This article presents a framework and case study of knowledge revision and maintenance. Doing so it relies on a real world application of medical decision making, namely acute abdominal pain in children, or AAPC for short, and makes use of an integrated learning and knowledge revision system, namely MOBAL. Presented framework integrates expert knowledge with empirical knowledge. MOBAL serves as a vehicle of interaction between the expert user and medical records, which carry in to the process the empirical knowledge. An iterative-cyclic framework is used to formalize knowledge revision and maintenance. Results are demonstrated using AAPC. The article focuses on expert user -- revision system interaction while it offers a complete case study account of using MOBAL in a real world setting.

Work reported in this article was partially supported by the Health Telematics project GAMES-II (AIM, A2034) and by a research grant from the Hellenic Ministry of Development, Basic Research Program (PENED 1248). We would like to thank the MOBAL team, at GMD, Germany, that made available the MOBAL system. We thank our colleagues, Michalis Blazadonakis, Lena Gaga, Dimitra Athenaki, and the clinical staff of the Pediatric Surgery Clinic (University Hospital of Crete) for their assistance in data and knowledge modeling. Needless to mention that the responsibility for the work reported herein is entirely ours and that opinions do not necessarily correspond to those of the GAMES-II Consortium, European Commission or Hellenic Ministry of Development.
This article presents a methodology that supports medical knowledge acquisition and maintenance. The methodology integrates expert with empirical knowledge. The expert component presents up-to-date formal medical knowledge encoded in textbooks, either by means of theory or related case studies. In the current study this knowledge comes as expert decision making rules which represent indicative treatment scenarios about Acute Abdominal Pain in Children (AAPC). Empirical knowledge presents past decisions, captures implicitly context and it is encoded in past patient records, stored in a clinical information system.

The proposed methodology proceeds in two phases. In the first phase expert provided knowledge is applied on patient records. Conflicts between expert opinion and real world practice are discovered. In the second phase we use patient records to revise and to refine the initially provided expert knowledge in such a way that the user himself is satisfied in terms of accuracy and comprehensibility of the revised knowledge. Then, revised knowledge is validated using unseen patient cases. Validation is based on accuracy and qualitative metrics measuring distance between initially provided and revised knowledge. If accuracy is less than the desired threshold then knowledge is revised again using additional patient records or expert input. Thus the methodology follows a cyclic and iterative mode giving the expert always a “second chance” to improve the domain model.

The case study presented in this article composes a framework for medical knowledge revision and refinement were the two main actors are: the knowledge engineer - KE and the domain modeling system. KE introduces initial knowledge (in the form of expert devised rules) and a set of treated past medical records, part of which is mismatched by the input knowledge. The task is to test alternative revisions of the original knowledge in order to achieve a desired level of correct matching. KE interacts with the system, guiding the revision processes to produce results that, (1) achieve desired level of correct matching and (2) keep as much of the comprehensibility of original knowledge. In other words, the domain modeling system MOBAL, acts as a benchmarking system, offering a framework for testing different domain modeling alternatives.

To support this methodology we rely on the integration of classical knowledge elicitation approaches, used to capture expert knowledge in the first place, and model based machine learning techniques, used to learn and to revise initially captured expert knowledge. We compose the proposed methodology building around the MOBAL model based learning system (Morik, 1993; Morik et al., 1994; MOBAL, 1994). Our main emphasis is to show the balanced synergy between a domain modeling system, like MOBAL, and the KE in the task of maintaining compiled decision knowledge. We demonstrate our approach by applying it on a real world application related to the management of children with Acute Abdominal Pain. The article differs from usual accounts of classification accuracy assessment, i.e., Ohmann et al. (1996). It focuses on expert user – system interaction focusing on the complex tasks of knowledge acquisition, revision and maintenance.
In the following section we discuss the main aspects of medical knowledge acquisition as related to the need of coupling empirical and domain expert background knowledge. The next section presents the basic philosophy and procedures underlying balanced cooperative domain modeling with the main focus on the interactive nature of the approach. The section that follows presents the relative case study on the AAPC medical exemplar. We devote a special section to discuss about the necessity and utility of balanced cooperative modeling in medicine. In the final section we conclude on our work and related results.

MEDICAL KNOWLEDGE ACQUISITION

Medical knowledge acquisition is a complex process. Information comes in from different sources and it is subject to time and access constraints. For instance a rule which combines historical and clinical findings with respective laboratory results should consider that the latter are, generally, available with some delay. Medical practice is also prone to context practice bias. Context plays a very important role and previous research has demonstrated that it may lead the physician to deviate from standard practice (Morik et al., 1994; Vassilakis and Moustakis, 1996).

These observations suggest that active interaction between an expert user and knowledge acquisition system is necessary to achieve maximum benefit. Respective knowledge acquisition and modeling processes should be conducted in a way that the final elaborated knowledge based system should be an amalgam of the suggested expert medical theoretical knowledge and the related past medical experience as offered by relative past patients' records. A schematic view of expert decision making as realized by this methodology is shown in Figure 1. In other words, our goal is to elaborate an appropriate knowledge acquisition methodology and define respective procedures in which user interaction remains vivid and balanced between experience, context and theoretical background.

