Integrating Problem-Solving Methods into Cyc

James Stuart Aitken
Artificial Intelligence Applications Institute
Division of Informatics
University of Edinburgh
Edinburgh EH1 1HN, Scotland
Email: `stuart@aiai.ed.ac.uk`

Dimitrios Sklavakis
School of Artificial Intelligence
Division of Informatics
University of Edinburgh
Edinburgh EH1 1HN, Scotland
Email: `dimitris@dai.ed.ac.uk`

Abstract

This paper argues that the reuse of domain knowledge must be complemented by the reuse of problem-solving methods. Problem-solving methods (PSMs) provide a means to structure search, and can provide tractable solutions to reasoning with a very large knowledge base. We show that PSMs can be used in a way which complements large-scale representation techniques, and optimisations such as those for taxonomic reasoning found in Cyc. Our approach illustrates the advantages of task-oriented knowledge modelling and we demonstrate that the resulting ontologies have both task-dependent and task-independent elements. Further, we show how the task ontology can be organised into conceptual levels to reflect knowledge typing principles.

1 Introduction

Developing reusable ontologies which specify the structure and content of domain knowledge has become a central problem in the construction of large and scalable knowledge based systems. For example, a key step in KBS construction using the Cyc system [Lenat and Guha, 1990] is to extend the existing upper-level ontology by creating new classes and representations. Methodologies for ontology development have been proposed [Lenat and Guha, 1990; Uschold and Gruninger, 1996; Blázquez *et al.*, 1998]. However, many unsolved problems remain. Other important issues concern the relationship between the domain representation and its intended use [Wielinga *et al.*, 1994; van Heijst *et al.*, 1997]. We shall concentrate on the representational and performance issues focusing initially on the reasoning processes, and reflect on the implications for domain representation in the light of these findings.

Versions of Cyc are currently being used as an integration platform by the DARPA-funded High Performance Knowledge Bases (HPKB) program. Key issues on the HPKB program are the **scalability**, **robustness**, and **reusability** of knowledge-based system solutions. Cyc is unique in that it has potential solutions to each of these problems.

Cyc uses a resolution-based inference procedure that has a number of optimisations that improve the scalability of the architecture. For example, a specialised taxonomic reasoning module replaces the application of the logical rule for transitivity of class membership. Where specialised modules are not implemented, Cyc makes use of weak search methods to perform inference. Cyc lacks any principles for structuring inference at a conceptual level. Problem-solving methods provide precisely this structure, hence the importance of integrating structuring principles into a scalable KBS architecture.

Robustness and reusability are related properties of the knowledge representation scheme and the inference rules: Predicates such as *bordersOn* and *between*, defined in the upper-level ontology, can be reused in many different contexts. The combination of predicate properties (such as symmetry) and existing inference rules means that the use of these predicates is robust. Reconciling units of measure is a similar problem. In this case, Cyc has sufficient knowledge to prove *(greaterThan (Meter 1) (Centimeter 2))* using its existing definitions and rules about units of measure. Reusability is also an important motivation for defining a upper-level ontology as the basis of knowledge representation. The upper-level ontology can be shared among more specialised reasoning contexts or applications. Extensions to the upper-level can themselves be shared, and can be regarded as ontologies in their own right.

We describe the implementation of a PSM for fault diagnosis in Cyc. The diagnostic method was applied to two different domains to investigate whether the potential for method reuse was actually achievable. As implementation was preceded by a significant amount of domain and task analysis, this work allows us to review the value of the methodological approach and to investigate issues such as the task-dependence of the ontologies constructed. This paper begins with an introduction to the component technologies used—CommonKADS and Cyc—and then describes the implementation of the PSM and the associated knowledge modelling.

