Structuring and Modules for Knowledge Bases: Motivation for a new model

Grigoris Antoniou
Fachbereich Mathematik/Informatik
Universität Osnabrück
Albrechtstrasse 28
D-4500 Osnabrück
Germany
e-mail: ga@informatik.uni-osnabrueck.de

Ipke Wachsmuth
Technische Fakultät
Universität Bielefeld
Universitätsstrasse 25
D-4800 Bielefeld 1
Germany
e-mail: ipke@techfak.uni-bielefeld.de

Contact information: Please address correspondence to the first author

Abstract: Evolving out of theoretic and practical work, this paper presents the motivation and basic ideas for the construction and use of modular knowledge bases. The approach bears upon earlier work by the two authors done separately from each other. We introduce a model which attempts to merge two previous approaches while maintaining their benefits: Modules for logical knowledge bases, and ordered by generality domains. Central aims are reusability, restriction of memory search, and management of inconsistent (competing) knowledge within one knowledge base. We explain the model by motivational examples and discuss the formal semantics of structured, modular knowledge bases for knowledge representations that are based on logic programming.

Keywords: Knowledge representation in the large, knowledge sharing, consistency

Category: Scientific Conference - techniques&methods (theory also possible)
1 Introduction

As artificial intelligence technology is moving towards more ambitious applications, the development of large-scale knowledge bases has become one of the most challenging tasks. Although many knowledge based systems restrict their applicational scope to narrow world domains, experience has shown that for more advanced applications like text understanding (e.g. [5]) richer knowledge bases are necessary. We believe that in order to manage such knowledge bases structuring is essential. Let us give some reasons for this claim:

• In large knowledge bases it is necessary to restrict the search space of deduction by way of principle and not simply based on heuristics. In the ideal case only the relevant domain segments should be considered. Also, it should be possible to add knowledge to a knowledge system without making the deduction process much slower.

• Though common sense knowledge contains inconsistencies, humans can deal with them without problems. Structured knowledge bases provide a possibility for managing inconsistencies by considering partitions that can be selectively accessed. In this case, it is possible to have alternative views on a knowledge base leading to context-dependent answers.

• Finally, structuring makes knowledge bases more easily comprehensible and maintainable, an important task in any large system where several developers and users are involved in the assimilation of increasingly many knowledge items [7].

Starting point of our work is a model for structured knowledge bases developed by the second author in [9], [10] whose basic ideas are grounded on findings from empirical research about how human knowledge is structured. The findings suggest that a major feature of human intelligence lies in focussing on a part of the entire knowledge small enough to be tractable. In case a problem cannot be solved in a satisfactory way, other (perhaps competing) parts of the knowledge must be tried.

According to this approach, knowledge is partitioned into knowledge packets by semantical (content-related) criteria. Packets should consist of knowledge elements that are closely related to each other and should define a well-defined, self-contained part of the knowledge base. At any time there is one module called focus denoting the most specific knowledge available for the moment. Only knowledge contained in the focus or in more general modules is active.

This approach (presented in section 2 in some more detail) is appropriate for managing inconsistent knowledge in one knowledge base. Neverthe-
less, it can be criticized as not addressing modularity for consistent parts of the knowledge base in a sufficient way. Also, the knowledge parts (packets) are not independent from their environment, so reusability is not achieved. Note that these points of criticism are well-known as benefits of module concepts in computer science. A theory of modularity for logical knowledge bases is presented in [1], [2]. According to this approach, modules are independent entities communicating with their environment via their interfaces. A knowledge base is obtained by combination of several modules by gluing their interfaces. More details can be found in section 3.

The central theme of the present paper is to combine these two approaches to the extent possible while maintaining their advantages, i.e. management of competing knowledge on one side, and modularity and reusability on the other. The new model is described in section 4, while in section 5 we give a formal foundation of the new model based on logic programming.

2 Managing competing knowledge

In the context of text-understanding systems [4], earlier work of the second author was devoted to the development of large-scale knowledge bases containing semantic background knowledge. Findings from an empirical study on human knowledge organization led to principles of domain-oriented knowledge structuring [9] which take specificity as an aspect of hierarchic knowledge organization. Some principles pertain to the way how the knowledge elements are organized in a static knowledge base. Other principles concern the way in which structured knowledge is accessed and made available to the knowledge-based system. Static and dynamic access conditions give rise to a finite set of knowledge base partitions (packets) each of which can be considered as an autonomous knowledge base characterizing a certain segment of the modeled domain. The core ideas of this approach are sketched below.

A structured knowledge base consists of a number of parts (packets) that are organized as ordered by generality domains. A simple example of such a knowledge base is illustrated in figure 1.