--- Please insert Figure 1 around here ---

Active interaction should prove valuable for medical, and other as well, domains were a key factor is the evolving nature of the domain knowledge. Towards this end we propose a methodology for medical knowledge acquisition and refinement based on model based machine learning techniques relying on the capabilities of the MOBAL machine learning system (Morik, 1993; Morik et al., 1994; MOBAL, 1994). The main advantage of MOBAL relies on its ability to offer a cooperative, continuous and dynamic construction of domain models. The physician, via the knowledge engineer, interacts with the system during the whole knowledge base construction process guiding knowledge acquisition, refinement and maintenance. However, a complete presentation of MOBAL is well beyond the scope of this article; the interested reader may refer to Morik (1993), Morik et al. (1994), or, to the system itself, which is available to the World Wide Web (MOBAL, 1994).
KNOWLEDGE REFINEMENT AND BALANCED COOPERATIVE MODELING

A simple way is to view knowledge acquisition and revision as a two-phase process. During the first phase, knowledge engineers acquire from domain experts an initial knowledge represented in a machine executable form. During the second phase, acquired knowledge is applied on real world domain tasks and cases in order to access its reliability and effectiveness. Potential contradictions are located and special revision processes are triggered in order to come up with a more refined knowledge that matches cases on an accepted level of classification accuracy.

In a more procedural setting the knowledge revision and refinement process is stated as follows:

- **Given**: An imperfect, partially correct, domain theory for a set of medical decisions and a set of pre-classified patient records each described by a set of observable features. Patient records are supposed to be conflict-free, that is, there exist no two identical records assigned to different classes. Domain theory is supposed to be self-consistent that is, it produces no conflicting class assignments to records.
- **Find**: An approximately minimal syntactic revision of the domain theory that correctly classifies all patient records.

The need for knowledge refinement process is clearly stated drawing from the following observations (Morik, 1993, Morik, et al, 1993):

- A domain model is always incomplete: the domain changes and our knowledge about the domain changes also. So, modeling is an infinite process.
- A domain model is more or less adequate for the uses of knowledge and the tasks it performs. So, modeling is an approximate process, which is expected to provide increased adequacy and reliability.
- A domain model is to be consistent in itself and reflect the present state of the art; this cannot be achieved at once. So, modeling is a deficient process.

However, the process of knowledge refinement and revision should be targeted to changing initially provided expert knowledge “as little as necessary”. Using MOBAL we exploit the syntactic minimality features embedded in the system to achieve minimal knowledge revision.

In Figure 2 we show a general outline of the continuous domain modeling process with Theory Revision Engine as its kernel process. Theory Revision Engine may be conceptually presented along the following operations:
Operation-1: We apply expert provided patient treatment rules $R$ on a set of past patient records $C$. If classification accuracy is greater than a desired (user specified) threshold then stop, otherwise goto step 2. Application of rules is based on normal *deductive* mechanisms.

Operation-2: The subset of rules $R_n$, that produce contradictive decisions and the respective missclassified set of patient records $C_n$, are identified. For each rule $R_i$ in $R_n$ the respective set of correctly, $C_{i;p}$, and incorrectly classified cases, $C_{i;n}$, are recorded. Only $R_i$ rules are subject to changes, all other rules will remain untouched. Furthermore, revision of rules will focus only to the cases that the rule covers, namely, $C_{i;p}$ and $C_{i;n}$. *Focused revision* operations offers the ground towards minimal knowledge changes.

Operation-3: Correctly classified records, $C_{i;p}$, are explicitly assigned to a dummy class $P$, and the respective $C_{i;n}$ records to another class $N$. Calling an *inductive operation* over $P$ and $N$ records we identify features (actually conjunctions) which distinguish between the two dummy classes. Adding these features as additional conditions to the rule under revision we conclude into a revised version of the rule which, if triggered, will leave uncovered $C_{i;n}$ records, but will still keep covering (correctly) the $C_{i;p}$ ones. Inductive operations offers the vehicle towards minimal revision. After performing Operation-3 we may potentially end up with a set of records being unclassified; take for example the border situation where there are no overlapping rules and that the missclassified records were covered just by the original rule being revised. Then we have to elaborate and introduce new rules that covers correctly these records.

Operation-4: The unclassified records may be composed from records assigned to different classes or from records that belongs just to one class. In the first case, we call an inductive operation to distinguish between classes. In the second case we are confronted with the problem of generalizing from a set of positive cases. The only alternative for performing such generalization is to introduce *general rule schemata* and instantiate them accordingly in order to cover consistently all unclassified records. This operation differs from classical learning from examples procedures which base their operations on heuristics for discriminating between categories. MOBAL, offers the alternative of learning with cases assigned just to one class. The set of rules introduced by Operation-4 are joint with the set of revised and originally correct rules in order to form the final revised target knowledge base. It is proved that the final revised knowledge will show a perfect accuracy over all initially given records (provided that a conflict-free data set is given).

