A Mobile Reasoning System for Supporting the Monitoring of Chronic Diseases

Aniello Minutolo[1,2], Massimo Esposito[1], and Giuseppe De Pietro[1]

[1] Institute for High Performance Computing and Networking, ICAR-CNR
Via P. Castellino, 111-80131, Napoli, Italy
[2] University of Naples "Parthenope" Department of Technology Naples, Italy
{minutolo.a,esposito.m,depietro.g}@na.icar.cnr.it

Abstract. Advances in health care technologies are radically impacting the management of chronic diseases by providing a new long-term care option that combines supportive systems for monitoring and assessing the patients' health status with activities of daily living. In this respect, this paper presents a mobile reasoning system which can be used to build knowledge-based Decision Support Systems for monitoring and managing ubiquitously and seamlessly chronic patients, specifically designed and developed as a light-weight solution suitable for resource-limited mobile devices. The system is devised to offer knowledge representation and reasoning facilities able to face and efficiently reason on the continuous and real-time flow of data generated by the sensor devices with the final aim of providing answers within a prescribed time and given constraints on the processing power and resources.

Keywords: Decision Support, Mobile Inferential Reasoning, Ontologies.

1 Introduction

Nowadays, advances in health care technologies are radically impacting the management of chronic diseases by providing a new long-term care option that combines Decision Support Systems (DSS) for monitoring patients' health status with activities of daily living so as to promote individual independence and well-being.

In particular, knowledge-based DSSs are more and more widely adopted in such scenarios: they model medical knowledge and experts' know-how for inferential reasoning in order to supply alarms as a response to a worsening of the patient's status, plus suggestions about the actions to do. Such a typology of DSSs, typically associated with desktop systems, are recently undergoing a radical transformation in order to face a set of new challenging scenarios, where information must be supplied, received, and/or used anywhere for supporting individuals or organizations seamlessly and ubiquitously in their decision-making tasks.

In this respect, mobile health DSSs for chronic disease monitoring are increasingly appearing on smart phones or Personal Digital Assistant devices (PDAs), with the aim of facilitating self care and communications with physicians by reasoning on data gathered by sensor devices.

K.S. Nikita et al. (Eds.): MobiHealth 2011, LNICST 83, pp. 225–232, 2012.

This has been facilitated by the spread of pervasive computing – a growth in small sensors, wearable or handheld devices, and wireless networking technologies. However, the focus has been twofold. At the lower layers, the attention has been on getting data from the sensors and on creating architectures and frameworks that will support the integration of their data. At the upper layers, efforts have typically focused on using the sensed information to infer conclusions, e.g. a suggestion or an action to do. This has taken several forms, ranging in complexity from simple numerical threshold type systems to more advanced systems that seek to build rich models of the domain described in semantic web languages and then reason over them.

Mobile reasoning over domain knowledge bases generally involves sensed data that are updated at a relatively low frequency – a person entering a room, a device being turned on, etc., and, in this respect, some mobile reasoning systems have been built to handle sensed data formalized by means of semantic web languages [1-3].

Unfortunately, such mobile reasoning approaches do not scale to real-time and computation intensive applications, where sensed data are updated at a very high frequency, e.g. mobile health DSSs where vital signal data are constantly generated by sensors placed on the body.

In detail, on the one hand, they are not able to handle the huge volume of dynamically changing facts and information, since the lack of operators for updating and/or retracting the existing knowledge forces the adoption of the strategy to rebuild the domain knowledge base in accordance with the incoming new data every time, so generating a recurrent inference processing overhead.

On the other hand, they implement computationally heavy inferential procedures neither characterized by a light-weight and efficient implementation suitable for resource-limited mobile devices, nor optimized to provide inferences within a prescribed time and given constraints on the processing power and resources, so involving that the incoming data rate is much higher than the time taken by the inferential procedure.