![A structured knowledge base](image1)

![P3 is the current focus](image2)

Figure 1
The packet structure is static and is only changed when reorganization of the knowledge base is necessary, but not at runtime. Inner packets contain more specific knowledge. For motivation, in the example above, P1 could contain information about English language, P2 about the idiom spoken in Scotland, P3 about the language spoken at universities and P4 information about specific words and phrases used by computer science researchers and students. Note that the outmost module includes those knowledge items that are useful for all problems in the area the KBS is to be used in. In contrast, P2 and P3 may comprise competitive knowledge to be used alternatively.

According to the principle of focussing, at any time there is one module called focus denoting the most specific knowledge available for the moment. Only knowledge contained in the focus or in more general packets is active (e.g. the dashed area in figure 1). One may ask why P4 is not also used in the example above. The reason is that we try to rule out parts of the knowledge base that are not relevant for the current problem in order to increase efficiency. The real challenge is, of course, how to determine the appropriate parts of knowledge for the problem under consideration. This has to do with the specification of knowledge packets, and we shall come back to this point in a moment. The model described above matches some of the aims of structuring:

- It allows decomposition of a large knowledge base into separate parts.
- It increases efficiency of knowledge processing by restricting the number of knowledge items that have to be considered at some time.
- It allows management of competing and inconsistent information.

On the other hand there is a severe shortcoming inherent in this approach: the packets are not independent from their environment. They have, for example, no well defined interfaces, so when solving some problem, not only the knowledge of the actual focus, but also the knowledge of all more general modules can be used. Knowledge items in a module may have different meaning depending on which focus for the structured knowledge base is used. While this feature is of advantage for producing context-dependent answers, it makes local verification of single packets impossible, and consistency checking of single packets does not make much sense. Furthermore, separate lower packets on the same level (e.g. P2 and P3 in figure 1) are always seen as being competing; this means that they cannot be visible at the same time, even if they refer to disjoint different aspects of knowledge.

3 Modules for Knowledge Bases

The points of criticism at the end of section 2 arose from a comparison
to modules as they exist in classical computer science (programming languages [11], algebraic specification [4], formal module concepts with imperative implementation part [8]). Recently, an adaption of these ideas to logical knowledge bases has been worked out, see [1] and [2] for details. In this approach, a knowledge base is seen as a collection of independent entities interacting with each other via import and export interfaces, the interfaces being first order specifications.

From a bird’s eye view a module looks as follows.

```
<table>
<thead>
<tr>
<th>EXP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODY</td>
</tr>
<tr>
<td>IMP</td>
</tr>
</tbody>
</table>
```

A module includes an *import interface* describing the knowledge required in order to define the module’s functionality, and an *export interface* describing the knowledge the module is offering to the public. The module’s *body* is the part defining the functionality of the module, i.e. its knowledge.

A modular knowledge base is built by putting modules together. This is done by gluing their interfaces while assuring that the integrity conditions are respected.

The example in figure 2 shows a very raw sketch of the knowledge someone requires when selecting a conference for submitting a paper. It requires knowledge about whether the person’s research interests are included among the conference topics, whether it is financially possible to attend the conference, whether the conference location is interesting, etc. Each unit is a module with interfaces. The module on personal reasons, for instance, could include in its export interface predicates as \( \text{like(Conference)} \), in its import interface predicates as \( \text{conference_location} \) or \( \text{like_country} \), and its body rules as

\[
\text{like(Conf)} :- \text{conference_location(Conf,Loc)}, \\
\text{is_located(Loc,Country)}, \\
\text{like_country(Country)}. 
\]

Results in [1], [2] concern compositionality of semantics and correctness preservation of such systems. The latter means that the entire knowledge base is correct by way of principle if its constituent modules are correct. This means that we can apply verification and validation techniques *in a local setting*. 
Such modular knowledge bases address the points of criticism we mentioned at the end of the previous section, i.e. maintainability, reusability, and decomposition of consistent parts of knowledge into smaller units. Nevertheless, this approach suffers from some serious deficiencies:

- No competing or even inconsistent knowledge items can be managed in one knowledge base in a reasonable way because all knowledge of all lower modules is accessible from the top module at the same time.
- It is also impossible to rule out lower (more special) parts of knowledge even if this knowledge is not relevant for the current problem under consideration.

Note that these points are those solved well in the approach we presented in the previous section. Therefore it is reasonable to combine both approaches in a way that preserves as many of the benefits of them as possible. This is what we shall do in the rest of the paper.

## 4 Combining both approaches: The new model

Let us start with an example that demonstrates our idea of combining the approaches presented above. It describes the knowledge I need to determine my behavior when being downtown.
My behavior when being downtown

- weather
- my mood
- my aim
- my traffic situation

- my health
- last night

- by car
- by bus
- pedestrian

- going to work
- buying
- spending my time

Figure 3

Note that in some connections, the "lower" modules contain knowledge on different aspects (for example the top connection), whereas in others they contain competing knowledge items that exclude each other (for example, module going to work could contain in-hurry, while spending my time could include ¬in-hurry). The basic idea of our model is to distinguish between these kinds of connection.

