Knowledge Representation and Inference Two

Lecture Nine:

AI-Based Design:

The Edinburgh Designer System

and

AI Toolkits: An Overview

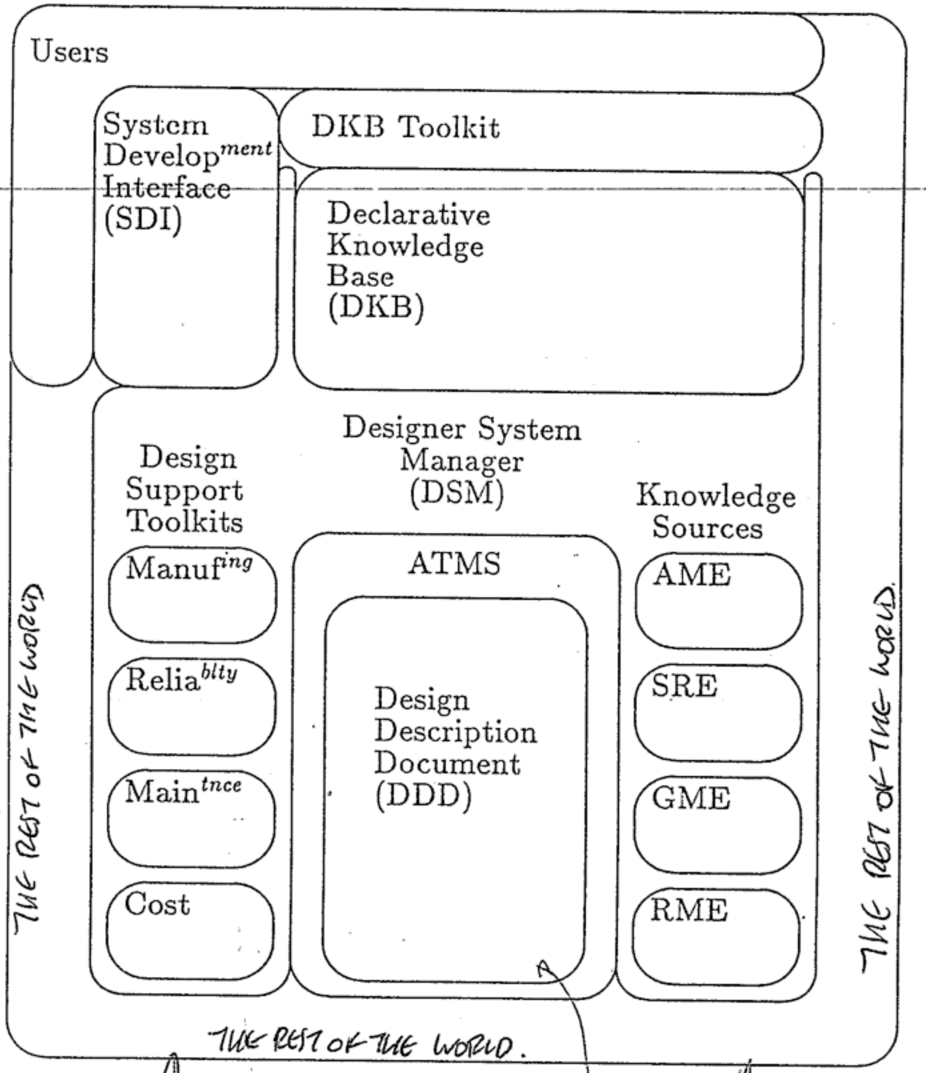
1 Introduction

These notes are divided into two parts. Part one describes the Edinburgh Designer System (EDS), and part two is a copy of the slides Dr Paul Chung (AIAI) will use in presenting his overview of AI Toolkits.

2 The Edinburgh Designer System

The Edinburgh Designer System (EDS) is an AI-Based Design Support System being built in the Department of Artificial Intelligence at Edinburgh University as part of the Alvey Large Scale Demonstrator Project "Design to Product" [Smithers 85]. It is primarily being constructed to support mechanical engineering design, and the sub-systems within it reflect this.

The aim of the EDS is to provide a computer based system which, by the integrated application of a number of Artificial Intelligence techniques, will support engineers carrying out the processes involved in designing a product, and thus to aid the engineers in arriving at a consistent and suitable design specification, not only of its functional and geometric aspects, but also of its maintenance, reliability, cost, and manufacture etc. The EDS differs from present day CAD systems in two major respects: first it sets out to support a wider range of the activities carried out during the design process, not just drawing; and the representation of designs, built using it, is founded on a representation of function, rather than geometry, as is the case in conventional CAD systems. The function based representation can be further enhanced by the consistent addition of other types of relevant knowledge, such as geometry, spatial relationships, manufacturing knowledge etc., and is therefore best seen as a rich representation primarily centred on functionality [Smither 87a] [Smithers 87b].



