A Medical Terminology Server

Carole Goble, Peter Crowther and Danny Solomon

Medical Informatics Group, Department of Computer Science,
University of Manchester, Oxford Road, Manchester, M13 9PL, UK
tel: +44 61 275 6133; fax: +44 61 275 6932;
internet: <cag/pjc/wds>@cs.man.ac.uk

Abstract. GRAIL is a semantically constrained, generative compositional descriptive logic with subsumption and multiple inheritance designed to cope with the scale, complexity and variable granularity of medical concepts. It effectively represents a medical vocabulary for any application such as decision support systems, hospital record systems, clinical workstations and bibliographic systems. The EU-funded GALEN project aims to create a multilingual Terminology Server capable of being used as a resource by any medical application. The Terminology Server relates descriptive medical concepts represented in GRAIL to their natural language terms in a number of European languages and to corresponding (or best match) codes in conventional clinical coding schemes. In this paper we present GRAIL, the GALEN Terminology Server and make some remarks on the coupling of the classification-based clinical knowledge base (implemented in Smalltalk) with a relational database (Sybase).

1 Introduction

Scientific disciplines generally develop precise terminology or notation that is accepted and standardised. Medicine has surprisingly failed to develop a standardised vocabulary and nomenclature. Data retrieval and analysis is difficult with discrepancies between the meanings intended by the observers and recorders. What is an 'upper respiratory infection'? Does it include infections of the trachea? Is an 'ache' a pain? Such imprecision is problematic when we aggregrate data recorded by a number of professionals for trends and we must especially identify synonyms if we are not to get misrepresentations.

Free text is flexible, but is slow and unpopular with clinicians as well as being insufficiently structured and often ambiguous. The alternative is to use a coding or classification scheme to structure entries (the first classification of diseases was undertaken by Carl von Linne in 1749). Unfortunately, existing schemes were often developed for purposes with very different requirements, such as data aggregation for population statistics, rather than for the representation and manipulation of detailed clinical concepts which is required today.

Coding and classification schemes can either be enumerative or generative. Enumerative systems (e.g. ICD-9 [1]) rely on being able to define *pre-hoc* all that

can be said clinically; this becomes explosive if sufficient detail for everyday patient care is included, and unmanageably large and complex especially if cross-referencing or multiple axes are incorporated.

Multi-axial generative systems, such as SNOMED [3] can be acceptably expressive, but most insufficiently restrict what can be generated—permitting medically nonsensical constructs such as 'orange headaches' or 'broken eyelashes'. The claim that these would be rejected by doctors is irrelevant, for the need for manual intervention prevents the system validating or actively supporting data input by clinicians (or other clinical systems)[4], or the reliable exchange of information. The lack of constraints also limits consistency of structure; the semantics of the classification hierarchies are often heterogeneous and implicit, which severely limits the inferences that can be drawn.

A representation of the semantics of medicine and medical terminology and its active enforcement is essential for the integrity of the patient record. The medical terminologies required are moving away from the paper-based coding systems to computer-based representations of medical concepts with their semantic relationships. Such terminologies, especially if they are multi-lingual, not only support the Electronic Patient Record (EPR) but also faciliate information exchange and transfer.

Research into the exploitation of the knowledge inherent in medical terminology, coding systems and classifications for the EPR has progressed along seven main streams [5]; the three most relevant to us are:

1) developing controlled vocabularies of medical information systems;
2) finding methods for computer-supported automatic translation between different external representations; for example from medical text to one coding system, or in pairs between coding systems;
3) combining existing coding systems and controlled vocabularies within a common framework to support mapping between these systems for information retrieval.

1.1 A Descriptive Logic for Medicine

Rector and Nowlan [6] suggest that in order to deal with the scale, complexity and variable granularity of the clinical record requires a knowledge representation which is generative but is sure to only generate statements that are medically semantically correct. A descriptive model seems to be more promising than a traditional prescriptive one since the description of what was actually observed cannot be constrained to fit within a predefined view of what ought to have happened. The consequence of this is the ability to control what can be sensibly said about the observations of a patient—the schema of a patient's record. A patient can have a fractured bone, but not a fractured eyebrow; a fracture can only occur in one place

at a time; there is no such thing as an orange headache; drugs can be prescribed for patients, diseases cannot.