According to these considerations, this paper presents a mobile reasoning system which can be used to build knowledge-based DSSs for ubiquitously and seamlessly monitoring and managing chronic patients, specifically designed and developed to i) support the definition, updating and retracting of medical knowledge by means of existing ontology languages, such as OWL (Web Ontology Language)[4] and RDF (Resource Description Framework)[5], and a proposed rule-based formalism including non-monotonic operators; ii) provide an inferential reasoning algorithm based on a lazy evaluation [6] to enable the generation of inferences within a prescribed time and given constraints on the processing power and resources. so reducing the space complexity and improving the time response.

2 Mobile Reasoning System

The proposed mobile reasoning system has been designed to support inferential reasoning procedures in mobile DSSs with the final aim of monitoring the health status of chronic patients. The main components of the system are shown in figure 1.

The *Working Memory* (WM) is the repository in which both medical knowledge and experts' know-how expressed in terms of OWL ontologies are stored.

In more detail, at the system start-up, the *Knowledge Base Manager* (KBM) encodes the terminological knowledge describing the specific domain in well-defined and semantically-rich OWL descriptions, expressed in terms of classes and properties, and then rearranges and stores such descriptions into the WM as a collection of facts, i.e. RDF triples expressed in the form of <*subject,predicate,object*>.

The KBM will dynamically update the WM by inserting assertional knowledge consisting in new facts, represented by individuals (instances of concepts) with the corresponding instances of properties.

More specifically, starting from the patient data coming from sensing devices, a set of individuals of the ontology concepts is instanced. Moreover, the values associated to the input data are also associated to the corresponding properties of the defined individuals. All the individuals are finally added to the WM as new facts. The WM will be also updated with the inferred facts, i.e. the new facts generated at the end of the reasoning process.

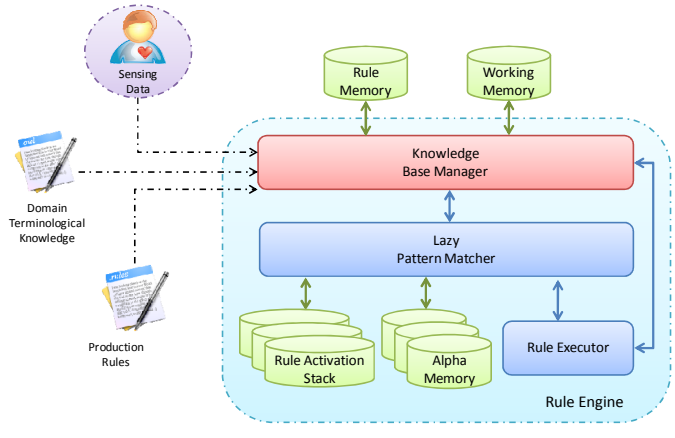

Fig. 1. The main components of the Mobile Reasoning System

The *Rule Memory* (RM) is the repository where production rules (i.e. if-then rules) are stored. The syntax used to formalize the rules has been defined in order to represent, in a natural and understandable manner, procedural knowledge about actions and suggestions to be generated for supporting clinical operators in the management of chronic patients.

Starting from the terminological knowledge expressed in triples, a set of production rules is built, each of them made by a conjunction of condition elements (CE) in its left-hand side (LHS) and a set of actions in its right-hand side (RHS), respectively.

```
name: antecedents -> consequents

where
        name        = a string identifying the rule
        antecedents = bodyterm, bodyterm, …
        consequents = headterm, headterm, …

bodyterm ∈ { (sub,pre,obj) , not(sub,pre,obj) , functionToCall(arg1,arg2,…)}
headterm ∈ { (sub,pre,obj) , functionToCall(arg1,arg2,…)}
```

Fig. 2. The rule syntax

The proposed rule-based formalism, showed in figure 2, has been designed in order to extend the expressiveness of ontology languages, allowing the explicit use of non-monotonic operators enabling closed-world reasoning over the WM, such as *retraction*, for updating an assertion according to new input data, and *negation-as-failure*, for determining negative information in the case of not completely represented knowledge.