TASK KNOWLEDGE BASE.

Figure 1 - The Edinburgh Designer System Architecture

Special Purpose Design support systems

General Purpose Design Support Systems.

The reason for developing a function based representation formalism, rather than follow the CAD systems geometry based representation approach, is so that the EDS can reason about the effects upon functionality when changes are made to a proposed design either of other aspects of its functionality, or its geometry, manufacture, maintenance, reliability, or cost for example. During the design of a product all these aspects will need to be taken into account, and will often impose conflicting constraints upon the overall design. However, no matter what weighting might be given to these aspects of a product's design, its required functionality must at all times be maintained - there is no point designing a product which is very easy to manufacture, or maintain, for example if it does not function in the manner required. It is therefore important that when considering all these possibly conflicting aspects of a product's design any changes which remove or change the functionality of the product being designed can be identified to the designers. This is, therefore, one of the primary aims of the EDS as a design support system, and is why a function based representation formalism is being developed for it.

To provide, in an integrated way, the type of support functions required by engineering designers, the different AI technologies being employed in the EDS have to be interfaced to each other in a uniform and coherent manner. The current EDS architecture contains what is considered to be a core set of sub-systems necessary to support engineering design. The architecture and its implementation is intended to form an open architecture system. In other words the EDS is to be seen as the core of a system which can be flexibly added to and/or reconfigured to meet different requirements. One of the major goals of the EDS research programme is to gain an understanding of how previously separately developed AI systems can be put together to form an integrated and uniform system which behaves in a controlled and comprehensible way.

The major components of the EDS which are integrated in a system architecture are described in more detail in the following sections. Each section relates to one of the labelled parts of the system architecture diagram presented in figure 1.

2.1 The Declarative Knowledge Base (DKB)

The way in which the EDS supports the process of engineering design is by providing a knowledge base containing knowledge from which engineers are able to construct a description of their design, or designs. For a computer based system, such as the EDS, to maintain such a description of an engineering design, or set of designs, the description has to be a formal representation. The term formal here means mathematically based. This need for a formal representation of the design description in turn means that any element, or part, of that design description must be uniquely and unambiguously identifiable. In other words every element of the representation of the design description must represent one thing and one thing only. If elements of the representation were to represent more than one thing the EDS would have to have the ability to resolve the ambiguities which would consequently arise in the design description - this is something that AI research has shown is very hard for a computer based system to do, it typically requires contextual knowledge which is often not present in the representation.