GRAIL (GALEN Representation and Integration Language) [12] is a descriptive logic with subsumption and multiple inheritance based on semantic networks. It belongs to the KL-ONE [8] family of knowledge representation schemes, and is generative by being recursively compositional, defining complex entities in terms of composite descriptions composed of a limited set of elementary concepts. Such descriptive entities are classified and placed in the correct place in the lattice. In addition, there is a semantic constraint mechanism which constrains the composition of the complex entities by the use of layers of sanctions. It also controls redundancy, tautologies and equivalent concepts (a process called canonisation). It is the sanctioning mechanism which separates GRAIL from such systems as CLASSIC [9] and BACK [10]. The sanctions ensure that only *semantically sensible* concepts are created, eg. a *Fracture which hasLocation Bone* and not *Fracture which hasLocation Tongue*. Without such sanctioning mechanisms it is easy to create nonsensical concepts. These particularization concepts are often implicit, unstored (un-reified) and must be generated during the classification process—they do not represent permanent data.

The medical concepts modelled are not confined to terminology but include assertions, such as *Cancer necessarily hasSeverity severe*. The 'severity' is not strictly terminology but clinical fact, making it impossible to record information about 'mild cancers'. GRAIL not only models the clinical concepts, eg. *Fracture which hasLocation Femur*, but also models the information model *Patient which has Condition*. The GRAIL classifier has a functional approach like Krypton, though the medical concepts are an enhanced form of the T-Box [2]. Transitive relationships across part-whole compositions are also controlled. GRAIL and its classifier is described in detail in [7,12].

2 A Medical Terminology Server

GRAIL effectively models the medical concepts required for any application such as decision support systems, hospital record systems, clinical workstations and bibliographic systems. It is a controlled vocabulary for medical information systems.

The GALEN [11] (Generalised Architecture for Languages Encyclopaedias and Nomenclatures in Medicine) project aims to create a multilingual Terminology Server (TeS) capable of being used as a resource by any medical application. The Terminology Server relates descriptive medical concepts represented in GRAIL to their natural language terms in a number of European languages and to corresponding (or best match) codes in conventional clinical coding schemes.

An example application for the TeS is that of supporting the accurate collection of data for the EPR. Doctors cite the time and effort required to enter data as the

single biggest barrier to using computers. PEN&PAD [6] is a prototype clinical workstation that uses a early version of GRAIL to represent the intensional concept and information mode. PEN&PAD uses predictive data entry as a particularly effective way of entering highly structured data quickly and easily. Given a complaint, the system generates a data entry form from the terminological knowledge base held in the TeS which contains all the likely options for modifiers and additional statements. The process of predictive data entry can be thought of as a clinician asserting or refuting belief in potential concepts generated by the knowledge base.

PEN&PAD not only used GRAIL to represent the intensional conceptual model but also the extensional asserted patient-related instances of the concepts, such as *Mrs Smith which <has (Fracture which hasLocation Femur) on 23/12/92>*. This unified model (described in detail in [13]) ensures that the data in the clinical record is accurate and semantically correct, as well as using the same classification mechanisms for data as for concepts.

2.1 Architecture

The TeS is a networked resource; it responds to requests from applications, and is intended as a service for any clinical application that needs to have an understanding of medical knowledge and medical terms. The TeS could be asked to ratify the description of a concept as being sound, or respond with known information about that concept.