Operation-5: Final revised knowledge is deductively applied on a set of unseen cases (not taking part in the revision process), in order to access the degree to which the performed revisions continue to remain valid in the course of continuous practicing within the evolving domain.
MOBAL

The MOBAL model based learning system offers an excellent working framework for an interactive balanced cooperative modeling process. It integrates a number of different tools for representing, acquiring and revising knowledge bases.

Representation

The basic representation formalism of MOBAL is first-order logic. All information items are stored into MOBAL’s internal data base as a set of facts. For example,

\[
pain\_history(CaseIndex, PainDuration, PainStarted)
\]

is a first-order predicate representing the history of pain for particular AAPC patient cases. It receives three arguments, the patient index number, the duration of pain (“today” or “1_to_3_days”) and a qualification of how pain actually started (“gradually” or “suddenly”).

Actually, the original set of records were accessed and stored, in the respective clinical information system, in an attribute-value format. That is, “pain_duration” and “pain_started” were two separate attributes with values, “today”, “1_to_3_days”, and “gradually”, “suddenly”, respectively. However, domain physicians always consider these two attributes as a whole and want to know both of them. Given the original attribute-value domain representation we used MOBAL’s internal deductive operations in order to elaborate and introduce a relational model of the domain. For example, introducing the following rule (see below for more details about rules in MOBAL),

\[
pain\_duration(CaseIndex,PainDuration) & pain\_started(CaseIndex,PainStarted) \rightarrow pain\_history(CaseIndex,PainDuration,PainStarted)
\]

we were able to deduce facts for “pain_history”. From a total of 72 attributes we concluded into a relational AAPC model which includes 31 predicates and captures domain specifics in a natural way. This reduction and economic domain representation has a strong influence to the final revised knowledge in terms of comprehensibility. All domain facts are inspectable and editable from the special MOBAL’s “Facts” window (see Figure 3).

Within the same formalism MOBAL represent rules; the construct:

\[
abdominal\_examination(CaseIndex,abdominal\_rigidity) \rightarrow actual\_decision(CaseIndex,operate)
\]

represent a rule in MOBAL format (following Prolog standards, arguments starting with capital letters represent variables and arguments starting with lower case letters represent constants). Introducing such rules in MOBAL and using its deductive Inference Engine we are able to deduce facts. For example, activating the above rule and provided that the fact “abdominal_examination(e110,
abdominal_rigidity)” is already stored in the knowledge base then the fact “actual_decision(e110,operate)” is inferred and stored. MOBAL keeps track of all the facts in the knowledge base accompanied with their derivation from other input or inferred facts.

Furthermore, MOBAL incorporates the powerful ability to represent rule schemata in a second-order logic formalism; the construct:

\[
\text{revision_mpred}(\text{pred,constant1,constant2}): \\
\text{abdominal_examination(CaseIndex,rigidity) \& pred(CaseIndex, constant1,constant2) \rightarrow} \\
\text{actual_decision(CaseIndex,operate)}
\]

declares a general rule schema, metapredicate in MOBAL’s terminology, that the rules to be learned are obliged to fit. For instance, the variable predicate “\text{pred(CaseIndex, constant1, constant2)}” is to be instantiated by existing and declared or newly introduced grounded predicates, which make the resulted rule consistent with the data base of declared and deduced facts. With the mechanism of metapredicates the user has the full control over learning and/or revision results; he/she can determine what kind of rules wishes to learn as well as their complexity. In this way, different decision may be tried letting the user to decide which, if any, fits to his/her needs and requirements.

MOBAL incorporates a variety of tools. Tools used in the present work include:

- **IE**: The *Inference Engine* able to perform deductive inferences.
- **RDT**: The *Rule Discovery Tool* able to perform inductive inferences.
- **CLT**: The *Concept Learning Tool* able to introduce new concepts; CLT uses RDT in order to define newly introduced concepts.
- **KRT**: The *Knowledge Revision Tool* able to perform complex theory revision operations. KRT integrates deductive and focused inductive processes that couple *Explanation Based Learning* and *Learning Form Examples* principles and procedures.

**CASE STUDY: INTERACTIVE MODELING OF AAPC TREATMENT SCENARIOS**

**The domain**

Acute abdominal pain in children (AAPC) encompasses a set of symptoms that cause severe pain, discomfort and increased tenderness in the abdomen of the child. AAPC originates from disorders either in the intra-abdominal or extra-abdominal areas (Waldschmidt, et al., 1990).

Management of patients is based on an explicit protocol by de Dombal (De Dombal, 1991) that captures pain specifics, related symptoms and results of laboratory tests. The attending physician needs to diagnose the cause of pain and then make one of the following mutually exclusive decisions, either “discharge” the child (in case the cause of the pain is not pathologic), or, to proceed to immediate “operation”, or, to “follow-up” the case for a period of six to eight hours.
at the end of which patient condition is re-assessed and the child is either discharged or admitted for operation. In case that an operation decision is followed the physician should already have in mind a spectrum of different potential causes which are to be confirmed or rejected and treated accordingly during the surgery operation.