2 Component Technologies

2.1 PSMs: The CommonKADS View

In CommonKADS, problem-solving methods are the product of expertise analysis - one of several analysis steps which are specified by the methodology. PSMs are also used in Protege [Puerta *et al.*, 1992], and in Expect [Gil and Melz, 1996] (although in different forms). PSMs define distinct methods for performing a task, for example, diagnosis can be modelled as involving a heuristic association between observations and solutions, or as a process of decomposing a system and testing its subcomponents for correct operation. In addition to specifying an inference procedure, PSMs require that domain knowledge be modelled in particular ways, i.e. a method ontology is associated with a PSM.

CommonKADS is a methodology for KBS development which addresses not only the desired problem-solving performance of the end system, but the context in which it will operate. A number of models are constructed in the analysis phase: an *organisational model* represents the processes, structure, and resources of the organisation which is to use the KBS, a *task model* describes the activities of the process of interest, an *agent model* represents the agents involved in the process and their capabilities, a *communication model* describes agent (human and machine) communication, an *expertise model* defines domain and problem-solving knowledge, and, finally, a *design model* describes the structure and function of the system that will implement the knowledge-based task. More details of the various models, and appropriate modelling techniques can be found in [Kingston *et al.*, 1997].

CommonKADS is relatively neutral on questions of implementation. However, expertise modelling does make a number of assumptions about knowledge representation constructs and their interaction. The expertise model has three layers: the domain layer represents knowledge about the domain, the inference layer defines the procedures applied during problem solving, and the task layer specifies the ordering of inference steps. As the expertise model is the only CommonKADS model that captures expert problem-solving behaviour, we shall limit our attention to representing this model in Cyc.

2.2 Cyc

Cyc is a very large, multi-contextual knowledge-based system which is currently being used commercially by **Cycorp**. Cyc is also used for research purposes, and, in the HPKB program, Cyc is being used as a platform for technology integration.

The arguments for Cyc proposed in Lenat and Guha [1990] remain the cornerstones of the Cyc project; namely, the need to overcome the brittleness of traditional expert systems, and the means of achieving this through the development of a shared ontology representing 'consensus reality'. The upper-level ontology, which constitutes the basis of knowledge representation in Cyc, has been made publicly available. However, this represents only a fraction of the knowledge which has been entered into Cyc.

The upper-level ontology is represented in a variant of first-order logic known as *CycL*. The ontology includes: classes used for constructing representations, for example *SetOrCollection* and *Predicate*; classes for high-level concepts such as *Event* and *Agent*; and more specific classes representing commonly occurring objects and events such as *Book* and *BirthEvent*.

Assertions in CycL are always associated with a *microtheory* context. The *BaseKB* contains the upper-level ontology and new contexts can be defined which specialise this theory. Multiple inheritance of microtheory contexts is allowed. Alternative specialisations of a microtheory need not be consistent with each other: a microtheory can contain ontology extensions and assertions which are inconsistent with those defined in a different theory - providing neither context is defined as subsuming the other. The microtheory mechanism plays an important role in structuring inference.

Cyc performs inferencing in response to a query by the user (by backward chaining) or in response to an assertion (by forward chaining with rules which are explicitly specified to be forward rules). Queries are made in a specific microtheory which forms the local search context. Typically, a microtheory will be a specialisation of one or more theories and in this case search will progress out to wider contexts should a solution not be found locally. Queries are treated in a purely logical manner: the order of conjuncts is not considered to be significant and may be changed by optimisations operating at the clause-form level. The preconditions of rules are also treated in this way - prohibiting the user from influencing the search process in a predictable way. The dependencies between derived facts, rules and assertions are recorded and maintained by a truth maintenance mechanism.

Cyc's purely declarative treatment of rules differs from other approaches to logic-based knowledge representation, such as Prolog, where the ordering of clauses, and of literals within clauses, is used to determine the order of search.

The Cyc system includes a number of tools for viewing and browsing the ontology. In common with other browsers, including that for Loom [MacGregor, 1994], terms in the ontology are hyperlinked in a web-based interface. This allows the user to explore the concepts which define, or are subsidiary to, the concepts currently being displayed.