Internally, the TeS is divided into three modules, illustrated in Figure 1:

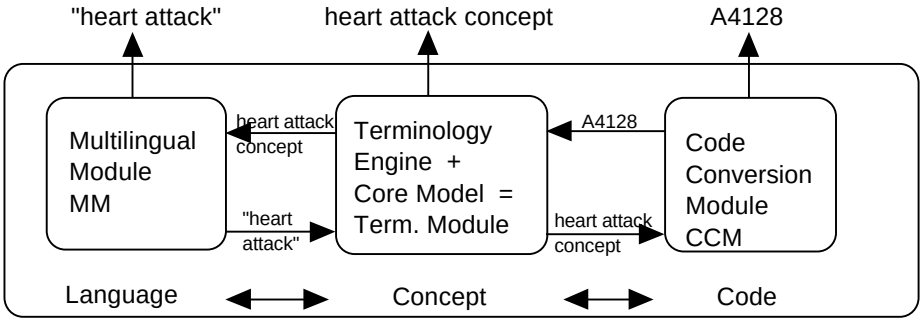

Fig 1: The GALEN Terminology Server

The **Terminology Module** (TM) deals with clinical concepts as abstract concepts. The Core Model is the representation of medicine (the Universe of Discourse data model). The Terminology Engine is the data model used to express the Core Model (the schema model). Together these form the *concept* model of medicine. The concept model is used as an 'interlingua' in translating between different external representations: natural languages or elements of existing coding schemes.

The **Multi-lingual Module** (MM) translates these clinical concepts to and from natural languages. It must also be able to take a name and map this to a clinical concept in the TM, and it maps concepts back to strings or names. Names are unique and are usually in the language of the modeller, and can therefore be thought of keys with cognitive semantics. The names might be used to infer strings—strings are the rendering in any language of the names.

The **Code Conversion Module** (CCM) takes a clinical concept and does a best fit mapping to a standard medical coding scheme such as SNOMED. Any number of coding schemes are supported. It can also take a code and map this to a clinical concept in the TM.

Thus the **Terminology Server** provides a framework for computer-supported automatic translation between pairs of coding systems and combines existing coding systems and controlled vocabularies within a common framework. It encapsulates the three modules; handles all the inter-module communication and all the external communication to users, applications and databases. In addition, it provides a model of reference by which external applications can refer to concepts.

An internal representation language, TesLa, is used as the integration medium for modules in the TeS. External communication with the TeS is via the API; client bindings are available for Smalltalk and C.

The TeS has been implemented in Objectworks/Smalltalk. The data requirements are substantial—the estimate for the number of reified medical concepts in the Core Model is around 1 million with up to 5 million edges. Hence the TeS is integrated with a database, or rather with several databases, implemented in Sybase.

2.2 Functionality

The TeS has two support roles:

1. A read-only role to support an application such as PEN&PAD. An application is likely to have predictable behaviour and hits, and a database could be clustered and optimised to suit (however this presumes that the TeS is only supporting one application area). Different applications will have different emphasis on the Core model, and different levels of specifity applicable. Eg. *Fracture which haslocation Humerus* could be the most specific description required for an epidemiology system but is not sufficient for a osteopathology application.

2. A read-write role to support a co-operative incremental distributed development environment for the Core Model of medicine. The Core Model is experimental and evolutionary with many versions requiring evaluation and possible reconciliation. In addition is is intended that users of applications will be able to extend a read-only common core with their own models, perhaps later seeing these incorporated in the common core. This role has unpredictable behaviour and hits.

Consequences of these two roles on the functionality of the TeS include:

- multiple Core models or sub-models, represented by multiple semantic networks or sub-networks;
- each sub-model must have authoring/versioning information;
- each user/application can be related to a specific Core Model/sub-model.

3 The TeS in detail

A more detailed representation of the TeS architecture is given in Figure 2. Note that different solutions may be appropriate for each module. Superficially the MM and CCM have a good deal in common and have a similar data and execution model, (though different databases). The support of the TE requires a different solution.

Entities are the categorial elementary or composite (particularised) entities and are only dealt with in the TM. *Rodents* are encapsulations of an entity in the Terminology Engine.and are the currency of concepts within the Terminology Server. Rodents could encapsulate a name or a description (a particularization) which can be thought of as a specification. Entities can be generated by lazy evaluation from their specifications (particularizations can be thought of as views for those readers familiar with databases). The fact that particularization concepts are often implicit, unstored (un-reified) and must be generated during the classification process causes some difficulties when the TM is interfaced with a database (see Section 3.3). The rodent also provides the *interface* to the TM for the other internal modules. The CCM and MM communicate with the TM via the rodents. The TM puts the wrapper on entities as they are sent out of it.