In the current case study we rely on a set of 31 predicates to represent AAPC patient cases (we list predicates in Appendix A). These predicates cover demographic, clinical and laboratory tests' characteristics of patient cases. A total of 500 past AAPC patient cases was selected randomly from a data base (installed and running in the Pediatric Surgery Clinic, University Hospital at Heraklion, Crete, Greece). A subset of 406 most representative cases were carefully extracted from the respective clinical information system. From these cases, 166 were used for knowledge revision and the rest 240 were used as test cases, in order to test revision reliability. All selected cases were assigned either an “operate” or a “follow-up” treatment decision.

In the current case study AAPC medical treatment knowledge comes in the form of 17 expert devised rules concluding into the two different treatment decisions namely, 11 rules for “follow-up”, and 6 rules for “operate” (the complete set of expert rules is presented in Appendix A).

**Knowledge Revision with MOBAL**

The whole AAPC expert rules’ revision process was directed towards three basic operations presented in full details below. Figure 3, shows a screen layout of the full MOBAL system as it is arranged for the AAPC domain modeling process.

The input and learned knowledge is inspected and edited from the respective “Facts”, “Predicates” and “Rules” windows. The “Input Scratchpad” window serves for entering various items into the knowledge base -- facts, rules and metapredicates. The “Parameters” window offers the ability to tune and adjust accordingly, the different parameters used by the various tools during the learning and revision processes. For example, the “confirmation criterion” offers the user the ability to determine which rules are valid and can be entered into the knowledge base. Validity is expressed in terms of the cases that the rule covers correctly or missclassifies (see below for a more detailed explanation). In the “Agenda” window all the conflicting facts are posted and shown.

--- Please insert Figure 3 around here ---

**Operation-1**: The 17 expert rules are applied on the whole data set of 166 past patient cases. This is achieved by calling the inference engine tool of MOBAL. All derived facts, including facts for treatment class decisions, are recorded into MOBAL’s internal knowledge base and posted in the “Facts” window. The corresponding knowledge revision tasks are recorded into MOBAL’s agenda automatically for further consideration. For example, assume that case e111, originally assigned to class “follow-up”, is classified as an “operate” case. This
cause a new fact to be implied namely, “not(actual_decision(e111, follow-up))”, which coupled with the input fact, “actual_decision(e111, follow-up)”, forces the following knowledge revision task to be added in the agenda:

Knowledge revision on actual_decision(e111, follow-up) -- [1000,1000]

The [1000,1000] interval reflects the certainty by which the specific fact is implied; the left and right most numbers, represents the positive and negative evidence of the fact, correspondingly, with 1000 being the upper value that MOBAL accepts. In these cases, the specific fact has a complete positive and negative evidence that presents a conflict to be resolved.

At the same time MOBAL records all possible ways that the specific fact is derived and composes the corresponding derivation graph of it. This graph is subject to inspection by the user who may identify which facts and/or rules in the derivation needs to be considered for revision (see screen layout in Figure 4 where a very simple derivation of a fact is shown).

---- Please insert Figure 4 around here ----

Operation-2: The KRT is activated over all or part of the posted knowledge revision tasks. It explores all derivation graphs in order to form the possible minimal revisions that are necessary and sufficient for removing the offending inferences.

To perform minimal revisions KRT first computes the application set of the derivations. The application set of a derivation consists of all the facts in the knowledge base that take part in the derivation (graph nodes). Considering these sets it computes the minimal exclusion set of the rule which is composed from all variable instantiations for which the rule is not applicable. For example, the conflicting fact, “actual_decision(e111, operate)”, is derived by the application of the rule: “abdominal_examination(X, rigidity) \(\rightarrow\) actual_decision(X, operate)”, whose application set (in MOBAL’s representation) is: “(all X abdominal_examination)”. With the variable instantiation “\(\{X/e111\}\)”, the conflicting fact is implied. So, the possible minimal revisions to be considered concern case “e111” that composes the exclusion set of the rule under revision. Indeed KRT replaces the rule by,

\[\text{abdominal_examination}(X,\text{rigidity}) \rightarrow \text{actual_decision}(X,\text{operate}) - \text{all}\{(e111,...)\}\]

which is fired in all applicable cases except for cases in the set \{e111,...\}.

Operation-3: The next task is the refinement of revised rules. This task is accomplished by calling the Rule Reformulation operators, part of the KRT module. These operators aim to specialize rules by adding appropriate extra conditions in order to eliminate their exclusion sets.

In the current study we tried a two phase reformulation process. In the first phase we forced MOBAL to add just one extra condition to rules. This operation concludes into the elimination of most of the contradicting facts but still some
are present. So, we proceed a step further, forcing MOBAL to reformulate rules by adding two extra conditions. The final outcome is a set of reformulated rules with no contradicting inferences.