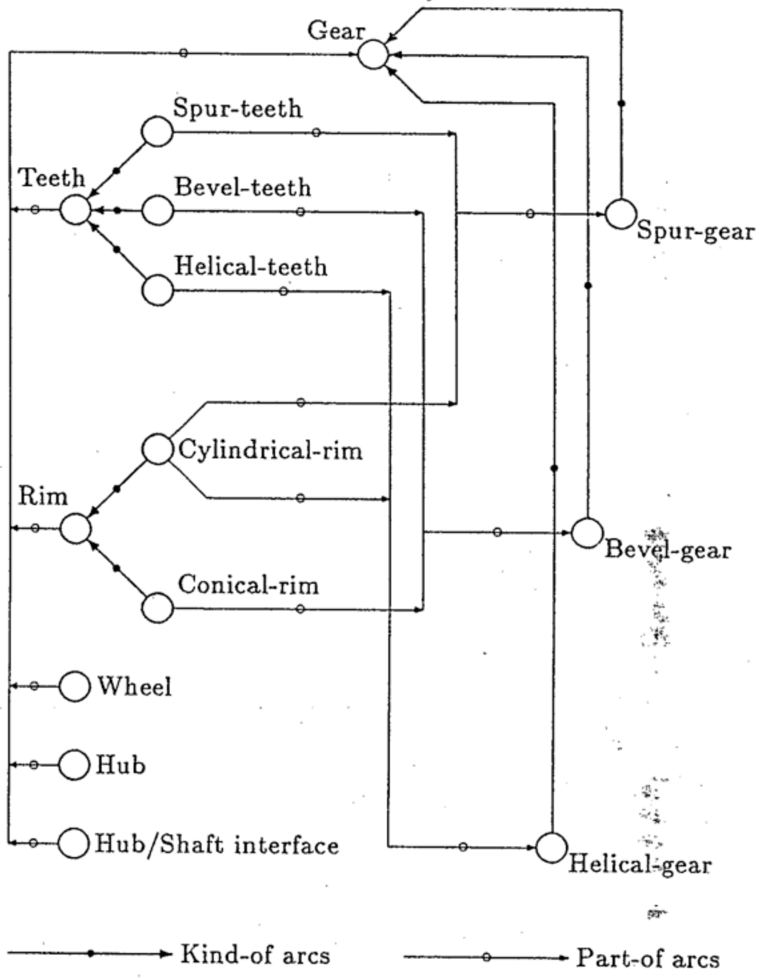


Figure 2 - An example of part of the Functional Unit Module Class Structure of the DKB for kinds of gears and their element parts.

The Declarative Knowledge Base is thus the base of engineering knowledge provided by the system from which suitable knowledge is taken to construct a description of a design. The knowledge it contains is organised in a hierarchical way in which different levels of the hierarchy represent knowledge at different levels of abstraction. This hierarchical structuring is achieved by packaging up knowledge, appropriate to a particular level of abstraction, into Functional Unit Module Classes; the term module being borrowed from the work of Barrow [Barrow 83] on digital hardware design verification. A Functional Unit Module Class presents, in a declarative form, knowledge about a single engineering functional entity which has a concrete referent, at a particular level of abstraction. The declarative representation of knowledge is to be distinguished from the procedural, or rule based, representation of knowledge by the fact that the representation either has nothing to say about how the knowledge is to be applied, or it in no way implies how it is to be used. Within each module knowledge pertaining to other aspects of the class are also represented; concerning geometric shape, spatial relationships between parts of the module, and manufacturing knowledge, for example.

A Functional Unit Module Class within the DKB can be related to other module classes in two ways: by a "Kind-of" relationship which places it below the module class of which it is a kind-of in a hierarchical structure, or by a "Part-of" relationship which relates it to other module classes whose knowledge it makes use of by including them within itself as parts. In order to maintain the necessary unit, or single, functionality representation a module class may only be a Kind-of one other module class. In other words a module class may not be a kind-of two different types of module classes higher up in a hierarchical structure.

The Functional Unit Module Classes contained within the DKB can thus be thought of as forming an entity-relationship directed graph structure whose entities are Functional Unit Module Classes, and whose arcs are either Kind-of relationships or Part-of relationships. An example of what part of a DKB concerned with kinds of gears and their element parts is presented in figure 2. It should be noted that this example represents only a very small part of what a useful DKB would need to contain.

2.2 Declarative Knowledge Base Toolkit (DKBTk)

The building and maintaining of a knowledge base, just like a data base, requires the provision of suitable tools. The DKBTk contains a set of tools for building the DKB from a set of Functional Unit Module Class definitions, the production of Module Class definitions, and tools for checking the syntax of module class definitions.

2.3 The Design Description Document (DDD)

The term Design Description Document is used to refer to the body of knowledge about a product which is built up using the EDS: it is where the description of a design, or set of designs, is built up and maintained in a consistent way, together with a record of the design process. It consists of: a collection of instances of particular Functional Unit Module Classes; the relationships between them; values for parameter instances declared in the Functional Unit Modules used; and knowledge inferred from these. The knowledge contained in the DDD at the end of the design process will be a complete specification of a product and its manufacture. It will also form the basis of an historic record of the design process.

From the users' point of view the DDD is thus the working document with which she or he primarily interacts.

2.4 Knowledge Sources

The term "Knowledge Sources" is used to describe a set of EDS sub-systems which are able to infer new knowledge from knowledge already existing in the DDD. It comes from the term inference engine commonly used to describe one part of many expert system architectures. They are not, however, to be thought of as expert systems. Their main role is to provide an integrated level of support functions to the user to help her or him in carrying out any of the diverse types of operations executed in the process of mechanical engineering design. Currently the EDS has four types of Knowledge Sources which are referred to as Engines, and which provide support for: algebraic expression solving and manipulation; spatial relationship reasoning; geometric space occupancy modelling; and the manipulation of Codd type relational models (Tables for example). Each of these Engines are described in more detail below.