The TeS has a four level model of reference for concepts which external applications can use, and which take into account the fact that a TeS is accessible on network:

• **Handles.** An object id whose lifetime is restricted to a particular session (single connection with the TeS).

• **Local Name.** Linguistically relevant strings for concepts, valid across connections but restricted to one TeS running a particular version of the model; generated by applications.

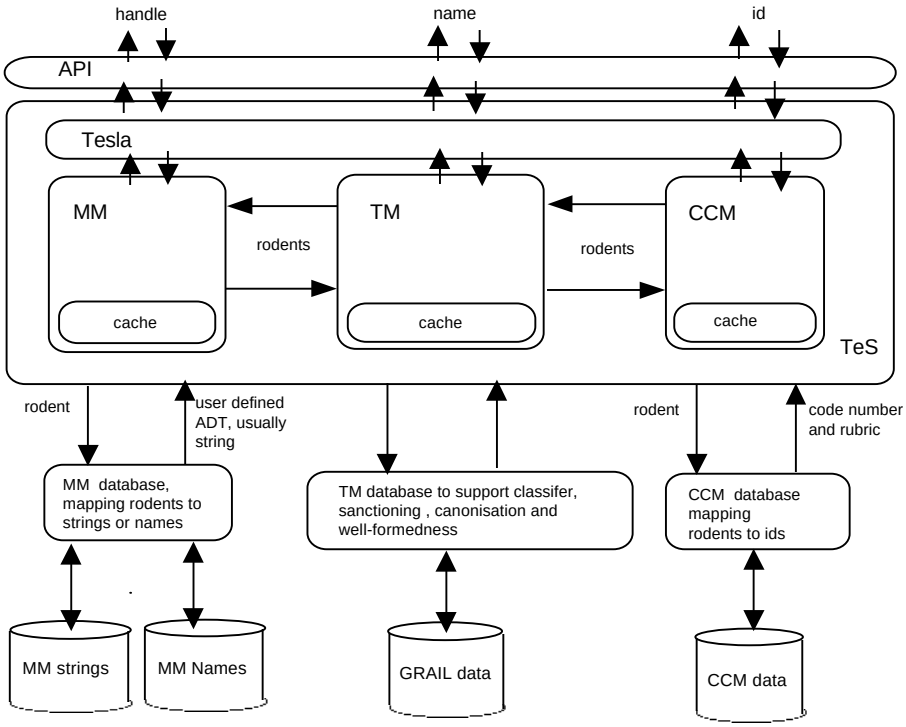

Fig 2. Architecture of the TeS

• **Local Id.** These have the same persistence and visiblity as local names, but are generated from within the TeS. They are an easily storable (linguistically meaningless) identifier for an entity—valid across time, but only to the originating TeS. These can be used to generate 'application-specific local coding schemes'.

• **Global Id.** A persistent object id. Valid across versions of the TeS—the global id is the only way of connecting across servers, and hence it is generated by the TM, but managed by the TeS.

The TeS maintains a set of mappings: between rodents and handles; between rodents and application-specific names and between rodents and application-specific local coding schemes which persist between connections with those applications. The database is needed to populate the data types in those implementations.

3.1 Multilingual Module (MM)

The MM is concerned with maintaining the relationship between the language-independent concept model and natural languages. The mapping between the concepts and their names is completely separate and independent of the Terminology Module. The MM passes a rodent to a database, which can returns any type, but usually returns a string. This doesn't necessarily mean that the

database stores rodents—they don't need to exist as objects outside the TeS. The MM has a cache for each session and we are considering whether it is better to have multiple caches, one for each language or whether a language should have a cache used by multiple sessions. Languages are completely pluggable with an unlimited number running concurrently in different sessions and within a session.

3.2 Code Conversion Module (CCM) and the Code Store

The Code Store is capable of storing any (known) coding scheme in a flexible but consistent way. In particular, it can handle different versions of the same coding scheme; subdivisions of coding schemes, such as SNOMED's modules and chapters; no, single and multiple inheritance within a coding scheme; different languages for rubrics; cross-references between coding schemes and boolean properties of codes. The CCM stores typed linked between these codes and concepts in the CORE model.