The Cyc system also gives the KBS developer access to a LISP-like environment where new procedures can be defined in the *SubL* language. The Cyc knowledge base and inference engine can be accessed via the SubL functions *ask* and *assert*. Due to the treatment of rules described above, imposing structure on the search process necessarily requires SubL coding.

3 Systematic Diagnosis in Cyc

This section describes the expertise modelling process and presents its products. The implementation of these models in Cyc is then outlined. We begin with a brief introduction to the domain and the diagnostic task.

The task of diagnosis was selected because a set of well understood methods for solving such tasks already exists [Wielinga *et al.*, 1992]. An important part of expertise modelling is the selection between alternative methods - with their accompanying behaviours and assumptions. The choice of a specific diagnostic method was not made prior to domain analysis. It is readily apparent that we have chosen a problem type that falls within the scope of the methodology we intend to apply. However, it is not at all obvious that diagnosis—which is inherently an incremental procedure requiring information gathering—can be adequately implemented by backward chaining driven by a query-based interaction (i.e. by the default environment provided by Cyc). We shall return to this point below.

Fault finding in personal computers (PCs) was chosen as the primary task domain. This task can be modelled accurately, i.e. the actual behaviour of human experts is known and has been documented [Kozierok, 1998], yet the amount of electronics knowledge required is low as fault finding never progresses to a level where sophisticated test equipment is required. The second domain chosen was fault finding in an automobile ignition system. This task ought to be soluble by the method developed for PC diagnosis, despite differences in the characteristics of the domain and in the method ontology.

3.1 Modelling Expertise

The selection of a problem-solving method is one of the central modelling decisions in CommonKADS. This will typically have an impact on domain representation. Following this approach, the PC-diagnosis problem was addressed by investigating candidate PSMs. As PSMs may be refined in several different ways, alternative instantiations were also investigated. This is a notable contrast with a domain-oriented approach which would focus on developing an ontology of the domain being reasoned about, PC systems and their components in this case.

The systematic diagnosis PSM was found to match the expert reasoning process most closely. The generic model had to be adapted to reflect expert reasoning more faithfully. The central steps in systematic diagnosis are the decomposition of the system being diagnosed into subsystems, and the testing of the subsystems for correct operation by making tests and comparing the observed with the predicted outcomes. The subsystem currently being tested is said to play the role of the current *hypothesis*. Testing may rule out this hypothesis, in which case another subsystem becomes the hypothesis. Testing may yield an inconclusive result, in which case more tests are required, or testing may indicate a fault, in which case the diagnosis is concluded—if the current hypothesis cannot be further decomposed (i.e. it is a component), or diagnosis continues at a lower level of

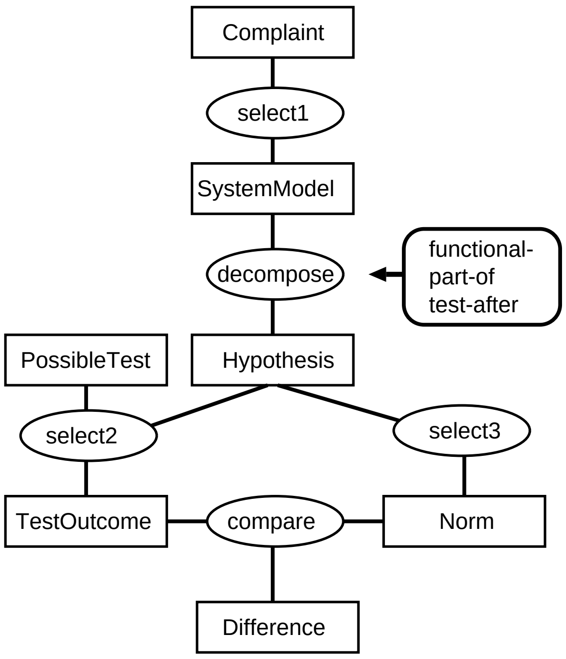

Figure 1: PSM for systematic diagnosis

system decomposition—if the current hypothesis can be decomposed (i.e. it is a system). The system model may describe how the system is decomposed into (physical) parts, or may describe the functional relationships between systems.