**Focused Induction and Concept Formation**

The exact procedure that MOBAL follows in order to reformulate rules is based on a *focused induction* process, being part of the CLT tool. That is, MOBAL assign a dummy class (by default it is called “concept X”, but user may assign a more meaningful name) to the cases not in the exclusion set of the rule (i.e., cases that the rule classifies correctly; if such cases does not exist then all cases assigned to the class of the rule and correctly classified are considered). Then it calls its **Rule Discovery Tool - RDT**, which acts as a learning from examples system, in order to *discriminate* these cases from the cases in the exclusion set of the rule.

The final outcome is a set of rules about “concept X”. Adding the extra condition, “conceptX(Caseindex)” to the rule, we disallow its applicability on the cases from its exclusion set. Actually, we replace each “concept X” occurrence by its respective induced definitions in order to preserve as much as possible the original comprehensibility power of knowledge. This results into a final revised set of rules that refer just to the original input predicates (conceivable by domain practitioners) and not to intermediate newly introduced concepts that serve just revision operational demands and may not receive a straightforward domain interpretation. Of course the user may assign a more natural meaning and rename accordingly each “concept X” entry and let the revised rules with the respective renamed “concept X( )” entries. In this cased study we followed the first alternative with the disadvantage of forming a large set of revised rules.

The above operations achieve minimal revisions based on two factors: (1) focus just on the conflicting cases and (2) use standard inductive processes which warranty *discrimination minimally* between cases.

The whole revision process is not just transparent to the user but it is also guided by him/her: the format and complexity of the final revised form of rules as well as their predictive accuracy level, are in the full specification responsibility of the user. For example, in the current study we formulated the general metapredicate, “revision_mpred”, mentioned above, which forces MOBAL to produced reformulation of rules that fits this metapredicate schema.

Furthermore, MOBAL offers the ability to the user, to specify the desired accuracy level of the revised rules. For example the entry (entered in the respective MOBAL’s “Parameters” window),

```
pos >= 3 & neg = 0
```

forces the valid revised rules to classify correctly at least 3 cases with no contradictions.
The theoretical background of the knowledge revision operations performed by MOBAL and especially by the KRT module may be found in (Wrobel 1994; Morik et al., 1993).

RESULTS

First we present a quantitative prospective of results regarding predictive classification accuracy. Then we proceed into a qualitative assessment regarding the deviation of revised knowledge from the original one.

Quantitative results

Our first concern is about the reliability of the performed knowledge refinement, expressed in terms of classification predictive accuracy results. In Table 1, a summary of prediction results are.

The original classification accuracy (original rules applied on the 166 patient cases) is 67%. The revised set of rules shows a 100% classification accuracy. This was expected because we guided MOBAL to produce revised rules with no contradicting results (the “neg = 0” specification entry). Of course, this was achieved following both reformulation phases; that is, the original rules were revised by adding one or two extra conditions.

---- Please insert Table 1 around here ----

On the set of unseen cases, 240, the original accuracy was 30%. The revised set of rules produces a classification accuracy of 69%. This shows a significant increase of about 43%, which consist an encouraging result for automated knowledge refinement approaches. Of course, on the remaining incorrectly classified cases the same revision operation should be performed in order to produce a perfect match. The crucial advantage of the knowledge revision and refinement operations is achieved because, only the revised knowledge (rule set) is to be considered for future classification and decision making operations and not the whole original set of cases; this achieves a great improvement in data storage and in potential future patients retrieval operations.

---- Please insert Table 2 around here ----

Qualitative results

A crucial aspect concerning, not the reliability but, the utility and efficiency of an automated knowledge refinement process is the deviation of revised knowledge from the original one. It is expected that original, expert provided, knowledge posses a high comprehensibility power with respect to the experts and the practitioners of the domain. So, we need to measure “how much knowledge is changed”.

11
We approach this estimation by computing various statistics on the original and revised set of rules. Table 2 summarizes these results with respect to the number of unchanged, deleted and specialized rules.

Final revised knowledge consists of 67 rules; note that about 47% of the original rules remain unchanged (8 from 17).

Many of the revised rules are just specialization of original ones, by just adding the induced definitions of respective “concept X” predicates. This results into overlapping rules - rules that cover the same set of cases. This is because RDT of MOBAL does not follow a hill-climbing procedure in its inductive operations. Different explanations and interpretations of the same set of cases may be produced. This consist a useful aspect regarding comprehensibility of results and diversity in decision making scenarios, at least for medical applications. In terms of efficiency in storage and retrieval this fact sourly consists a disadvantage. Other knowledge revision systems such as, NEITHER (Ourston and Mooney, 1994), produce more economic revision results; however, the system is limited to attribute value based representation and offer no interactive capabilities.

Table 3, presents another qualitative measure for the results regarding the number of predicates - still used, not used anymore and newly used - in the final revised knowledge. The result indicates a very small deviation from original knowledge.