2.4.1 The Algebraic Manipulation Engine (AME)

The Algebraic Manipulation Engine aims to support all the expression simplification, manipulation, and solution requirements of an engineering designer using the EDS. It will also be used by the system to follow paths in the relational structure of the design, or designs, described in DDD, mediated by constraint equations, or expressions, or other types of symbolic representations arising from the knowledge declared in Functional Unit Module Class definitions, and to infer new relationships.

The AME currently contains four basic components: the equations solving system mini-press (after the Press system of [Bundy et al 82]); the expression simplifier simp; a symbolic linear equation solving system; and the parameter evaluator eval. The invocation of all of these components of the AME is under the control of the EDS, but the user can, by use of suitable commands, invoke mini-press, the linear equation solver, and simp specifically.

2.4.2 The Spatial Relationship Engine (SRE)

The Spatial Relationship Engine provides an inferencing system able to infer the location of objects from a description of their relative spatial relationships. This Engine is based upon the inference engine at the centre of the RAPT Robot Programming system developed by Popplestone et al [Popplestone et al 80] and [Corner et al 83].

2.4.3 The Geometric Modelling Engine (GME)

This Engine provides both a language for describing geometric space occupancy, a means of manipulating the resulting shapes, and the carrying out certain types of operations on them. The representation language used is based upon Requicha's constructive solid geometry (csg) [Requicha 78].

The GME in the EDS currently has two Geometric Modelling systems within it: the Robmod system developed at Edinburgh by Cameron [Cameron 84], which is a simple but effective polyhedral csg modelling system; and the NONAME system [Armstrong 82], which is a quadric surface csg modeller developed at Leeds as part of the GMP1 project.

2.4.4 The Relation Manipulation Engine (RME)

The fourth Engine of the EDS is the Relation Manipulation Engine. Much of the information used in the design of things mechanical, and indeed electrical, is to be found in the form of tables and graphs. The need arises, therefore, for a means of manipulating and effectively accessing this type of knowledge - to choose certain rows and columns, to join two tables to form a third one, to interpolate between values in a table or lines on a graph, or to graphically display a table on the screen. All the operations mentioned above except for interpolation are identical to those provided by the Relational Calculus developed by Codd for his Relational Model [Codd 70] and [Codd 71], and can be incorporated into a forward chaining predicate logic language [Popplestone 79].

The RME thus provides a subset of the Relational Calculus operations for application to tables of knowledge found in the DDD.

2.5 The Assumption based Truth Maintenance System (ATMS)

The process of design involves the exploration of the space of possible designs and the reduction of that space by the identification of constraints. At any one time during the design of a product, both of its functional and geometric aspects, and of its manufacture, a number of possible alternative designs will typically be being entertained by the engineers working on a product design. In order for the system to be able to represent such multiple designs and to be able to distinguish what knowledge is inconsistent within the DDD, ie. describing different designs - so that the Engines are prevented from inferring new knowledge from existing but inconsistent knowledge in the DDD (which it is quite possible for them to do, but would not be useful in advancing the exploration of the design space), a Truth Maintenance System has been developed to manage the contents of the DDD and the manner in which entries are made in it. This EDS sub-system is based on the work of deKleer [deKleer 84], and is referred to as an Assumption based Truth Maintenance System to distinguish it from other earlier types of dependency directed backtracking Truth Maintenance Systems, of [Doyle 79] and [McAllester 78].

2.6 The Designer System Manager (DSM)

The integrated control and use of the various EDS sub-systems is handled by the Designer System Manager. In the current EDS implementation a simple Blackboard agenda control mechanism is used to control the invocation of the various sub-systems, and to queue up the various operations that can be executed as a result of some user interaction. See [Nii 86a] and [Nii 86b] for a good survey article on Blackboard architectures.

2.7 The System Development Interface (SDI)

Clearly any system whose aim is to support engineering designers in carrying out the design of a product needs to have a powerful and sophisticated user interface. The

design, of both the ergonomic and computational aspects, of such a user interface is beyond the expertise and resources of the Edinburgh EDS team. However, an interface to the system as it is being built is required, and an interface which goes a long way to meeting the demands of those involved in its construction, debugging, and testing. This part of the EDS is therefore called the System Development Interface. Its design and implementation is being carried out with a view to it supporting a more sophisticated prototype user interface later on in the project. In the current EDS implementation the SDI consists of all the commands used to interact with and control any of the sub-systems, plus a prototype multi-window graphical display.

2.8 Design Support Toolkits

The final set of sub-systems included in the EDS architecture diagram, see figure 1, is the set called Design Support Toolkits. These will provide sets of tools to aid users in the design of specific aspects of a product's design. Those identified so far for supporting mechanical engineering design are manufacturing, reliability, maintenance, and cost.