3.3 The Terminology Module

The problem here is the support of large models expressed in GRAIL. Various degrees of coupling have been proposed for the GRAIL databases and terminology. Three are considered below:

1. A loosely coupled approach where the database and Terminology Engine are independent and the entities and their relationships are stored atomically;
2. A layered approach with the database layer trying to anticipate what goes on in higher layers. The data concepts are stored in an optimised way where possible, but make it all appear atomic to higher levels of code.
3. A tightly coupled approach, with the database highly integrated with the terminology engine. This accommodates advanced schemes like pre-fetching and cacheing, but is the most complex to implement.

In an early version the loose coupling approach was used; however there were major problems with performance. The latest version adopts the tightly coupled approach. The majority of the classification, sanctioning, well-formedness etc activities of the classifier are undertaken by the classifier with little active support by the database—the database is really just an extension of the virtual memory. CLASSIC couples with an SQL database and attempts to do some of the classification work within the database [14]. The next version of the TeS moves to a more active role for the database. For example, if the question is 'does this entity exist' that might be done by the database without having to page in the network for the test. Although [14] has interfaced a descriptive logic representation with a relational database it supports classification only—not sanctioning or canonisation. In addition, the GRAIL classifier must support two kinds of query:

1. Given concept X, is it legal and what is its position it in the classification lattice?

2. Given concept X, how can I further specialise it?,

[14] deals with the first, but not the (expensive) second use, essential in the support of predictive data entry applications.

The high latency involved in any database request means that we need to minimise the number of requests made; we also need to maximise the parallelism between the TeS and the RDBMS. This can be done by requesting entities as far in advance as possible—one reason for tight coupling. The database can also help by performing closures on entities to find eg. parents, and passing those back as well. However, a balance needs to be found: in our experience the DBMS is slower at finding paths than the TeS, and the TeS frequently has parent entities cached anyway. The retrieval operations running inside the SQL server need to know what the TeS has in its caches, so that the minimum of extra querying is done at the server and the minimum is transferred across the TeS/DBMS link. There are two kernel page sets for every session that must be in memory permanently, assuming that the network can be semantically partitioned into pages : 1) application independent part of the network (e.g 'TopObject') and 2) application dependent part. The semantic partitioning is part of future work.

There is a spectrum of approaches to storing entities and relationships, two fundamental approaches originate from the *generative* nature of particularisations possible in GRAIL:

1. Store a definition of the entity or the relationship, and derive the interrelationships between entities when they are paged in.

2. Store the inter-relationships with the entity, and update them as concepts are added to the knowledge base Core Model.

The spectrum is produced by combining portions of these approaches—what information is stored versus what is derived each time. Definitions are faster to create, as less is stored, and may not need to be altered if additions to the network cause them to be reclassified. However, the approach relies on canonisation of all input entities and appropriate indexing (to ensure that the definition is retrieved if it exists) and is critically dependent on the speed of classification in memory.

Storing a network places less reliance on the classifier, as retrieval of an entity does not require it to be (re)classified. However, it is dependent on the speed of the database. The problem is presumed NP-hard—classification is probably exponential on some property of the network. As networks get larger, the time to classify an entity is likely to grow relatively quickly.

A major problem is the latency of the database (Sybase) coupled with the Smalltalk implementation of the classifier. It appears that (somewhat unsurprisingly):

1. storing a definition is better for applications with relatively small knowledge bases and high creation-to-retrieval rates.

2. storing a classified network is better for applications with large knowledge bases and low creation-to-retrieval rates.

An entity is retrieved even if it is only required for the classification of another entity, or for an investigation of another entity's properties; depending on the cache, this will tend to decrease the creation-to-retrieval rate. We have adopted the stored classified network approach.

Core model network names are integrated with the name of the database opened by the DBMS. In this way we can use the DBMS security mechanisms to prevent unauthorised changes to a model stored using that DBMS, as the user of the database is now the user of the network.

