KQML

Partial report on

A proposed knowledge acquistion language

For intelligent applications

based on discussions

At a DARPA, NSF, AFORS sponsored
Workshop in Santa Barbara
March 1990

Gio Wiederhold
Stanford University
April, 1990, revised January 1991

1. Introduction

The report comprises the work of group 2: External interfaces to Knowledge-based Systems. It consists of the following major components:

1 A classification of the modules that may be part of a large multi-site knowledge-based system and their interfaces.
2 A proposal for a KQML to serve some specfic interfaces in such a system
3 Protocol definitions and examples for the use of such a KQML
4 Details of the MOVE (ask/TELL) operation to be used within the KQML
5 Tasks to be carried out next to proceed in the standardization effort.

1.1 A Classification

When we are talking about future large knowledge-based systems we envisage systems having many components. Organizationally they will differ depending on their layer, i.e., are they close to the user, are they deeply buried internally, or are they the interface with the real-world? Figure 1 shows these interfaces, and indicates also the external interfaces needed. External interfaces should be ‘friendly’, but user-friendliness is not directly part of this standardization effort

For the transmission of knowledge among these components a full set of internal interfaces among can be visualized. These are sketched in Figure 2.

1.2 Modules in a Knowledge-based System

To define the interfaces we will first give names to the typical component node types.

1.2.1 EUA

End-user-applications. Here knowledge is acquired, possibly from multiple sources, fused, analyzed, and presented. Enough meta-knowledge must be made available to recognize sources, cost, expected volumes, and presentation devices. Knowledge of the source may lead to assessment of quality and completeness.

```
 -----------------------------------------------------------------------
                       --------------------------     -----------
                      | End-user Applications  |     | Other    |
    End-users <------->|    ( EUA )             |     |  EUAs    |
                      | mainly on workstations |     |          |
                      -------------------------     -----------
                       ---------------------------      -----------
                      | Knowledge-based systems |      | Other    |
    Knowledge         |     ( KB )              |      |   KBs    |
       engineers <---->|    on processing nodes  |      |          |
                      |                         |      |          |
                      ---------------------------      -----------
                                   -----------------        ---------
    Knowledge maintainers <------->| Knowledge      |        | Other |
                                  |  repositories |        |  KBRs |
                                  |      ( KBR )  |        |       |
                                  -----------------        ---------
    Database                 --------------------         ---------
        maintainers <---->| Database systems |         | Other |
                          |   ( DB )         |         |  DBs  |
                          --------------------         ---------
    Real             ------------------           ---------
       World <---->| Active Sensors |           | Other |
                    |   ( AS )       |           |  ASs  |
                    ------------------           ---------
       <--- friendly interfaces --->      <--- vigorous interfaces --->
 -----------------------------------------------------------------------
```

Figure 1. Potential major modules of Knowledge-based Systems

1.2.2 KB

Knowledge-bases. Here knowledge and meta-knowledge is made available to EUAs, in response to their requests. KBs may monitor data resources (DBs) and active sensors (ASs) in order to acquire and monitor knowledge. Responses to EUA's may be defered until certain conditions in source DBs or ASs are met.

1.2.3 KBR

Associated with KB will be a repository for KB content (KBR), to assure persistency of the knowledge. Often a full history will be kept, so that older versions of a KB will be accessible.

1.2.4 DB

Here we group conventional databases and datafiles. Their focus is collecting and storing information persistantly. Such DBs are typically maintained and updated autonomously to reflect accuratly operational status of an organization or operation. Such requirements inhibit close coupling to KBs.

1.2.5 AS

We must also consider direct input from real-world phenomena. Under active sensors (AS) we group all data sources that are not persistent. This means that subsequent requests are likely to provide new values, and older values will not be retrievable. An extension, not specifically considered here, are actuators that directly effect change on the real world.