It was discovered that the the *part-of* model, which lies at the heart of systematic diagnosis, had to be instantiated to *functional-part-of* in the PC diagnosis domain. That is, problem solving requires a functional view of the system, rather than a component/subcomponent view. The *functional-part-of* predicate is clearly a representational construct at the domain level, and is one of several part-of views that might be taken of a system. In fact, there was no need to represent the *physical-part-of* relation in order to solve this problem.

Another important refinement of the generic model was the addition of theories of test ordering. Where there are several decompositions of a system, the model permits any subsystem to play the role of hypothesis. However, in PC diagnosis it is important to establish first, for example, that the power system is operational, then that the video system is operational. Once the video system is known to work we can be sure that the results of BIOS system tests are being displayed correctly. Similar ordering constraints were found for all subsystems, and at all levels of decomposition. Consequently, there is a need to impose an order on hypothesis selection (or, equivalently, system decomposition) and we chose to represent this knowledge in a heuristic fashion via a *testAfter* predicate. Figure 1 shows the specialised PSM in a diagrammatic form.

Determining the overall view of the desired problem-solving behaviour aided knowledge acquisition, much of

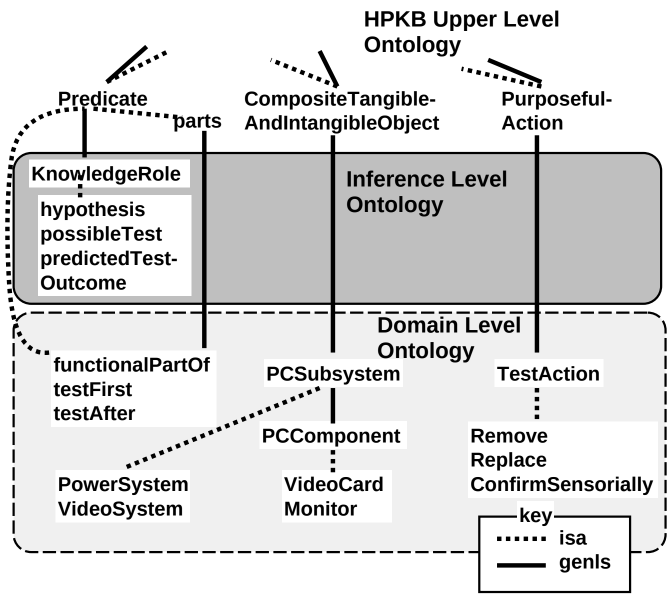

Figure 2: Upper-Level ontology extensions - distinguished by level

which concerned the extraction and structuring of information from an on-line source [Kozierok, 1998]. Our experience confirmed the claimed advantages of the modelling approach. In addition to specifying an inference-level procedure, knowledge acquisition also requires the content and scope of domain knowledge to be determined. The task of representing domain knowledge in Cyc followed the standard procedure of extending the ontology by defining new collections and predicates, and linking these to existing constants. We now describe the Cyc implementation in more detail.

3.2 Cyc Implementation

Diagnosis requires interactive data gathering, and the subsequent evaluation of test results and updating of the current hypothesis. Such a procedure cannot be implemented by logical inference alone, and so it is clearly necessary to use Cyc's LISP-like language, SubL, to implement a control regime. In CommonKADS, control knowledge is divided between the inference layer, where knowledge roles and inference steps are defined, and the task layer, where the order of application of inference steps is specified. Our aim was to represent the levels of the expertise model in Cyc in as faithful a manner as possible. We begin by considering domain knowledge.