Three kinds of term can be used in a CE, namely *triple pattern*, *negated triple pattern* and *function call*. A triple pattern is defined as a generalization of a triple object with the additional property that it can contain variables instead of only static values. A negated triple pattern can be used only in the LHS of a rule and it will be interpreted as the test of the absence in the WM of facts matching the triple pattern. Function calls allow to invoke internal procedures able to evaluate logical conditions, to compute arithmetic expressions, and to retract facts.

The *Rule Engine* (RE) is based on a forward chaining scheme, i.e. a data driven method that can be described logically as repeated application of the generalized modus ponens. In other words, available data are supplied as facts and used to evaluate eligible rules and draw all possible new inferred facts. The most computation intensive and resource-consuming part of the inferential reasoning is the matching process, which determines the set of satisfied rules that can be executed given the current set of facts, since it is a hard combinatorial problem.

In order to grant an efficient handling of memory and computational resources and achieve good performance, a *lazy pattern matching algorithm* has been proposed, specifically designed and implemented as a light-weight solution suitable for resource-limited mobile devices.

The idea of the algorithm is to evaluate the so called *rule activations*, i.e. which rules are satisfied and which data satisfy them, with the final aim of computing only one rule activation in each cycle, based on the observation that only one of them is fired anyway.

Differently, other matching algorithms [7, 8] first compute all the applicable rule activations and then execute them according to a selection strategy, and, thus, are characterized by an exponential worst-case complexity in terms of time and space.

The lazy evaluation does not waste time in computing superfluous rule activations which could be never executed and enables to fire the first eligible rule as soon as it has been identified, so as to improve the performance in terms of average response time. Moreover, by avoiding the computation of all possible rule activations, the worst-case space complexity can be reduced to a polynomial bound.

Thus, in the case when applied to remote monitoring scenarios, the proposed lazy algorithm is able of efficiently elaborating the huge volume of vital signal data, for example generated by body sensors, thanks to its reduced space complexity, and, contextually, of facing the real-time and computation intensive requirements by granting better performance in terms of response time.

More in detail, the whole inferential process is described as follows. The *Lazy Pattern Matcher* (LPM) operates on working memory elements (WMEs) expressed as RDF triples characterized by a unique and incremental ID.

In order to determine eligible rule activations, the LPM builds memory structures, named *alpha memories,* for explicitly storing information about the results of *intra-condition* tests. *Intra-condition* tests apply to determine which WME satisfies or violates a specific CE of a rule and store the results into the associated alpha memory. Since the same CE can occur in many rules to be successively evaluated, an alpha memory can be shared between different rules.

A non-negated CE usually contains variable references to be bound in order to assume values according to the matching facts stored into its alpha memory. When the same variable reference occurs multiple times in different CEs of a rule, the consistent binding of that variable must be evaluated by means of *inter-condition* tests that involve multiple CEs. If a variable occurs only once in the LHS, no test is necessary.

Thus, for a rule, the combination of facts stored into non-negated alpha memories which satisfies the inter-condition tests, does not violate any negated CE and verifies the conditions specified in any function call represents a *rule activation*.

To support the research of eligible activations for each rule, i.e. the inspection and combination of facts stored in the alpha memories, trying to find a match for the whole rule, the LPM maintains *a rule activation stack*. When a WME satisfying the intra-condition test of a non-negated CE is found, a new element is pushed onto the stack by the LPM, which contains both the IDs of the last recent facts inserted into every non-negated CE and a reference to that specific CE (*Dominant Reference*).

Each stack element enables to enumerate a subset of potential rule activations by combining the facts stored into the non-negated alpha memories according to the DR and the IDs stored into the current stack element.

After investigating all the available combinations associated to the current stack element, another stack element is popped. When the stack is empty, no combination of facts can be further tried, and, thus, no other rule activation can be found. The rule activation research is computed only if the rule is active and its stack is not empty. A rule is active when its non-negated alpha memories are not empty and its negated alpha memories are empty.

As soon as an eligible rule activation is found, the research of other possible rule activations is paused by storing a pointer to the interruption point into the stack, and, then, the corresponding rule is executed by the *Rule Executor*.