1.3 The Interfaces

We can now discuss systematically the possible interfaces for these five system nodes, and indicate our focus. Since we do not distinguish direction we have 1/2(n*(n+1)= 15 potential interface types for 5 module types. In Figure 2 only one instance of each interface is shown.

```
------------------------------------------------------------------------
                      The end user
      ----------------------------------   (1)   -----------
     |       EUA                      |<------>|   EUA     |
     ----------------------------------   IL    -----------
         WW               v      v    v
         ||               |      |    |             etc
         ||(2) KQML       |(3)   |(4) |(5)
         ||               |      |    |
         WW               |      |    |
   -------------------    |      |    |   (6)   -----------
   |       KB        |<----------------------->|   KB      |
   -------------------    |      |    |   IL    -----------
     WW    WW     WW      |      |    |
     ||    ||     ||(7)   |      |    |             etc
     ||    ||(8)  WW      v      |    |
     ||(9) ||   -------------    |    |  (10)   -----------
     ||    ||   | KBR       |----------------->|  KBR      |
     ||    ||   -------------    |    |   IL    -----------
     ||    ||     v      v       |    |
     ||    ||     |(11)  |(12)   |    |             etc
     ||    WW     v      |       |    |
     ||   ------------   |      /     |         -----------
     ||   |   DB      |<---------     |  (13)   |   DB      |
     ||   |           |<---------------------->|           |
     ||   ------------   |            |   DBD   -----------
     ||       WW         |            |
     ||       ||(14)     |            |             etc
     WW       WW         v            v
   ------------------------------------- (15)   -----------
   |         AS                       |<----->|    AS      |
   -------------------------------------        -----------
                        The real world
    || major flow                    IL   Interlingua functions
    | minor or null flow            DBD   Distributed Database function
------------------------------------------------------------------------
```

Figure 2. Internal interfaces among major modules.

1.3.1 EUA to EUA

If an application (EUA) accepts a server role for another EUA, it must satisfy the requirements we place on a general knowledge server (KB), as defined as (2) below.

1.3.2 EUA to KB

The transmittal of knowledge or knowledge-based derivations from knowledge servers to applications is a major path and is the focus of this document.

1.3.3 EUA to KBR

Direct access to the representation of knowledge (KBR), without the mediation of a (KB) can provide history, structure, and other information about the corresponding KB. However, uninterpreted knowledge may be risky to use. We will discuss requirements for such access in this document.

1.3.4 EUA to DB

Standards for DB access are well established, as SQL and RDA. They are however fairly awkward and not Turing-complete. We will not discuss those here. A number of commercial products and semi-products exist here. For instance, the Apple CL initiative generalizes such access for multiple SQL-based DBs. Often a LISP interface to standard SQL is all that is required and provid intelligent access to databases.

1.3.5 EUA to AS

Differences in sensor technology have inhibited high-level standards here, although a variety of low-level standards exist. We will not discuss these here. As sensor-based systems start using more standard types of computing equipment we can expect to achieve more commonality.

1.3.6 KB to KB

This is the focus of the concurrently ongoing Interlingua effort of group 1. It may only be effective where the ontologies match or can be circumscribed.

1.3.7 KB to KBR

Questions on how to store knowledge representations in an effective and persistent way are part of our charter. While we do specify some high-level operations in this document, it appears that any detailed specification will be difficult to implement effectively and identically throughout the wide variety of possible KB implementations.

Even though this interface falls within the range of responsibility of our subgroup, we currently view this interface as *not* very amenable to standards specifications because of efficiency demands. Such efficiencies are gained today by very close coupling.

1.3.8 KB to DB

To make KBs effective it is desirable that all voluminous factual information be maintained outside of the KB, although this may engender high costs in terms of performance. Research into suitable techniques, as caching, close coupling, etc., is in progress in many research sites. It appears that existing DB standards are insufficient to serve KBs well. Object-oriented approaches are of interest. We are not now focusing on this issue.