Domain knowledge was represented by extending existing collections where possible. Figure 2 shows a small illustrative set of the extensions made. The collection PCSubsystem was added as a subcollection of CompositeTangibleAndIntangibleThing, and PCComponent was defined as a specialisation of it. Both types of object have a tangible component, and may carry information hence have an intangible component also. TestAction was defined as a new collection of PurposefulAction, and the instances of Remove, Replace, and ConfirmSensorially (i.e. confirm by observation) were added [Sklavakis, 1998]. functionalPartOf was introduced to represent the functional decomposition of a system, and stated to generalise to parts, being the most general existing *part-of* predicate in the upper ontology[1]. Other specialisations of parts include physicalDecompositions and timeSlices.

The predicates testFirst and testAfter were introduced as predicates to represent the test ordering theory. A test is defined by three components: a TestAction, a PCSubsystem and a PossibleObservable. The collections PossibleObservable, PossibleObservableValue, and ResultType were defined as subcollections of AttributeValue. The representation of testing knowledge can be made more robust by grounding it extensively in the upper ontology. In contrast, part-of facts are not likely to be derivable by appeal to background knowledge.

At the inference level, knowledge roles are represented by predicates, and inference steps are rules which have knowledge roles as preconditions and conclusions. Figure 2 shows the introduction of the KnowledgeRole collection, a specialisation of the Predicate class of the upper ontology. Instances of KnowledgeRole predicates take domain-level formulae or collections as arguments. Examples include; the unary predicate hypothesis - applicable to PCSubsystem - denotes the current hypothesis, possibleTest holds of applicable tests, and the relation predictedTestOutcome holds of a test, PossibleObservableValue and a ResultType. More complex mappings to the inference level, and the definition of additional collections and terms, are also possible within this approach. The CycL language is sufficiently expressive to allow complex mappings of the type described in [Wielinga *et al.*, 1994] where the inference-level ontology (in our terminology) might define relations holding of domain-level ontology, e.g. we could express the fact that PhysicalPartOf is a relation: *relation(PhysicalPartOf)*.

In a similar way, the currently invoked inference step (e.g. *decompose*, *select*) is also explicitly asserted in the KB by predicates which belong to the inference level.

Inference steps are invoked by querying or asserting knowledge roles. For example, the role *hypothesis* holds of the subsystem currently playing the role of the hypothesised fault. The rules for selecting the test ordering theory depend on the current hypothesis, for example:

```
F: (implies (and (hypothesis PCSystem)
                 (plausibleInference Decompose))
        (and (testFirst PowerSystem)
             (testAfter PowerSystem VideoSystem))).
```

This is a forward rule which fires when *hypothesis* and *plausibleInference* are asserted. The current hypothesis assertion must be deleted and replaced as diagnosis proceeds. These operations are implemented in SubL by

[1]Note that terms defined in the ontology, or its extension, are written in sans-serif, following the Cyc convention, names of collections begin with a capital letter and predicates begin with a lower case letter.