5 Summary

In this paper we have presented the GALEN Medical Terminology server. Version 1 of the server has been implemented and is being used within the GALEN project consortium to support the distributed building of a large medical concept model, and to support demonstrator clinical applications. The main difficulty is the co-ordination of the classification process of the GRAIL model with a database, especially as many of the GRAIL concepts are 'virtual' and merely specifications. Sophisticated caching and pre-fetching requires a tightly coupled system with semantic partitioning of the concept (GRAIL network) space integrated with the application the server is supporting. Eventually we envisage electronically-linked distributed communities of users extending a common concepts model held on Terminology Servers as part of the common infrastructure for developing coherent clinical systems which can interwork and be integrated.

Acknowledgements

The authors would like to acknowledge the other members of the Medical Informatics Group and GALEN consortium. This research is supported by the European Union under the Advanced Informatics in Medicine (AIM) GALEN project 2012.

References

1. World Health Organization. International Classification of Diseases: Ninth Revision. Geneva, 1977.

2. Brachman RJ, Fikes RE and Levesque HJ (1983) “KRYPTON: A functional approach to knowledge representation” in: IEEE Computer **16**(10) pp. 73-76

3. College of American Pathologists. Systematized Nomenclature Of Medicine. (Versions 1-3) Skokie, Illinois, USA: College of American Pathologists, 1977-1992.

4. Rector AL, Horan B, Fitter M, Kay S, Newton PD, Nowlan WA, Robinson D and Wilson A (1991) "User Centred Design Development of a General Practice Medical Workstation: The PEN&PAD Experience" in: Bauersfeld P, Bennett J, Lynch G (eds) Proceedings of Computer Human Interaction CHI '92, ACM, Monterey, Addison Wesley pp.447-453

5. Linnarson, R: Methods, design and components for a computer-based patient record to promote quality care in general practice. Linkoping University Medical Dissertations No 378, 1993

6. Rector AL, Nowlan WA and Kay S (1992) "Conceptual Knowledge: The Core of Medical Information Systems" in: Lun KC, Degoulet P, Pierre TE, Rienhoff (eds) MEDINFO 92, Proceedings of the Seventh World Congress on Medical Informatics, Geneva, North-Holland pp.1420-1426

7. Goble CA, Glowinski AJ , Jeffrey KG (1993). ''Semantic Constraints in a Medical Information System'' in: (Eds) Worboys M, Grundy F, Proceedings of BNCOD11, Lecture Notes in Computer Science 696 Advances in Databases, Springer-Verlag, pp. 40-57

8. Brachman RJ and Schmoize JG (1985) ''An overview of the KL-ONE knowledge representation system'' in: Cognitive Science **9**, pp. 171-216

9. Borgida A, Brachman RJ, McGuiness DL, Resnick LA, CLASSIC: A structural data model for objects, Proc 1989 ACM SIGMOD Conference on Management of Data, SIGMOD Record 18(2) Portland OR, June 1989,pp. 58-67

10. Nebel B, (1988): Computational complexity of terminological reasoning in BACK Artificial Intelligence, **34**(3) pp. 371-383

11. Rector A.L., Nowlan W.A., Glowinski A.J: Goals for Concept Representation in the GALEN project. in the 17th annual Symposium on Computer Applications in Medical Care, Washington, USA, SCAMC '93, 1993, .pp. 414-418

12. Goble CA, Glowinski AJ, Nowlan WA, Rector AL (1992) ''A Descriptive Semantic Formalism for Medicine'' in: Proceedings of the Ninth International Conference on Data Engineering, IEEE Computer Society Press, pp. 624-632

13. Goble C.A., Crowther P.J: Schemas for telling stories in medical records, in M.Jarke, J.Bubenko K.Jeffery (eds) Proceedings of the 4th International Conference on Extending Database Technology (EDBT94), Cambridge, UK

March 1994, Lecture Notes in Computer Science 779 Springer Verleg 1994 pp 393- 406

14. Borgida A, Brachman RJ, Loading Data into Description Reasoners, Proc 1993 ACM SIGMOD Conference on Management of Data, 1993, pp. 217-226