique the system uses is to try to match the raw text of a criterion against the database. We retrieve the raw text from the last "text" element of the current criterion in the XML input to this step. We remove any leading or trailing whitespace, and retain word order and stopwords. Thus, for example, we try to directly map "LDL-C 130-190 mg/dL" from the first criterion in Figure 3, " Heart disease" from the second, and “No plans to move from the area over the next 9 months” from the third. The query for this match and the queries in all subsequent steps are case-insensitive.

Since these first two mapping strategies rely only on the text of the criterion and do not require a predicate calculus formula, they are executed for every criterion. The remaining steps are executed only for criteria that are successfully parsed into predicate calculus formulas.

The next matching method tries to match all of the labels of the terms in the predicate calculus formula describing the criterion. For example, the criterion “heart disease” may be described by the formula: heart(x) & disease(x). In this stage the mapper would look for coded concepts in the HDD that contain both the word “heart” and the word “disease”. From the resulting concepts, the system selects the one with the fewest extra characters as a match. In this phase, stopwords are removed and the order of the words is ignored. If we do not find any concepts containing all of the significant words, no match is found and the system moves on to the next matching strategy.

In the next matching strategy, the system tries to match the predicate as a numeric measurement. Numeric measurements have a name, a numeric value and units of measure. The predicate generation process handles numeric measurements that it is able to recognize in a consistent manner. For this match to succeed, the predicate must have all of the elements of a numeric measurement listed above. In addition, the name portion of the measurement must map to a coded concept in the target database. Without the name mapping to a known concept, the value portion is of little use. For example, the first predicate calculus formula of Figure 3 has a “measurement” term because the process in Step 1 recognized the criterion as a measurement. It has a magnitude, “130-190,” and units, “mg/dL.” Finally, this criterion is mapped as a measurement because the system can map “LDL-C” to a concept in the HDD.

If all of the previous methods fail to find a match, the system finally looks for partial matches and matches that involve more than one concept. The first step in this process is to find the best partial possible match. We do this by matching every word in the label of every term for the current criterion against the HDD individually. From all of the coded concepts that match, we select the one that corresponds to the most words in the criterion, and contains the fewest extra (i.e. non-matching) characters. In doing this we also take into account the part of speech of each word in the criterion. For example, consider the criterion "active neoplasms" from the trial in Figure 2. If the concept mapper cannot find one concept containing both of the words, "active" and "neoplasms", but it can match each word individually in separate concepts, then it would use the part of speech information to chose a concept corresponding with "neoplasms" in preference to a concept corresponding with "active" as a match. This helps the mapper choose terms that are more likely to be significant in evaluating eligibility.

This part of speech heuristic relies on the assumption that words of certain parts of speech are more discriminating or important than others when making a match. Currently, the system only considers two part of speech categories, nouns and everything else, and gives precedence to nouns. For example, when we have a phrase consisting of an adjective and a noun that do not map to a single concept in the target database, concepts containing the noun are generally better matches than concepts containing the adjective. Although this heuristic is generally useful, it is not always correct. For example, in the concept "renal disease," the adjective "renal" is probably more discriminating than the noun "disease."

If a partial match is found, we then attempt to make some sense out of the remainder of the predicate formula. This process has three possible outcomes. The first possibility is that we cannot match any other part of the criterion. In this case we simply map to the partial match. The second possibility is that we can determine that the remaining

```xml
<VMRQuery class="Observation">
	<value op="equals">
		<cd code="1450395" displayName="heart disease"/>
	</value>
</VMRQuery>
```

Figure 15 - A sample VMR query.

portion of the predicate is related to the partial match in a name-value pair relationship. For example, the words in the criterion “diagnosis of appendicitis” do not occur together in a single concept in the HDD, but both “diagnosis” and “appendicitis” map individually to concepts in the HDD. In addition, we can determine from the HDD that “diagnosis” is a valid name for clinical observation and “appendicitis” is a valid value for a clinical observation. Given this information we determine that the separately mapped pieces could represent a name-value pair in the CDR. In this case we map to the name-value pair.

The third possible outcome is that we can determine that the remaining portion of the predicate calculus formula is a conjunction or disjunction of more than one criterion. For example, the criterion, “Hyperthyroidism or hypothyroidism” from Figure 2 is a disjunction of two separate criteia, “hyperthyroidism” and “hypothyroidism.” In this case we map the individual pieces and relate them with a conjunction or disjunction.