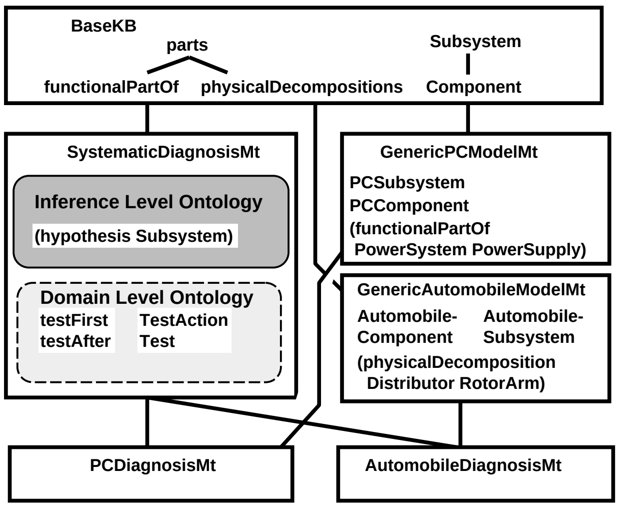

Figure 3: Microtheory structure

functional-interface functions, within the larger structure of the systematic diagnosis task function. The user could make this series of deductions themselves, and in the implemented system, the user is able to inspect the state of the reasoning process as it progresses. As an example, the following SubL code is called at the start of diagnosis and simply asserts that the entire system is the hypothesis, and then calls another SubL function, `sd-select2-3`, which performs system decomposition.

```
(define sd-select1 (system)
 (fi-assert (#$hypothesis system) *defaultMt*)
 (sd-select2-3))
```

We have achieved an explicit representation of knowledge roles and of inference steps in Cyc that reflects the knowledge typing principles advocated in [van Heijst *et al.*, 1997]. Control over the search process is achieved by making a series of simple queries, structured to implement the pattern of inference of the PSM. We found no need to extend the functionality of Cyc, or the expressivity of its representation language(s) in order to implement the PSM. The central problem was to combine the available features into structured architecture, in order to to take advantage of the model-based approach to problem solving.

We tested the reusability of the domain and inference level definitions, and of the SubL code, by considering diagnosis in the domain of automobile ignition systems. This experience is discussed in the wider context of the reusability, scalability, and robustness of our approach.

4 Representation and Reasoning

4.1 Domain Ontologies

The view of domain ontology construction which results from the prior selection and adaption of an explicit problem solving method is more focussed on concepts relevant to problem solving than a task-neutral view would be. The resulting domain ontology is not task-specific in its formalisation, e.g. the definition of the *functional-part-of* relation has no intrinsic task-related properties. But, the coverage of the resulting ontology may only be partial – we did not need to elicit *physical-part-of* knowledge.

Had we taken a view that focussed on the domain alone, we would have had no explicit guidance as to which concepts were or were not relevant to the ontology definition effort. We have gained experience of constructing ontologies where the primary aim was to represent the domain, with ontology definition only informally guided by considering the task. Under these conditions it is difficult to determine the relevance of a potential domain concept, and the distinction between concepts that are intrinsic to the representation of the domain, and those that are related to the task to be performed was difficult to make.

Reusability of domain knowledge is an important issue, and our approach has been to use the microtheory mechanism of Cyc to encapsulate the generic components of the extended ontology. The resulting mircotheory structure, shown in Figure 3, places the generic system models for PCs and automobile systems in distinct microtheories, that are extensions of the BaseKB, and are included in the specific diagnosis microtheories. Strictly speaking, these microtheories are not extensions of the ontology as they make no new specifications. Instead, the BaseKB is extended by adding the definitions of the functionalPartOf predicate and the collections Subsystem and Component as these concepts are sufficiently general to be reusable across domains. The method-specific ontology, comprising domain and inference level components, is also a specialisation of the BaseKB, and this theory is shared by both PC and Automobile diagnosis theories. The microtheory structure shows that the generic system models can be used in any context which includes the (now extended) BaseKB, and that these theories can be thought of as parameters of the diagnosis microtheories.

4.2 Inference Knowledge

The application of systematic diagnosis to the automobile domain required a change in system theory from *functionalPartOf* to *physicalDecomposition*. While this is a significant change in the modelling of the diagnostic process (physical parts play the role of hypotheses) there were few implications for formalisation of the inference level as no new knowledge roles were found. Similarly, the SubL code was only modified to take the specific diagnosis microtheory as a parameter. In future, we aim implement other PSMs and this may permit us to generalise inference-level theories across PSMs.

4.3 Scalability

The domain and inference level knowledge representations that we have used are extensions of the basic representation, and can make use of the existing optimisations for indexing large KBs, performing taxonomic reasoning and theory structuring. Our approach to PSM implementation is based on structuring a series of queries and assertions to implement a problem-solving method. As the individual queries are simple, the space searched is small (we can specify the depth of search to be 1–3 levels). This contrasts with the basic query mechanism where the only means of getting an answer to query which requires many rules to be combined is to increase the depth of search - with the resulting exponential increase in search space.