1.3.9 KB to AS

A KB can have an important role in abstracting and monitoring sensor (AS) information. Certain information, when acquired, will force changes in knowledge, so that some learning mechanism much be invoked. A special case here is an active DB, which will also signal changes that may require knowledge update. This field is not well understood now, and probably not ready for standardization.

1.3.10 KBR to KBR

Interchange of knowledge content is hard to distinguish from a KB to KB interface, although it seems feasible if the internal representations match. Use of an Interlingua may motivate such internal consistency.

1.3.11 KBR to DB

Active research and development is ongoing to use DB facilities for persistent storage of knowledge. Although current DBs are too inflexible, there seem to be no principles that would inhibit use of DBs. The focus will be on efficiency. Again — object-oriented structures hold out some promise in this arena.

1.3.12 KBR to AS

This interface does not make sense.

1.3.13 DB to DB

Distributed databases use replicated data to enhance availability. Consistency requires use of concurrency protocols. This is a relatively mature area, and outside of the purview of this work.

1.3.14 DB to AS

Databases can be used to make sensor data persistent, but the update facilities provided by SQL are implementation and performance bottlenecks here. The DADAISM database design is configured to accomodate sensor data.

1.3.15 AS to AS

This is not a meaningful interface. Sensors are only senders of information, not receivers. An extension of the AS module to encompass actuators could enable this interface, but no intelligent processing would occur.

This enumeration of interfaces helps us in defining precisely what interfaces we are addressing now. The highest interest is in EUA $\leftarrow$ KB, with some need for EUA $\leftarrow$ KBR, in order to have access to some internals of the knowledge representation.

2. KQML

We now define a Knowledge Query and Manipulation Language. The queries will typically be issued by End-User Applications (EUAs) and directed towards systems providing processing, representation, and storage support of knowledge (KBs and KBRs). The atomic objects being transmitted are to be Interlingua knowledge units.

2.1 Language layers

The KQML has to provide facilities for the items of knowledge themselves, for the commands to move the items from knowledge bases to applications, and assure integrity of the transmission. Corresponding with these three objectives we specify three layers in the KQML.

1 At the base layer we find the *Interlingua*, as defined elsewhere.

2 At the transmission layer we find MOVE commands, specifically

  2.1 ASK: request Interlingua items

  2.2 TELL: ship Interlingua items

  All responses to ASK requests are themselves in TELL format

3 At the control layer we find the commands to manage the interaction of applications and knowledge sources. The primary command here is PRESENT, which accesses a knowledge base and initializes the interaction.

The PRESENT operation defines a handle for a specific knowledge acquisition transaction. All subsequent interactions use this handle, permitting flexible interaction during the transaction. Such flexibility is essential to deal with the unpredictable volume, content, significance, and complexity of the knowledge being acquired.

The use of handles also permits asynchrony and parallel use of multiple knowledge sources.

The ASK operation can request basic knowledge sentences, as well as descriptive information. Descriptive information is essential to identify the domain and range validity of knowledge being acquired. A difficulty issue will remain: the description of the ontology that is inherent in the particular knowledge server. However, since a KB, more than just a KBR, will include its own interpreter, we can assume that the specific interpreter in a KB does match the ontology of its knowledge base.

2.2 Operation

A typical, simple operational sequence could be as follows:

```
-------------------------------------------------------
         ...
        ...
handle = PRESENT TO <travel-advisor@United.Denver.Co>   -- name of the KB
              ASK (New-York, X, Santa-Barbara            -- parameters
                     COND(Aircraft-class,jet) )          -- constraint
              (SIZE 10) (ORDER Cost);
        ...
         ASK handle store-area                            -- action !
        ...
         IF (handle = IDLE) DEACTIVATE handle             -- complete ?
        ...
    --   Process information placed in store-area
-------------------------------------------------------
```

Figure 3. Basic KQML process.

Note that these statements do not represent the syntax to be proposed, they only indicate the operational sequence.