If none of the strategies above create a mapping to the target database, we determine that we cannot map this criterion and move on to the next criterion. We keep track the the criteria that are not successfully parsed into predicate formulas, and those for which we receive a predicate formula but are unable to generate a mapping. This information can be passed along in a format such as that illustrated in Figure 5, or by passing references to objects in memory if Step 3 is executing as part of the same process. This

```
Criterion1 := READ {
        <VMRQuery class=”Observation”>
                <value op=”equals”>
                        <cd code=”1450395” displayName=”heart disease”/>
                </value>
        </VMRQuery>
};
```

Figure 16 - A sample Arden Syntax read statement containing a VMR query.

information can be useful to the user of the system when evaluating the final output of the system.

3.3 - Code Generation

The second part of Step 2 is Code Generation. In this part we take the output of the Concept Mapping process and use it to generate executable code. The code that we generate for this project is an Arden Syntax MLM (Medical Logic Module). As described in Chapter 2, we use VMR queries (Virtual Medical Record queries) within Arden's curly braces for data access.

Generating code for determining eligibility occurs in two steps. The first step takes place in tandem with the mapping process described above. When the system can establish a database mapping for a criterion, it generates a query against the VMR and associates it with the criterion. Figure 15 shows an example of a VMR query. In the opening "VMRQuery" tag we specify the VMR class that this query is against. As mentioned in Chapter 2, all VMR queries in this project are against the Observation class. In this example, the criterion is mapped to the coded concept "heart disease" in the target EMR (Electronic Medical Record). Using metadata from the target EMR, the system determines that "heart disease" is valid in the value part of a name-value pair. Thus, the

“VMRQuery” element contains a “value” element. If the mapped concept serves as the name part of a pair, then a “code” element replaces a “value” element in the query. The “op” attribute of the “value” element specifies a comparison operation for the value. The valid values of this attribute depend on the type of the element that is contained within the “value” element. In this case the “value” element contains a “cd” element representing a coded concept. The comparison operations that are valid for a coded concept include “equals” and “isa.” If the contents of the “value” element represented a numeric value, then numeric comparison operators such as “equals,” “less than,” and “greater than” would be applicable.

The second step in generating code to determine eligibility takes place after all of the criteria have been considered for mapping. In this step we generate the Arden Syntax MLM. The MLM we generate is focused on the executable logic. Even though the vast majority of slots in an MLM are required by the specification, only a handful are useful for machine execution. Most of the remaining slots are intended for human perusal. Therefore, for this project we populate only the small number of slots that are useful for automated processing. We do not generate any slots in the maintenance category. In the library category we populate the links slot with the URL of original clinical trial. In the knowledge category we populate the type, data, and logic slots. The only valid value for the type slot is “data-driven,” so we populate it appropriately.

To generate the data slot, we iterate through the eligibility criteria. For each criterion that does not have a mapping to the target electronic medical record, we generate a comment stating that this criterion could not be mapped, but we do not generate any executable code. For the criteria that do have mappings, we generate an Arden Syntax “read” statement. We assign the value of this statement to a variable as Figure 16 shows. The VMR queries that we generate are stated in such a way that a non-empty return value means the criterion was satisfied, and an empty return value means the criterion was not satisfied.

Finally we generate the logic slot. We first initialize an integer variable to zero and use it as a counter to keep track of how many criteria are met. We then iterate through each criterion. For each of the criteria that have mappings to the target database, we generate code that checks the value of each variable declared in the data slot and increments the value of the counter variable if the data variable has a value. After iterating through the criteria, we generate code that writes out the results. Figure 4 in Chapter 1 illustrates the generated code.

Although we have chosen to use Arden Syntax as the language of our executable code, we constructed the code generation subsystem using the same separation of interface and implementation that we used in other areas. Therefore generating code in a different language would only require the interested party to supply an appropriate implementation of the generator interface.

3.4 - Evaluation

The medical logic module that we generate could be used in a number of different ways. One possible strategy is to incorporate it in a process that searches through a large collection of patient records, looking for candidates for the trial. In this scenario the process could set a threshold for the percentage of criteria that need to be determined to suggest a patient for further consideration. An alternative approach would be to set the threshold on the number of patients to suggest instead of on the number of criteria met. This would present to the user a set number of patients that are most likely to be eligible.