4.4 Robustness

At present we are unable to reason about inference structures or about the mappings from the domain to the inference level within Cyc. There are no rules which allow PSMs to be modified or to be configured. Consequently, the system lacks robustness as it cannot fall back to first principles when an existing method is not immediately applicable. The problems of PSM modification and configuration are significant, even for human experts, but we believe that automatically specialising PSMs is a feasible proposition. We also plan to explore the idea of falling back to more general methods, when more specific methods are inapplicable, to regain robustness.

Inference steps (implemented by rules) require proving domain-level predicates, and robustness at the level of reasoning about domain knowledge occurs exactly as in Cyc.

5 Discussion

We have described an approach to implementing problem-solving methods in Cyc which makes use of the existing optimisations developed for large-scale knowledge bases, and adds additional structure to the inference process. Extensions to the existing ontology distinguished generic extensions to the upper-level ontology, extensions to the knowledge base, and task-related extensions. Knowledge typing principles were used within the task-related ontology to further structure problem-solving knowledge.

Our investigation of diagnostic problem solving has not only raised issues of knowledge reuse and scalability, but also of system-environment interaction. Intelligent systems cannot rely on large amounts of background knowledge alone as many classes of problems require information gathering or user interaction. If such interaction is to happen in an intelligent fashion then there is a requirement to represent and reason about the inferences which require interaction.

Acknowledgments

This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) under grant number F30602-97-1-0203. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, Rome Laboratory or the U.S. Government.

References

[Blázquez *et al.*, 1998] M. Blázquez, M. Fernández, J.M. Garcia-Pinar, and A. Gómez-Pérez. Building ontologies at the knowledge level using the ontology design environment. In *Proceedings of KAW'98*, 1998.

[Gil and Melz, 1996] Y. Gil and E. Melz. Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[Kingston *et al.*, 1997] J. Kingston, A. Griffith, and T. Lydiard. Multi-perspective modelling of the air campaign planning process. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 668–673, 1997.

[Kozierok, 1998] C.M. Kozierok. Troubleshooting expert, 1998. URL: `http://www.pcguide.com`.

[Lenat and Guha, 1990] D.B. Lenat and R.V. Guha. *Building large knowledge-based systems. Representation and inference in the Cyc project.* Addison-Wesley, 1990.

[MacGregor, 1994] R.M. MacGregor. A description classifier for the predicate calculus. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 213–220, 1994.

[Puerta *et al.*, 1992] A.R. Puerta, J.W. Egar, S.W. Tu, and M.A. Musen. A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition*, 4:171–196, 1992.

[Sklavakis, 1998] D. Sklavakis. Implementing problem solving methods in Cyc. Master's thesis, Division of Informatics, University of Edinburgh, 1998.

[Uschold and Gruninger, 1996] M. Uschold and M. Gruninger. Ontologies: principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–136, 1996.

[van Heijst *et al.*, 1997] G. van Heijst, A.T. Schreiber, and B.J. Wielinga. Explicit representations of problem-solving strategies to support knowledge acquisition. *International Journal of Human-Computer Studies*, 46(2/3):183–292, 1997.

[Wielinga *et al.*, 1992] B.J. Wielinga, T. Schreiber, and J.A. Breuker. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 11(2):93–136, 1992.

[Wielinga *et al.*, 1994] B.J. Wielinga, G. Schreiber, W. Jansweijer, A. Anjewierden, and F. van Harmelen. Framework and formalism for expressing ontologies, 1994. KACTUS Project Report (Esprit Project 8145) DO1b.1-Framework-1.1-UvA-BW+GS+WJ+AA, University Of Amsterdam.