The additional complexity of having a distinct layer for PRESENT and DEACTIVATE is motivated by the encapsulation it provides for the basic ASK and TELL processes. We sketch another example.

```
-------------------------------------------------------
         ...
        ...
handle-a = PRESENT TO <travel-advisor@United.Denver.Co>
                 ASK (New-York, X, Santa-Barbara)
                 (SIZE 10) (ORDER Cost);
        ...
handle-b = PRESENT TO <bulletin-board@University-center.NYC.NY>
                 ASK (New-York, X, Santa-Barbara
                       COND(class=ride, arrive < 3March90)
                 (SIZE 10) (ORDER departure);
        ...
         ASK handle-a flight-store-area
        ...
         ASK handle-b rides-store-area
        ...
         IF (handle-a = IDLE) DEACTIVATE handle-a   -- have primary result
         IF (handle-b = IDLE) CONTINUE              -- compare with
        ...                                                                          -
-    secondary result
        -- Process information placed in store-areas
        -- decide if rides look good
        -- decide if want to look at more rides
        -- are there many more
          ESTIMATE handle-b ride-count               -- try to improve
        -- look for more                             --    joint answer
         ASK handle-b rides-store-area
        -- More processing
        -- Give up
         DEACTIVATE handle-b
         ...
-------------------------------------------------------
```

Figure 4. Parallel activation of two KQMSL transactions.

The protocol is hence intended to provide the capability to manage

3.a voluminous (and unpredictable volumes of) Interlingua responses

3.b multiple concurrent transmission streams.

One specific requirement being fulfilled by the structure of the KQML, not served by most other approaches, is the capability to keep an application in abeyance until a change of status occurs in the underlying server, typically due to a new sensor input or an update of a database. A system built using KQML, hence, can be an active participant in a real-time situation, and serve as an intelligent monitor.

We have to defer the specific discussion of the Interlingua layer to the outcome of the concurrent standardization effort for this component. Since that effort focuses on the KB—KB transfer, some adaptations may be imposed from the KQML specifications. We will now discuss the other two layers in greater detail.

We start in the next section with outer, controlling layer (PRESENT). We refer to the central-layer commands as MOVEs and to the innermost layer as Interlingua. We use IK to denote chunks of Interlingua knowledge. We present all interactions in terms of the end-user application (EUA) acquiring knowledge from a knowledge based system (KB), although we expect that KBs may be layered themselves and communicate in an identical manner.

3. Protocol for EUA-KB transfer

The protocol defines the sequence of steps used to obtain knowledge. In order to make efficient execution possible we define a protocol that enables techniques as precompiling, caching, prefetching of IK and the like. For convenience, however, we describe this protocol here as OBif it were interpreted.

```
 -------------------------------------------------------------------------
      --  CONTROL
               Create and destroy           EUA ------> KB
               Acknowledge                  EUA <------ KB
      --  INFORMATION FLOW
            PRESENT request                 EUA  ------> KB
            Acknowledge                     EUA  <------ KB
           --  OBTAIN KNOWLEDGE
                  ASK for transmission      EUA  ------> KB
                  TELL results              EUA  <------ KB
           --  OBTAIN OPERATIONAL KNOWLEDGE
                  ESTIMATE cost             EUA  ------> KB
                  TELL estimation           EUA  <------ KB
           --  GIVE KNOWLEDGE
                  TELL Updates              EUA  ------> KB
                  TELL Acknowledgement      EUA  <------ KB
             DEACTIVATE request              EUA  ------> KB
            Acknowledge                     EUA  <------ KB
  -------------------------------------------------------------------------
```

Figure 4. KQML protocol steps.

3.1 Overview

The protocol consists primarily of 4 steps, a., b., c., and d. The protocol is always entered at step a. and terminates after step d. A fifth category, e., assembles control functions. Each step comprises an action to be presented (.p) and a reaction (.r).

a Initialization

a.p PRESENT the MOVE query text

a.r Receive a handle on the query or manipulation command (QMH), or an error message via TELL.