Another use of the MLM would be to incorporate it in a process that works on patients who are scheduled for office visits. When the appointment is scheduled, or at some set time prior to the appointment, the scheduled patient could be evaluated against a number of trials in which the clinicians in that office are participating. Patients who meet a certain level of likelihood would be flagged for further evaluation during their visit.

In addition to the numeric results that the MLM delivers, the information it provides about each criterion could also be useful in pre-screening patients. For example, the clinician may know that a certain type of medical data is only rarely stored electronically. Therefore, if a criterion related to that type of data is not met by looking in the electronic medical record, the clinician may discount this item and base their judgment about whether to seek further evaluation of the patient on other criteria. Figure 6 gives an example of a report that provides this type of information.

4 - Experimental Results

In this chapter we describe the experiment that we performed. We discuss the results of the experiment and attempt to give some insight into what worked well and what improvements could be made.

4.1 - Experiment

To evaluate the system, we randomly chose one hundred clinical trials from www.clinicaltrials.gov and ran them through Steps 1 and 2 in Figure 1. For the trials that successfully completed these steps the system automatically generated a report including the following information:

- the number of criteria extracted;
- the number of criteria parsed into predicate calculus formulas;
- the number of criteria that were parsed but not successfully mapped to queries against the target system; and
- the number of queries generated.

In addition, the generated reports listed the text of the original criteria as well as the associated predicate calculus formulas and generated queries where applicable. We then manually inspected each report, looking at the generated queries, and categorizing them into four groups:

- queries that correctly and completely represented the original criterion;

Trials evaluated	100
Trials successfully completing Steps 1 & 2	85
Criteria extracted	1545
Criteria parsed into predicate calculus formulas	473
Criteria parsed but not mapped into queries	49
Queries generated	520
Completely correct queries	140
Other useful queries	113
Technically correct queries	4
Incorrect queries	263

Table 1 - Results

- queries that did not exactly represent the original criterion, but that would still return information useful in evaluating the criterion;
- queries that were not useful in evaluating the criterion, but were correct representations of the predicate calculus formula generated in Step 1; and
- queries that were incorrect and not useful in evaluating the criterion.

We tallied these numeric results and present them in Section 4.2. In addition, while inspecting each report we noted examples of things that worked well and items that illustrated opportunities for improvement. We discuss these items in detail in Section 4.3.

4.2 - Results

Table 1 lists the results of the experiment. Eighty-five of the one hundred trials selected successfully completed both Steps 1 and 2. The system identified 1,545 eligibility criteria to evaluate from these eighty-five trials. In Step 1, the system successfully generated one predicate calculus formula each for 473 of the criteria. In Step 2 we generated queries against the target electronic medical record (EMR) for all but 49 of the criteria with predicate calculus formulas. In addition, since some of the query generation strategies do not require a predicate calculus formula, we generated queries for 96 other criteria, for a total of 520 queries.

Upon inspection of the 520 generated queries, we determined that 140 of these completely and exactly represented their original eligibility criteria. Of these, 120 were the result of special case handling for age and gender. Another 113 of the queries, while not being either correct or complete enough to fully represent the meaning of the original criteria, were still close enough to yield information that would be useful for a clinician in evaluating the criteria. We also note four cases where the generated query correctly represented the associated predicate calculus formula but not the intent of the original criteria. In total, 257 queries were either completely correct, usefully correct, or technically correct. The remaining 263 queries were neither correct nor useful in determining eligibility.

4.3 - Discussion

In this section we will discuss the results of our experiment. In Section 4.3.1 we briefly discuss the performance of the Step 1, Extraction and Formula Generation. While these are outside the immediate scope of this thesis, the quality and quantity of input that they provide for our system is critical. In section 4.3.2 we discuss the concept mapping and code generation portion of the system, the focus of this thesis. We touch on things that worked well, areas where the system could improve, and aspects of the problem that are not easily remedied.