A structured knowledge base thus consists of a number of modules; these are parts of knowledge closely (semantically) related to each other and defining some specific, self-contained part of the entire knowledge. The modules are equipped with import and export interfaces. The ideal case would be that these interfaces give a full description of the knowledge exported or imported (thus playing a role similar to that of abstract data types in conventional computer science). Unfortunately, such a complete specification of AI systems is usually impossible. In such cases, the export interface describes the signature of exported knowledge (like interfaces in imperative programming languages) and some integrity (consistency) conditions the exported knowledge should satisfy. The presence of formal interfaces and module semantics allows usage of formal verification methods in order to show that a module matches the requirements.

As the example above shows, we distinguish two kinds of module connections:
\textit{AND-connection} states that the modules from which the top one can import do not contain competing knowledge, but rather information on different topics of the modeled domain (Remark: Whereas it is often intuitively clear what competing knowledge means, it is difficult to give a general, formal definition; it is up to the knowledge engineer to decide). This means that the knowledge of all these modules (or of some of them) may be used at the same time.

This kind of module connection is thus inherited from the module concept described in the previous section. Note, though, that according to the new model knowledge of lower modules is not always visible to higher modules, but rather only when needed (according to the current focus; see below). In fact, we could further refine our model by distinguishing between links that indicate that the knowledge of a lower module \textit{must} be visible to higher modules (as in chapter 3) and such that \textit{may} be visible to higher modules (the approach we adopted here). For example, a medical expert system may require knowledge on the patient’s constitution in any case, whereas knowledge on special diseases is only used if it appears necessary in a particular case. We do not further discuss this extension for the sake of simplicity, but it should be clear that this idea can be easily integrated in our approach.

The \textit{OR-connection} of modules indicates that the modules (on the lower level) involved contain competing knowledge. In this case, only one of these modules may be visible at a given time, similar to the model described in section 2. But note that any such module may be AND-connected to other modules in a subsequent level. Finally, let us note that a module needed in distinct OR-connected knowledge parts, may be shared.

With both kinds of module linkage we must take their interfaces into consideration. It should be clear that the interfaces must fit to each other, at least in a syntactic way. If, additionally, conditions are included in the interfaces, then these must also be respected. This means that the exported knowledge of a lower module must respect the import knowledge of a higher one, i.e.
the export conditions of a lower module must logically imply the import conditions of the higher module.

Let for example M1 be a module that creates a lecture timetable in a university. Among others, M1 imports knowledge about lecturers and students. Its import interface contains thus predicates \textit{lecturer}(X) and \textit{student}(X) together with the integrity condition \(\neg \exists X(\textit{lecturer}(X) \land \textit{student}(X))\). Let M2 be a module containing such information and having in its export interface predicates \textit{lecturer}(X), \textit{student}(X) and \textit{employed}(X). If the export interface contains no integrity conditions, then connection to M1 is impossible, since M1’s import integrity conditions would not (necessarily) be respected. If, on the other side, the export interface contains conditions

\[
\forall X(\textit{student}(X) \rightarrow \neg \textit{employed}(X)) \\
\forall X(\textit{lecturer}(X) \rightarrow \textit{employed}(X))
\]

then connection is possible. Note that in this case all knowledge about predicate \textit{employed} will not be passed to module M1, as this predicate does not appear in its import interface.

As in section 2, the meaning of a structured, modular knowledge base is defined with respect to a \textit{current focus}. This focus defines a current view on the knowledge base and must be such that competing parts of knowledge are not visible at the same time. In the example of figure 3, a focus could consist of the modules \textit{by car}, \textit{going to work}, and \textit{last night}. Then, these modules and all modules above them are visible at the moment, i.e. their knowledge may be used. Note that for each OR-connection, at most one module can be included in a current focus. For each AND-connection, none, one, some, or all involved modules (of the lower level) may be included. In our example here, we have not included \textit{my health} to the focus. It could be the case that I have slept bad tonight, so even my good health cannot prevent my mood from being bad.

Apparently, it is not always sufficient to have only one module as a focus, but rather a set of modules. The definition of possible focuses is given inductively as follows:

- \{M\} is a possible focus, where M is the top module in the hierarchy
- If F is a current focus, M’ is an element of F, and M’ is AND-connected to M\textsubscript{1},...,M\textsubscript{n}, then replace M’ in F by an arbitrary subset of \{M\textsubscript{1},...,M\textsubscript{n}\}. The resulting set is a admissible focus.
- If F is a current focus, M’ is in F, and M’ is OR-connected to M\textsubscript{1},...,M\textsubscript{n}, then replace M’ in F by some M\textsubscript{i} from M\textsubscript{1},...,M\textsubscript{n}. The resulting set is a admissible focus.
All modules that are above some element of the current focus are visible. Note that the definition of admissible focus is such that competing knowledge cannot be visible at the same time. Furthermore, it is easily verified that the focus example above respects this definition.