The EUA can now proceed to step b. or to step c. Large interactions will typically require step c. first.

b. Execution

b.p An ASK command specifies the QMH handle with parameters for execution

b.r The EUA now expects an asynchronous TELL response containing Interlingua values (IK) or meta information.

c. Estimation

c.p An ESTIMATE command requests a cost and volume estimate for the actions implied by the QMH

c.r The EUA expects an asynchronous, but fairly rapid, response

d. Deactivation

d.p A DEACTIVATE cpmmand presents the QMH for termination.

d.r The EUA receives an asynchronous, but rapid, acknowledgement. Only after this acknowledgement is the EUA assured of no more interactions for this QMH.

3.2 ASK

The ASK operation with the handle with a specific destination name initiates the actual transmission request. Multiple ASK (or ESTIMATE) commands can be executed for one handle, so that information can be acquired from the knowledge server in manageable units. All parameters have been set during the initial PRESENTation, so that interaction now can be rapid. The responding TELL interactions provide chunks of IK knowledge.

3.2.1 Estimation

The estimation operation is the innovation in this approach, although there are precursors in bibliographic information retrieval systems. The responding TELL commands here provide meta-information, to permit scheduling and optimization of ASK commands using the same handle. The responses also tell a corresponding uncertainty for each estimate. A high estimate may cause the query to be deactivated, reformulated, and resubmitted. A significant problem in knowledge and data retrieval is that the volume of information being requested is not known a priori. Excessive volume may exceed the receivers' physical capability and may also imply that the request was not well framed.

Estimation (steps c.) may be interleaved with execution (steps b.), since after some execution the quality of the estimate will improve. The estimation steps must depend on heuristics, since the costs of a detailed evaluation would be close to the execution cost and hence ineffective.

3.3 Control

Some KQML operations will not require the four-step interaction. These are mainly meant to manage KBs and KBRs, and are the high-level operations sketched in the definition of that interface.

e.p A single control function will submit control commands as

e.p.1 Initialize an empty KB on some system node

e.p.2 Load a KBR into a KB

e.p.3 Dump the contents of a KB into some KBR

e.p.4 Clear a KB

e.p.5 Destroy a KB

e.r An asynchronous acknowledgement should be received before proceeding.

3.4 Asynchrony

The asynchrony which is supported throughout the protocol has several motivations

1 The EUA can initiate several concurrent requests.

2 Some requests will depend on conditions in the KB which will cause deference until some external event, perhaps from an AS, occurs.

3 The EUA can proceed with a best guess computation until relevant knowledge arrives.

The labeling of all presented actions and reactions with a handle (QMH) makes it possible for the EUA to deconvolute complex interactions.

The asynchrony specified also implies that many distinct knowledge bases and data resources can participate, in a variety of configurations, in any application.

The execution site for a KB is not bound to fixed site for the stored representation (KBR), and multiple KBs can be used by an End-User Application (EUA) on local and remote nodes. Since KBRs can be loaded into a new and clean KB a local execution can be independent of other activities. Owners of a KBR may wish to protect their KBR from being destroyed by a EUA dumping a modified IK into their KBR.

4. MOVE operations

The principal commands to define movement of Interlingua knowledge chunks (ILK) are

1 ASK: request a collection of interlingua statement

2 TELL: send a collection of interlingua statements

Parameters are associated with these statements to more fully define their function are set prior, by the PRESENT operation referred to via the QMH handle. These parameters achieve the following objectives:

1 Acquire or set meta-knowledge

2 Partition the retrieval into manageable segments. Each execution of a MOVE command then only moves one segment, although the initital MOVE command definition has specified the entire set. Each execution is in practice a NEXT command.

3 Define actions for a variety of abnormal conditions - typically override or abort.

4 Specify further computations on the defined set of Interlingua statements, as ordering or derivation.

4.1 Syntax of MOVE commands

¡to be defined¿

All parameter specifications are to follow Interlingua syntax.