4.3.1 - Input Preparation

The process of extracting criteria from the HTML trial documents relied mostly on structural cues to distinguish criteria from surrounding contextual information. The process was good, but not perfect. It would sometimes identify a contextual statement such as “Exclusion criteria:” or “Patient has one of the following:” as criteria. Since the creators of the trial documents have considerable freedom in the way they can enter the criteria, and since they do not always use structural cues such as colons or indentation consistently to separate context from criteria, it is nearly impossible to extract the criteria without error. That said, a rough visual inspection of the extracted criteria and the original trial documents suggested an accuracy of about 90%. This is reasonable since, despite the freedom available for entering the criteria, most of the criteria were in rather simple lists and most of the visual cues that the authors used to provide context for people who would read the trial were structurally discernible.

The fifteen trials that did not complete both Steps 1 and 2 failed for two main reasons. The first reason was that the trials contained content that our system did not know how to interpret such as special HTML characters. For instance, the HTML code “ü” represents the umlat u character, and was not understood by the system. This is a consequence of having no control and almost no restriction on the possible input of the system. By modifying the system to act appropriately each time such a condition occurs, errors like this could likely be reduced to minimal frequencies.

The other reason for failure at this stage was related to the complexity of the criteria and available system resources. The complexity of certain criteria required more system resources to complete the final matching step than were available. Consequently they failed with an “out of memory” error. Possible solutions to this problem include running the system in an environment with more available memory or implementing code to identify significantly complex criteria and skip the last matching step on these criteria.

From the 85 processable trials, these initial steps prepared 1,545 criteria, with 473 associated predicate calculus formulas, as input for Step 2. The trials varied in size and complexity, having from 3 to 71 criteria per trial. They also varied widely in subject matter, covering conditions from cancer to infertility to gambling.

4.3.2 - Concept Mapping and Code Generation

The concept mapping and code generation processes resulted in the creation of structured queries against the target EMR. The strategies used in this process are outlined in Figure 14. As one might expect, the special case handling strategy worked very well. While it considered only age and gender, it accounted for the vast majority of the perfect queries, and nearly half of all queries that were at least useful. While age and gender were the only criteria that had a consistent representation across all trials, special case scenarios could be developed for other criteria as well. In particular, many trials dealing with cancer shared a common structure. This appears to be the result of most of these trials being submitted by the same institution, namely the National Cancer Institute. Special case handling could be developed to take advantage of this commonality as well as from common structure from other large submitters.

As previously described, most clinical data can be handled as a series of name-value pairs. A large number of the remaining successfully generated queries matched the name portion of some of the more well-structured name-value pairs. In particular, the system matched the names of many laboratory tests. One reason for this is that the value space for names is more limited and more constrained than the value space for values. Consider a lab test for hematocrit. As referenced in an eligibility criterion, the name of the test would likely be limited to the string “hematocrit” or a synonym such as the abbreviation, “HCT”. The possible values, however, range from all physiologically possible numeric values (e.g. 23 & 52.4) with their associated operators (e.g. equals, less than, not less than) to a variety of qualitative terms including “normal”, “abnormal”, “high”, “low”, “seriously low”, and “anemic”. In addition, we note that the name portion of the pair is usually more helpful. For a criterion such as “hematocrit greater than 39”, if an exact query is not generated, it would be much more useful to return all hematocrit measurements than it would be to return all observations with a value of 39.

To further illustrate this, in multiple instances, the system correctly mapped the name portion of a pair, but ended up with a query that was not useful because it incorrectly mapped the value portion. For example, consider the criterion “blood products or immunoglobulins within 6 months prior to entering the study”. The system found a mapping to a concept, “blood products used” which is used as a name in the target EMR. It also found a mapping to the concept, “months” which is present as a value in the EMR. However “months” is not a valid value for “blood products used”. If the mapping had simply stopped with “blood products used”, the resulting query would have brought back information useful in evaluating the criterion. However, as the query is currently formulated, it is guaranteed to never return anything. While simplifying the mapping process to stop after finding a name is one solution to the problem described above, the more elegant and useful solution would be for the system to determine which values are appropriate for a given criterion, and only allow queries that conform.