The full list of revised rules is given in appendix B accompanied by the induced definitions of respective “concept X” predicates.

**OBSERVATIONS**

The presented case study and the underlying proposed methodology realize three basic observations:

- Expert decision making is the compilation of theoretical and practical knowledge. Theoretical knowledge refers to domain background items as recorded in textbooks or other reference material and in established problem solving procedures (i.e., differential diagnosis in medicine). Practical knowledge in contrast refers to past practicing experience within the domain (i.e., past patient cases and their assessment). What differentiates experts from novice practitioners is the way that such a compilation (or coupling) is performed. On one hand, which parts of textual knowledge are to be recalled - when confronted with a domain task - is subject to past practicing experience. On the other hand, the interpretation of a specific domain task and the evaluation of the problems it posts are tasks that are strongly linked and influenced by textual knowledge. Such a dynamic “feed-back” between theory and practice is common in medical reasoning and originates from the
same texture of medicine as a case based science. Evidence to that are a number of studies that demonstrate that experienced clinicians comes to some diagnostic ideas and hypothetical clues, of a greater or lesser specificity, after a limited number of questions have been posted to the patient and/or after limited clinical findings are confirmed (Cambell, 1987). It is the compilation of this dynamic feed-back operation in experts’ memory organization that makes early diagnostic clues to form (De Dombal, 1986). So, the need to couple pre-established background knowledge with past experience is posted. In a knowledge based systems setting, this puts the inevitable consequence to elaborate and device appropriate methodologies and procedures to operationalize this coupling.

- Medicine is an evolving science, new techniques and diagnostic instrumentations are presented or even new, more social (“patient's quality of life”) health care mentalities are introduced, acquired and made universally acceptable. If a medical knowledge based system is to be of any use and real world utility, it has to address the problem of maintaining its “core” entries and procedures. Most than often domain models are incomplete and with varying applicability (Morik, 1993, Morik, et al., 1993). For example, specific medical protocols and guidelines vary in different countries even if they are based on the same core background and established knowledge. These facts give rise to a special requirement: devise appropriate knowledge maintenance methodologies and adaptive procedures that will keep-track of the related evolution in theory and practice and keep the target knowledge based system self-consistent and tuned in place and time.

- Furthermore, real world domain models and related problem solving approaches are by nature “ill-defined” and most of the times inconsistent, at least for life sciences such as medicine; there are not “clear-cut” scenarios of how to address a patient case, evaluate its condition and treat it. For example, how really an automated knowledge based system is going to access a particular “patient’s truth state of reported symptoms” or even worst, “patient’s social status”, both of which influence the final treatment decision. We have to keep the balance between automated knowledge acquisition processes on one hand and user interference on the other.

The methodology for interactive knowledge base construction and maintenance presented in this article offers a working framework where all the aspects mentioned above are addressed.

CONCLUSIONS

In this article we presented a methodology for continuous domain modeling construction based on machine learning techniques. We used MOBAL, a model based learning system and especially its knowledge revision services. We demonstrated this methodology on a real world medical domain namely, Acute Abdominal Pain in Children.
The MOBAL system, offers an operational environment where human interaction is kept vivid and tuned with the automated knowledge acquisition and revision processes, offering a flexible and transparent working framework.

Furthermore, the results on the AAPC case study demonstrate the effectiveness and reliability of the approach with respect to consistency and extendibility of the knowledge maintenance process.

REFERENCES

MOBAL. 1994. Software system. GMD. Germany. (available via anonymous ftp from ftp.gmd.de, directory /pub/gmd/mlt, for non-commercial academic use).
Appendix A

Predicates for the AAPC medical domain

(*number* = number of arguments)

**PATIENT:** age_group/2, sex/2

**PAIN - RECTAL - BOWELS - MICRURITION:** pain_history/3, pain_status/3, pain_effect/3, site_of_pain/2, other_symptoms/2, cough_change/1, nausea_vomiting/1, bowel_exam/2, rectal/2, micrurition/2

**PREVIOUS PATHOLOGIC RECORD:** disease_record/2, medication/2

**GENERAL EXAMINATION:** mood/2, color/2, pulse/2, blood_pressure/3, temperature/2

**ABDOMINAL EXAMINATION:** rebound/2, tenderness/3, tenderness_shallow/1, tenderness_shallow_deep/1, tenderness_rlq_surface_shallow/1, tenderness_yes/1, abdominal_exam/2, bowel_sounds/2

**LAB TESTS:** blood_test/3, urine_test/3, urine_test_abnormal/1

**TREATMENT DECISION:** actual_decision/2

Original expert rules

**OPERATE**

**RULE_1:** age_group(Caseindex,greq_4) & pain_status(Caseindex,steady,A) & nausea_vomiting(Caseindex) & temperature(Caseindex,gr_37) & blood_test(Caseindex,w_cells,increased) & blood_test(Caseindex,neutrophiles,increased) & tenderness_rlq_surface_shallow(Caseindex) --> actual_decision(Caseindex,operate).