Successively, if the WM is not changed at all, the research is resumed from the last interruption point, i.e. by removing the next element from the top of the stack. Otherwise, if the assertion or retraction of facts has been generated in the WM by the executed rule or by new input data produced by the sensing devices, the LPM recognizes the rules which can be involved by these changes and updates both the alpha memories associated to their CEs and the related rule activation stacks. After that, the LPM determines the new active rules with a non-empty stack, selects one of them and restarts the research of its rule activations. The whole lazy matching procedure is summarized in figure 3.

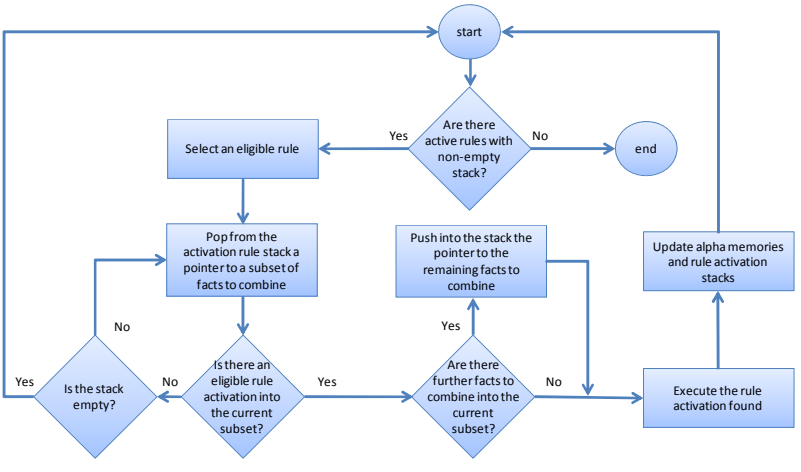

Fig. 3. The main components of the Mobile Reasoning System

It is worth noting that the proposed lazy approach does not save the intermediate results of inter-condition tests, but it recalculates them every time it is required. Other pattern matching algorithms [7] save all partial results of inter-condition tests into so-called beta memories in order to compute them only once and store them for later reuse.

However, the time gained by using such additional memory should be weighed against the inherent cost with respect to the specific application scenario. In particular, applications for chronic patient monitoring, where input data are constantly sent by sensor devices, require a continuous refresh of facts in the WM in terms of assertions and retractions. Thus, beta memories should be frequently updated, so as to make the storage of intermediate inter-condition tests' results useless, and, thus, decrease both space and time performance. As a result, omitting the memory support for inter-condition tests as proposed in the presented algorithm represents a significant factor to improve space and time performance in real-time and intensive applications.

3 Implementation and Results

The overall mobile reasoning system has been implemented for resource-limited mobile devices by using Java 2 Platform, Micro Edition (J2ME) in accordance with the Mobile Information Device Profile 2.0 (MIDP) and Connected Limited Device Configuration 1.1 (CLDP).

As a proof of concept, it has been applied to the case study described in [9] for monitoring cardiovascular diseases. In particular, to test the feasibility of the proposed system, the medical knowledge defined in [9] and formalized in terms of ontologies and rules has been used with the aim of detecting potentially abnormal situations.

Seven rules with an average number of antecedents equal to nine are stored into the *Rule Memory* (for more details about the rules used refer to [9]). The monitored parameters are: the heart rate, the estimated Standard Deviation Normal beat to Normal beat (SDNN), the patient's posture and his/her physical activity.

In accordance with the consideration that the admissible measuring range for the heart rate varies from 30 bpm (i.e. beat per minute) to 240 bpm, the heart rate can change its value up to a maximum of four times per second. This involves that the system has to reason on a variable number of heart rate measures per second, varying from 0 to 4. As a result, the system has been tested supposing the heart rate value is updated at a frequency equal to 1Hz and, also, the other parameters change their values at the same frequency.

Taking into account such considerations, five different measures for each parameter have been inserted into the *Working Memory,* where the different parameters are supposed to have been simultaneously acquired at each measurement. The five measures have been built ad hoc in order to activate rules reporting an abnormal situations only in three out of five cases.