As alluded to above, the system also found frequent success in the use of synonyms. Due to the many synonyms in the data dictionary, the system was able to recognize concepts with many different representations and generate the appropriate queries. However, this success was limited somewhat by the use of ambiguous, often spontaneous or novel, abbreviations. While a human can often disambiguate such abbreviations by context, regulatory bodies have recently made a significant effort to ban their use. For example, in the trials that we considered, we mapped the abbreviation “PCP” to the drug “phencyclidine” while the trial intended “pneumocystic carinii pneumonia”, a disease that commonly afflicts patients with AIDS. In another example, we mapped PG to “phosphatidyl glycerol” while the trial used that abbreviation for “pathological gambling”.

From the 473 predicate calculus formulas, the system was not able to generate virtual medical record (VMR) queries for 49. Most of these 49 formulas were relatively simple in structure and did not contain any concepts in common with the target data dictionary. For example, an experimental medication, by its very nature, may be referenced in a clinical trial, but may be unlikely to appear in the data dictionary of a normal hospital EMR until it has been evaluated by several trials and has begun to gain wider, non-experimental use. More complex formulas were more likely to result in VMR queries because the last of the mapping steps creates a query if it can match any portion of the formula. Thus more terms in the formula results in more chances to match something.

One possibility for increasing the number of matches is to use additional sources of clinical concepts such as the National Library of Medicine's Unified Medical Language System[Lin90] or a database of experimental drugs. However, the increase in matches by doing this would not result in an increase in our ability to determine eligibility since the absence of a concept from the target data dictionary implies the associated EMR would not have such a concept stored in any of the patient records.

A significant number (113) of the generated queries could not directly determine if a patient met the criterion at hand, but provided some information that would be useful in making that determination. An example of this is the criterion "women who are pregnant or lactating" which mapped to a query for "pregnancy". While knowing whether or not a patient is pregnant may assist in evaluating this criterion, it is not enough alone to always make the appropriate determination. In another common scenario, the query is generated for a supertype or subtype of a concept in the criterion. For example, the system mapped the criterion, "uterine papillary serous carcinoma", to the concept "papillary carcinoma". Finding "papillary carcinoma" in a patient's record does not necessarily satisfy the criterion, but it would suggest to clinicians that they look more closely to determine what type of papillary carcinoma the patient has.

While the results of this thesis leave significant room for improvement, it is important to note that the maximum accuracy of this type of system is limited. The results of this system can be no better than the data stored in the target EMR. If certain concepts do not exist in the EMR, then it is impossible to query the EMR about criteria dependent on those concepts. Examples of these criteria include, “plans to become pregnant during the study”, “male partners of women who are pregnant”, and “no life-prolonging therapy available”.

In addition to concepts that simply are not in the EMR, many criteria could be evaluated based on data in the EMR, but only through inferencing with external knowledge. For example, “meets psychiatric diagnostic criteria for depression” requires the system to know what these “diagnostic criteria” are before this criterion can be evaluated.

Another limitation stems from the fact that many items put forth by the trial authors as criteria are actually informational statements or instructions with little or no discriminating value. An example of an informational statement is “Concurrent medications: Allowed: Dapsone”. While this may be interesting for a clinician to note, in reality whether the patient is taking dapsone has no bearing on their eligibility. An example of an instruction posing as a criterion is “women of pregnancy potential must practice contraception”.[†]

Other limitations in our ability to automatically evaluate every criterion include the difficulties in working with natural language such as double negatives and other logically incorrect, yet humanly understood constructs. Statements that imply information specified elsewhere in the trial document are also troublesome. For example, the criterion “duration of less than 10 years” does not explicitly state what it is that must have the specified duration. This must be inferred from the context of the trial.

In summary, the system performed well when dealing with special cases and when mapping to the name portion of name-value pairs. It is not reasonable to expect all the

[†] As an aside, the author notes that the large number of criteria dealing with pregnancy in the example listed here is not based on a skewed data sample or a preoccupation with that health condition, but rather is due to the fact that researchers are very concerned about the possibility of a therapy or procedure adversely affecting a fetus or nursing child. As a result, the researchers commonly stipulate very specific eligibility criteria related to pregnancy and nursing.

criteria to yield good queries without significant rigor on the part of the trial authors to eliminate ambiguity and logical errors. Even then, all information needed to determine eligibility is not readily available in most EMR's. That said, we have illustrated a number of places where we could improve the system and generate a higher number of better quality queries.