**RULE_2:** age_group(Caseindex,less_4) & pain_status(Caseindex,steady,A) & nausea_vomiting(Caseindex) & temperature(Caseindex,gr_37) & pain_history(Caseindex,today,B) & tenderness_rlq_surface_shallow(Caseindex) --> actual_decision(Caseindex,operate).

**RULE_3:** disease_record(Caseindex,abdominal) & nausea_vomiting(Caseindex) & temperature(Caseindex,gr_37) & abdominal_exam(Caseindex,rebound) --> actual_decision(Caseindex,operate).

**RULE_4:** pain_status(Caseindex,steady,A) & temperature(Caseindex,gr_37) & tenderness(Caseindex,rlq,shallow) & not(urine_test(Caseindex,erythrocytes,normal)) --> actual_decision(Caseindex,operate).

**RULE_5:** medication(Caseindex,antibiotics) & disease_record(Caseindex,abdominal) & temperature(Caseindex,gr_37) & tenderness_rlq_surface_shallow(Caseindex) --> actual_decision(Caseindex,operate).

**RULE_6:** abdominal_exam(Caseindex,rigidity) --> actual_decision(Caseindex,operate).

**FOLLOW-UP**

**RULE_7:** nausea_vomiting(Caseindex) & tenderness_shallow(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_8:** nausea_vomiting(Caseindex) & micrurition(Caseindex,dysuria) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_9:** nausea_vomiting(Caseindex) & urine_test_abnormal(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_10:** bowel_exam(Caseindex,constipation) --> actual_decision(Caseindex,follow-up).

**RULE_11:** tenderness_yes(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_12:** blood_test(Caseindex,w_cells,increased) & blood_test(Caseindex,neutrophiles,increased) & tenderness_yes(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_13:** micrurition(Caseindex,haematuria) & nausea_vomiting(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_14:** other_symptoms(Caseindex,vomiting) & bowel_exam(Caseindex,diarrhea) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_15:** bowel_sounds(Caseindex,increased) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex,follow-up).

**RULE_16:** other_symptoms(Caseindex,nausea) & other_symptoms(Caseindex,anorexia) & age_group(Caseindex,less_4) --> actual_decision(Caseindex,follow-up).

**RULE_17:** temperature(Caseindex,gr_37) & cough_change(Caseindex) --> actual_decision(Caseindex,follow-up).
Appendix B

Revised set of rules
(The 64 revised rules were produced after "concept X" is instantiated in the rules by its respective induced definitions; predicates added as specialization conditions are underlined)

Specialized with concept1 :: RULE_1: age_group(Caseindex, gregq_4) & pain_status(Caseindex, steady, A) & nausea_vomiting(Caseindex) & temperature(Caseindex, gr_37) & blood_test(Caseindex, w_cells, increased) & blood_test(Caseindex, neutrophiles, increased) & tenderness_rlq_surface_shallow(Caseindex) & concept1(Caseindex) --> actual_decision(Caseindex, operate). => 3 rules

Unchanged :: RULE_2: age_group(Caseindex, less_4) & pain_status(Caseindex, steady, A) & nausea_vomiting(Caseindex) & temperature(Caseindex, gr_37) & pain_history(Caseindex, today, B) & tenderness_rlq_surface_shallow(Caseindex) --> actual_decision(Caseindex, operate).

Unchanged :: RULE_3: disease_record(Caseindex, abdominal) & nausea_vomiting(Caseindex) & temperature(Caseindex, gr_37) & abdominal_examination(Caseindex, rebound) --> actual_decision(Caseindex, operate).

Unchanged :: RULE_4: pain_status(Caseindex, steady, A) & temperature(Caseindex, gr_37) & tenderness(Caseindex, rlq, shallow) & not(urine_test(Caseindex, erythrocytes, normal) --> actual_decision(Caseindex, operate).

Unchanged :: RULE_5: medication(Caseindex, antibiotics) & disease_record(Caseindex, abdominal) & temperature(Caseindex, gr_37) & tenderness_rlq_surface_shallow(Caseindex) --> actual_decision(Caseindex, operate).

Specialized with concept1 :: RULE_6: abdominal_examination(Caseindex, rigidity) & concept1(Caseindex) & tenderness_rlq_surface_shallow(Caseindex) --> actual_decision(Caseindex, operate). => 3 rules

Deleted :: RULE_7

Unchanged :: RULE_8: nausea_vomiting(Caseindex) & micrurition(Caseindex, dysuria) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex, follow-up).

Specialized :: RULE_9: nausea_vomiting(Caseindex) & urine_test_abnormal(Caseindex) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex, follow-up).