Obviously, it is unreasonable to demand global consistency of a structured, modular knowledge base. Instead, only knowledge items that can be active at some time need to be consistent to each other. Following [9], we call this the local consistency requirement.

5 Formal description of the model

In this section we briefly introduce the semantics of structured knowledge bases in the setting of logic programming [6]. The terminology follows [8].

The body of a module M is a logic program (possibly with negation) kb(M). The interfaces exp(M) and imp(M) of M contain the predicates that are imported resp. exported. As usual in logic programming, we regard the constants and function symbols as being global (this restriction is made in order to keep the discussion here simple, but has nothing to do with the general model from section 4). As our idea is that knowledge about some predicates are imported from other modules, we protect imported knowledge by demanding that kb(M) is conservative w.r.t. imported predicates (meaning that imported predicates do not occur in heads of rules in kb(M)); see [1] for more details.

The meaning of a module, stand alone, is determined by the facts p(t₁,…,tₙ) with an exported predicate p that follow from the completion of kb(M) and the imported knowledge (if there is any; this will depend on the current focus as we shall see in a moment).

Combination of modules (either by OR- or by AND-connection) is done in such a way that predicates exported by the lower modules can be imported by the top module of the connection if these predicates also appear in its import interface. Of course, more flexible ways of combination are possible, for example signature morphisms [3] allowing renaming of predicates and non-injective mappings. We disregard this possibility here for the sake of simplicity. As in section 2, the semantics of a structured knowledge base S is given relative to a current focus F. If S is a single module, then its semantics has already been described as comp(kb(M))∩exp(M), where comp is the logical completion operation. Let M be the top module of S, and suppose that it is related to the modules involved underneath by an AND-connection.
S1,...,Sn are the structured knowledge bases with top modules M1,...,Mn such that M is AND-connected to M1,...,Mn. Let S1,..,Sk be the structured knowledge bases containing a member of the current focus. Then define Export(S), the exported knowledge of S (always w.r.t. focus F) as follows:

\[ \text{comp(Import(M,F)∪kb(M)))] \cap \text{exp(M)}, \]

where Import(M,F) is \([\text{Export(S1)}∪...∪\text{Export(Sk)}] \cap \text{Imp(M)}.\]

In particular, if M∈F, then the exported knowledge of S is comp(kb(M))∩exp(M).

In case of an OR-connection at top level, the definition of Export(S) is as above, the only difference being that (by definition of possible foci) only one subsystem from S1,...,Sn can include members of F.

6 Conclusion and future work

We have introduced a new model for structuring knowledge bases and presented some examples showing that it can be useful in praxis. The model combines advantages of two previous approaches and addresses the main requests associated with modularity (restriction of search space, maintainability and reusability) as well as the additional requirement of managing competing knowledge within one knowledge base. It seems to be compatible with experimental findings on human intelligence, while also addressing engineering problems.

There remains much to be done, though. One problem that has not been solved yet is that of specification. In the present paper we have restricted attention to syntactical information plus some integrity conditions describing desirable, or even crucial, properties. The question arises whether we are able to completely specify a knowledge unit as usually done for programs in computer science. This task will be especially difficult in case of nonmonotonic knowledge bases. However, even if it is utopic to expect complete specifications of knowledge, it should be clear that the more aspects of knowledge we are able
to specify, the more formal methods can be applied, leading to a higher quality of
knowledge base.

Another problem we have completely left out in the present paper is
that of determining the appropriate focus, i.e. the parts of knowledge relevant
for the current problem. We think that this question lies at the heart of intelli-
gent behavior. Until now there are only some practical solutions for special
cases (for example keyword-based access in text understanding problems),
but no generally applicable theory. In the context of a government-funded
three-year research effort just begun at the University of Bielefeld, a modular
medical knowledge base for hypertension consultation will be developed.
Based on the experiences gained in this practical work we will further pursue
the point of focus management.

Literature

IEEE Press 1992
AIMSA-92
Springer 1985
Springer 1990
Lecture Notes in Artificial Intelligence 546, Springer 1991
[7] Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, P. and Swar-
tout, W.R.: Enabling Technology for Knowledge Sharing, AI Magazine 12.3, 36-
56
ence, Addison-Wesley 1991
künstlichen Systemen, IWBS-Report 91, IBM Deutschland, Stuttgart/Heidelberg 1989
[10] Wachsmuth, I. and Gängler, B.: Knowledge Packets and Knowledge Pack-
et Structures, in [5]