The system has been deployed and tested on two mobile smart phones with comparable hardware features, namely Nokia N8 and HTC Legend, in order to evaluate its behaviour and effectiveness with respect to two different java compliant platforms, i.e. Symbian and Android.

In particular, with respect to both the Nokia N8 and HTC Legend, the overall reasoning time is approximately equal to 56 ms, whereas the response time required to detect and execute the first rule activation, i.e. the first abnormal situation, is more or less equal to 30 ms. So, the response time is sensibly less than the overall reasoning time, which can be reasonably equated to the response time of non-lazy approaches, where a response is generated only at the end of the whole reasoning process.

As a consequence, independently of the mobile device used, the proposed system is shown to be proficiently applicable to the presented case study regarding the cardiac monitoring. Indeed, on the one hand, it meets the real-time performance demand, since its response time is strongly less than the updating frequency of the monitored parameters. On the other hand, its response time is reasonably supposed to outperform the performance which could obtained by using classical approaches, even if this cannot be actually verified since, currently, the code of previous works supporting mobile reasoning are not accessible.

4 Conclusions

In this paper a mobile reasoning system able to build knowledge-based DSSs for monitoring and managing ubiquitously and seamlessly chronic patients has been presented. The system offers knowledge representation and reasoning facilities to face and efficiently reason on the continuous and real-time flow of data generated by the sensor devices.

The core of the reasoning system is a lazy pattern matching algorithm, specifically designed and implemented as a light-weight solution for granting the efficient handling of memory and computational resources and achieve good performance especially in real-time and intensive applications.

The proposed mobile reasoning system has been implemented for resource-limited mobile devices by using Java 2 Platform, Micro Edition and deployed on two mobile devices, namely Nokia N8 and HTC Legend, with comparable hardware features, in order to evaluate the library's behaviour and effectiveness with respect to two different java compliant platforms, i.e. Symbian and Android.

As a proof of concept, it has been applied to the case study described in [9] for monitoring cardiovascular diseases, showing its effectiveness to meet the real-time performance demand of that scenario.

Next step of the research activities will be to minutely compare the mobile reasoning system with respect to other existing one, in terms of performance evaluation and experimental assessment, and to apply it to other mobile health scenarios where patients affected by chronic pathologies have to be monitored.

References

1. Sinner, A., Kleemann, T.: KRHyper - In Your Pocket. In: Nieuwenhuis, R. (ed.) CADE 2005. LNCS (LNAI), vol. 3632, pp. 452–457. Springer, Heidelberg (2005)
2. Ali, S., Kiefer, S.: μOR – A Micro OWL DL Reasoner for Ambient Intelligent Devices. In: Abdennadher, N., Petcu, D. (eds.) GPC 2009. LNCS, vol. 5529, pp. 305–316. Springer, Heidelberg (2009)
3. Kim, T., Park, I., Hyun, S.J., Lee, D.: MiRE4OWL: Mobile Rule Engine for OWL. Accepted for publication at the 2nd IEEE International Workshop on Middleware Engineering, ME 2010 (2010)
4. Patel-Schneider, P., Hayes, P., Horrocks, I., et al.: OWL web ontology language semantics and abstract syntax. W3C Recommendation 10 (2004), http://www.w3.org/TR/owl-semantics/
5. Resource Description Framework, http://www.w3.org/rdf/
6. Weert, P.V.: Efficient Lazy Evaluation of Rule-Based Programs. IEEE Transactions on Knowledge and Data Engineering 22(11), 1521–1534 (2010)
7. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19, 17–37 (1982)
8. Miranker, D.P.: TREAT: A New and Efficient Match Algorithm for AI Production Systems. PhD dissertation, Columbia Univ. (1987)
9. Minutolo, A., Sannino, G., Esposito, M., De Pietro, G.: A rule-based mHealth system for cardiac monitoring. In: The 2010 IEEE EMBS Conference on Biomedical Engineering (IECBES 2010), Kuala Lumpur, Malaysia, November 30-December 2, pp. 144–149 (2010)