5 - Conclusion

5.1 - Conclusion

This thesis demonstrates that some degree of automatic evaluation of eligibility criteria is feasible. The initial steps of the process prepared about one-third of the eligibility criteria into predicate calculus formulas. Given this input, the mapping and code generation functionality of the system generated useful queries for about half of the number of criteria that had formulas.

Improvements in the upstream processes, criteria extraction and formula generation, would provide a larger amount of better quality input for the system to work with. However, improvement in rigor and precision of authoring clinical trial eligibility criteria may have an even greater impact. Moderating expectations, EMR implementers could reasonably develop a tuned version of this system that would not automatically determine eligibility, but rather present the clinician with a set of data that may be helpful in determining eligibility.

5.2 - Future Work

As described above, one of the more problematic areas in this process is getting from natural language statements that are adequate for clinicians to statements of the criteria that are computable. One approach to this problem is to specify the eligibility criteria in a more precise and computable format at the time they are authored. Another approach would be to build up a collection of medical knowledge in the form of ontologies and axioms that could be used to assist in bridging the gap. The benefits of the second approach include that the knowledge could then be used for problems beyond clinical trial eligibility and it does not place an additional burden on the authors of trials.

For example we could create one or more ontologies describing diseases and their relationships to laboratory values. Given this information, if we encountered a criterion of hypothyroidism, but could not find a coded concept for hypothyroidism in the EMR, looking in the ontology would tell us that certain laboratory values were sufficient for the diagnosis and we could then query the EMR for these laboratory values.

The system as presented was built on a general framework, but with a specific implementation for the target database. The implementation could be generalized to allow for broader application. For example, we could make use of the UMLS or other vocabularies in the mapping tasks. Doing this may increase our chances of mapping a predicate to a known concept, but that concept would still need to be mapped into the target database.

Other possibilities for improving the system include:

- Mapping criteria to more VMR classes than just the observation class. This would facilitate more accurate queries against information such as procedures, demographics, and medications.
- Improving the handling of parts of speech. Currently the code generation process only handles nouns in a special way. By recognizing and using other parts of speech the system could better validate good queries from nonsensical ones.

Bibliography

[ASN] "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", International Standard ISO/IEC 8824-1, ITU-T Recommendation X.680, 2002.

[CDR] 3M Health Information Systems, 2005, At http://www.3m.com/us/healthcare/his/products/records/care_innovation.html.

[CT] ClinicalTrials.gov - Information on Clinical Trials and Human Research Studies, At, http://www.clinicaltrials.gov.

[HCP+90] G. Hripcsak, P. D. Clayton, T. A. Pryor, P. Haug, O. B. Wigertz, J. v. d. Lei, "The Arden Syntax for Medical Logic Modules", R. A. Miller, ed., *Proceedings of the Fourteenth Annual Symposium on Computer Applications in Medical Care*, 1990, Nov 4-7, Washington D. C., IEEE Computer Society Press, 200-4.

[HRS+98] S. M. Huff, R. A. Rocha, H. R. Solbrig, M. W. Barnes, S. P. Schrank, M. Smith, "Linking a Medical Vocabulary to a Clinical Data Model Using Abstract Syntax Notation 1", *Methods of Information in Medicine*, 1998, Nov, 37(4-5):440-52.

[Lin90] C. Lindberg, "The Unified Medial Language System (UMLS) of the National Library of Medicine", *Journal of the American Medical Reccord Association*, 1990, May, 61(5):40-2.

[PRC+04] C. G. Parker, R. A. Rocha, J. R. Campbell, S. W. Tu, S. M. Huff, "Detailed Clinical Models for Sharable, Executable Guidelines", *Medinfo*, 2004, 11(Pt 1):145-8.

[RHHW95] R. A. Rocha, S. M. Huff, P. J. Haug, H. R. Warner, "Designing a Controlled Medical Vocabulary Server: the VOSER Project", *Computers and Biomedical Research*, 1994, Dec, 27(6):472-507.

[SC98] K. A. Spackman, K. E. Campbell, "Compositional Concept Representation Using SNOMED: Towards Further Convergence of Clinical Terminologies", *Proceedings of the American Medical Informatics Association Annual Symposium*, 1998, 740-4.

[ST91] D. Sleator, D. Temperley, "Parsing English with a Link Grammar", Technical Report CMU-CS-91-196, Carnegie Mellon Univeristy, 1991.