Specialized with concept2 :: RULE_10: bowels_examination(Caseindex, constipation) & concept2(Caseindex) --> actual_decision(Caseindex, follow-up). => 16 rules

Specialized with concept2 :: RULE_11: tenderness_yes(Caseindex) & concept2(Caseindex) --> actual_decision(Caseindex, follow-up). => 16 rules

Specialized with concept2 :: RULE_12: blood_test(Caseindex, w_cells, increased) & blood_test(Caseindex, neutrophiles, increased) & tenderness_yes(Caseindex) & concept2(Caseindex) --> actual_decision(Caseindex, follow-up). => 16 rules

Unchanged :: RULE_13: micrurition(Caseindex, haematuria) & nausea_vomiting(Caseindex) --> actual_decision(Caseindex, follow-up).

Unchanged :: RULE_14: other_symptoms(Caseindex, vomiting) & bowels_examination(Caseindex, diarrhea) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex, follow-up).

Unchanged :: RULE_15: bowels_sounds(Caseindex, increased) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex, follow-up).

Specialized :: RULE_16: other_symptoms(Caseindex, nausea) & other_symptoms(Caseindex, anorexia) & age_group(Caseindex, less_4) & tenderness_shallow_deep(Caseindex) --> actual_decision(Caseindex, follow-up).

Specialized with concept6 :: RULE_17: temperature(Caseindex, gr_37) & cough_change(Caseindex) & concept6(Caseindex) --> actual_decision(Caseindex, follow-up). => 3 rules

• INDUCED CONCEPTS: concept1 (3 rules) / 2 (16 rules) / 6 (16 rules)

blood_test(Caseindex, w_cells, decreased) --> concept1(Caseindex), pain_effect(Caseindex, eating, getting_worse) --> concept1(Caseindex), pain_status(Caseindex, colicky, not_specified) --> concept1(Caseindex), urine_test(Caseindex, leucocytes, rare) --> concept2(Caseindex), urine_test(Caseindex, erythrocytes, many) --> concept2(Caseindex), urine_test(Caseindex, bacteria, many) --> concept2(Caseindex), blood_test(Caseindex, w_cells, normal) --> concept2(Caseindex), blood_test(Caseindex, neutrophiles, normal) --> concept2(Caseindex), tenderness(Caseindex, ruq, shallow) --> concept2(Caseindex), tenderness(Caseindex, rlq, none) --> concept2(Caseindex), tenderess(Caseindex, rlo, deep) --> concept2(Caseindex), tenderness(Caseindex, luq, shallow) --> concept2(Caseindex), tenderness(Caseindex, liq, deep) --> concept2(Caseindex), tenderness(Caseindex, central, deep) --> concept2(Caseindex), pain_effect(Caseindex, eating, getting_worse) --> concept2(Caseindex), pain_status(Caseindex, intermittent, not_specified) --> concept2(Caseindex), pain_status(Caseindex, colicky, moderate) --> concept2(Caseindex), pain_history(Caseindex, days_one_to_three, gradually) --> concept2(Caseindex), tenderness(Caseindex, ruq, deep) --> concept6(Caseindex), tenderness(Caseindex, rlq, none) --> concept6(Caseindex), tenderness(Caseindex, central, deep) --> concept6(Caseindex).
Figure 1. Expert decision making as an amalgam of theoretical and practical knowledge.
Figure 2. Knowledge refinement in an iterative cyclic domain modeling process.
Figure 3. MOBAL operational screen layout for the AAPC domain modeling process.
Figure 4. Derivation graph of a contradicting fact.
### Table 1. Comparative predictive accuracy results between original and revised knowledge for given and test (unseen) data

<table>
<thead>
<tr>
<th>RULES</th>
<th>CLASS</th>
<th>ORIGINAL DATA (166 cases)</th>
<th>TEST DATA (240 cases)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>O</td>
<td>FO</td>
</tr>
<tr>
<td>ORIGINAL</td>
<td>58</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>54</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>56</td>
<td>50</td>
</tr>
<tr>
<td>Overall Accuracy</td>
<td>67%</td>
<td>Overall Accuracy</td>
<td>30%</td>
</tr>
<tr>
<td>REVISED</td>
<td>86</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>Overall Accuracy</td>
<td>100%</td>
<td>Overall Accuracy</td>
<td>69%</td>
</tr>
</tbody>
</table>

F: Follow-up, O: Operate, FO: Follow-up & Operate, U: Unclassified.
Table 2. Statistics on the number of rules before and after revision.

<table>
<thead>
<tr>
<th>RULES</th>
<th>CLASS</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Follow-up</td>
<td>Operate</td>
</tr>
<tr>
<td>Original</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Deleted</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Unchanged</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>New (from specialization)</td>
<td>53</td>
<td>6</td>
</tr>
<tr>
<td>Total after revision</td>
<td>57</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 3. Statistics on the number of predicates before and after revision

<table>
<thead>
<tr>
<th>PREDICATES</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used in original knowledge</td>
<td>21</td>
</tr>
<tr>
<td>Deleted after revision</td>
<td>2</td>
</tr>
<tr>
<td>Newly used (pain_effect)</td>
<td>1</td>
</tr>
<tr>
<td>Total after revision</td>
<td>20</td>
</tr>
</tbody>
</table>