¡We should borrow the meta language or BNF from the Interlingua document as well.¿

4.2 Parameters

We foresee a number of parameters for the various steps. The same parameter may be interpreted somewhat differently, specifically during estimation versus execution.

Since our experience in this area is limited we propose an extensible syntax for these parameters. We expect that those parameters will follow Interlingua syntax, and be mainly distinguished that their constants will be bound to system values. Default values are presented with an exclamation mark (!). The ∞ symbol indicates that the default is a very high number,

We will refer to the expected content of the MOVE (TELL and ASK) actions as the Interlingua Knowledge or IK. Many of the parameter names are valid for both query (i.e., ESTIMATE), as well as for setting operations.

4.2.1 VOLUME !∞

Control or inquire into the expected volumes in terms of bytes of the ILK representation. The high default result on queries is to ensure that the receiving EUA is prepared for an answer of undeterminate size.

Setting a bound will packetsize the ILK result into multiple segments for execution.

4.2.2 SIZE !∞

Control or inquire into the expected number of ILK statements. A bound on requests will similarly cause that multiple executions may be needed to retrieve all of the ILK.

4.2.3 TIMECOST !∞

Set a bound or inquire into the temporal delay before any ILK results will arrive.

4.2.4 EXTENT !any

The Extent parameter serves to control or inquire into the temporal range of the ILK.

4.2.5 SCOPE !all

This parameter is intended to control or inquire about the scope of the theory represented by the ILK referenced. Its specific operation will depend greatly on the higher-order facilities of the Interlingua.

4.2.6 VERSION !latest

Control or inquire about a version designator assigned to the KBR in use by the active KB.

4.2.7 ORDER !unordered

Specify or inquire into the ordering applied to the ILKs being obtained.

4.2.8 [!NO] META ONLY

Return (ASK) only the schema of the ILK specified, without any actual ILK values. This implementation will depend greatly on the capability of the Interlingua and the KBS, and should return any information that is not strictly first-order logic.

4.2.9 [!NO] TYPE

Only return (ASK) the type hierarchy or lattice of the ILK specified, not its values.

4.2.10 [NO] !SUBSUMPTION

Indicate the [no] check is to be made if IK statements subsume each other in the result (ASK) or in the KBR (TELL), i.e. return (ASK) all subsumed predicates if subsumption is indicated. Eliminate subsumed predicates from result (ASK) or KBR (TELL) if no subsumption is indicated.

4.2.11 [!NO] SUPERSUMPTION

Return (ASK) all shared predicates.

4.2.12 [NO] !CASCADE

Set (TELL) or retrieve (ASK) with complete unification if cascading is indicated. Set (TELL) or retrieve (ASK) only the primary IK specified if no cascading is indicated.

4.2.13 [NO]!DUPLICATES

Indicate that duplicates are or are not acceptable in the result (ASK) or in the KBR (TELL).

4.2.14 [NO]!CONTRADICTIONS

Indicate that contradiction are or are not acceptable in the result (ASK) or in the KBR (TELL).

4.2.15 [NO]!AMBIGUITY

Indicate that ambiguous IK statements are or are not acceptable in the result (ASK) or in the KBR (TELL).

4.2.16 [!NO]PREDICATE MATCH

Report on [or ignore] unmatched predicates in MOVE commands.

4.2.17 [NO]!SYNTAX CHECK

Report on [or ignore] syntax errors in MOVE commands.

5. Follow-up Tasks

5.1 Clarify

Discuss, clarify, and alter the KQML proposal

5.2 Refine

Accomodate proposal to Interlingua specification, format, and meta-specification.

A suggested time is March 1991, after the first presentation of the KIF proposal for Interlingua.

Write the specification in a formal syntax. Use either the Interlingua Syntax or the ANS version of the YACC syntax.

5.3 Validate

Do a simple demonstration or simulation of EUA — KB interaction which follows the protocol.

5.4 Disseminate

5.5 Prototype Implementation