[Tus04] C. A. Tustison, "Logical Form Identification for Medical Clinical Trials", Master's Thesis, Brigham Young University, August 2004.

Appendix A

The trial in Figure 2 is from the ClinicalTrials.gov website at:

http://www.clinicaltrials.gov/ct/show/NCT00056511

The complete trial is shown on the following pages.

ClinicalTrials.gov
A service of the National Institutes of Health

Linking patients to medical research
Developed by the National Library of Medicine

Comparing Effects of 3 Sources of Garlic on Cholesterol Levels

This study is currently recruiting patients.

Sponsored by:	National Center for Complementary and Alternative Medicine (NCCAM)
Information provided by:	National Center for Complementary and Alternative Medicine (NCCAM)

Purpose

The purpose of this study is to determine whether fresh garlic can positively affect cholesterol in adults with moderately high cholesterol levels. This study will also determine whether the same effects can be found for two main types of garlic supplements: a dried powdered garlic (designed to yield the same effect as fresh garlic) and an aged garlic extract preparation.

Condition	Treatment or Intervention	Phase
Hypercholesterolemia	Drug: Fresh garlic or garlic supplements	Phase II Phase III

MedlinePlus related topics: Cholesterol

Study Type: Interventional
Study Design: Treatment, Randomized, Double-Blind, Placebo Control, Parallel Assignment, Efficacy Study

Official Title: Comparing Effects of 3 Sources of Garlic on Serum Lipids

Further Study Details:

Expected Total Enrollment: 220

Study start: May 2002; Study completion: April 2005

Garlic supplements are the most consumed herbal products in the United States. The most common health claim made for garlic supplements is cholesterol lowering activity. This claim has not been supported by recent clinical trials and meta-analyses. However, data suggest that it is not necessarily the garlic that has been ineffective, but rather the particular garlic preparations being used. To date, the predominant type of garlic preparation used in these clinical trials has been dried garlic powders. A few clinical trials have reported beneficial lipid effects using an aged garlic extract, and only a small number of inconclusive uncontrolled trials have used fresh garlic. A rigorous trial directly comparing different types of garlic preparations for their effects on serum lipids is needed.

Adults with moderately elevated low density lipoprotein cholesterol (LDL-C) will be randomized to one of four groups for 6 months: fresh garlic, dried powdered garlic tablets, aged garlic extract tablets, or placebo control. The fresh garlic will be provided to patients with "study sandwiches"; all other groups will receive the same study sandwiches without the garlic. All patients will take daily study tablets, but the tablet assignment will be double-blind. Patients will pick up study sandwiches twice a week and study tablets once every 2 weeks for 28 weeks. Blood samples will be taken once a month, with additional blood draws at the start and end of the study.

Eligibility

Ages Eligible for Study: 30 Years - 65 Years, Genders Eligible for Study: Both

Accepts Healthy Volunteers

Criteria

Inclusion Criteria:

- LDL-C 130-190 mg/dL (fasting single sample)
- BMI (body mass index) 19-30 kg/m2 (42-66 lb/m2)
- Weight stable for last 2 months
- Not actively on a weight loss plan
- Ethnicity representative of local population
- No plans to move from the area over the next 9 months

Exclusion Criteria:

- Pregnant, lactating, within 6 months postpartum, or planning to become pregnant in the next year
- Diabetes (type I or II) or history of gestational diabetes
- Heart disease
- Active neoplasms
- Renal or liver disease
- Hyperthyroidism or hypothyroidism
- Lipid lowering medications (known to affect lipid metabolism, platelet function, or antioxidant status)
- Blood pressure medications
- Excessive alcohol intake (self reported, more than 3 drinks/day)
- Currently under psychiatric care or severely clinically depressed

Location and Contact Information

California

Stanford Center for Research in Disease Prevention, Palo Alto, California, 94304-1583, United States; Recruiting

Christopher D. Gardner, Ph.D. 650-725-2751 cgardner@stanford.edu

Christopher D. Gardner, Ph.D., Principal Investigator

More Information

Study ID Numbers: 1 R01 AT01108-01

Record last reviewed: October 2003
Record first received: March 14, 2003
ClinicalTrials.gov Identifier: NCT00056511
Health Authority: United States: Federal Government
ClinicalTrials.gov processed this record on 2004-10-12