3 A Postscript

The material used in these notes was largely written between June 1985 and June 1987. In most respects it still accurately describes the EDS and its component subsystems. However, as a result of teaching this KR+I-2 course and a growing concern about the problems experienced in trying to build DKBs, I am now proposing to radically change the approach taken to representing domain knowledge in the EDS.

As you will see from section 2.1, the approach taken so far has been a *structured object* approach, in which the builders and users of the DKB are directly concerned with the *structure* of the knowledge-base. The approach that I am now advocating is that of Brachman and Levesque: having tried to convince you of its utility, I think it is time I tried to put it into practice. Their approach, which I refer to as the *functional* approach to building and using knowledge-bases, is perhaps best summarised by quoting the abstract of Levesque's paper *Foundations of a Functional Approach to Knowledge Representation* [Levesque 84].

> "We present a new approach to knowledge representation where knowledge bases are characterized not in terms of the structure they use to represent knowledge, but functionally, in terms of what they can be asked or told about some domain. ... The overall result is a formal foundation for knowledge representation which, in accordance with current principles of software design, cleanly separates functionality from implementation structure".

I have included this postscript to illustrate that knowledge representation and inference is still very much a research subject. There are some powerful techniques available which can be put to very good use. It is just not easy to do, but these difficulties and the research problems we still face present the science of AI with some of its greatest challenges.

References

[Armstrong 82] "Noname Description and Users Manual", Department of Mechanical Engineering, Leeds University, 1982.

[Barrow 83] "Verify: A Program for Proving Correctness of Digital Hardware Designs", AI Technical Report No. 23, November 1983, Fairchild Laboratory for Artificial Intelligence Research, Schlumberger, Palo Alto, CA 94304.

[Bundy et al 82] "Solving Symbolic Equations with PRESS", DAI Research Paper No 171, Department of Artificial Intelligence, University of Edinburgh, 1982.

[Cameron 84] "Modelling Solids in Motion", Ph.D. Thesis, Department of Artificial Intelligence, University of Edinburgh, 1984.

[Codd 70] "A Relational Model of Data for Large Shared Data Banks", CACM 13, No. 6, June 1970.

[Codd 71] "A Data Base Sublanguage Founded on the Relational Calculus", Proc. 1971 ACM SIGFIDET Workshop on Data Description Access and Control, November 1971.

[Corner et al 83] "Reasoning about the Spatial Relationships Derived from a RAPT Program for Describing Assembly by Robot", Proc. 8th. Int. Joint Conf. on Artificial Intelligence, Karlsruhe, FRG, 1983.

[deKleer 84] "Choices without Backtracking", Proc. AAAI-84, pp 79-85, 1984.

[Doyle 79] "A Truth Maintenance System", Artificial Intelligence, Vol. 12, No. 3, pp 231-272, 1979.

[Levesque 84] "Foundations of a Functional Approach to Knowledge Representation", Artificial Intelligence, Vol 23, pp 155-212, 1984

[McAllester 78] "A Three-valued Truth Maintenance System", Technical Report Memo 473, MIT AI Lab., 1978.

[Nii 86a] "Blackboard Systems Part One - The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures", The AI Magazine, Vol. 7. No. 2, pp 38-53, Summer, 1986.

[Nii 86b] "Blackboard Systems Part Two - Blackboard Applications Systems, Blackboard Systems from a Knowledge Engineering Perspective", The AI Magazine, Vol. 7, No. 3, pp 82-106, August, 1986.

[Popplestone 79] "Relational Programming", Machine Intelligence 9, ed. Hayes, J.E., Michie, D., and Mikulich, L.I., Ellis Horwood, 1979.

[Popplestone et al 80] "An Interpreter for a Language for Describing Assemblies", Artificial Intelligence 14 1, pp 79-107, 1980.

[Requicha 78] "Mathematical Foundations of Constructive Solid Geometry: General Topology of Closed Regular Sets", TM27a, Production Automation Project, University of Rochester, USA, 1978.

[Smithers 85] "The Alvey Large Scale Demonstrator Project Design to Product", Proc. Third Int. Conf. on Advanced Information Technology: Artificial Intelligence in Manufacturing - Key to Integration?, November 7-8, 1985, Gottlieb Duttweiler Institute, Zurich, Switzerland, published by North-Holland.

[Smither 87a] "AI-Based Design v geometry-Based Design or Why Design Cannot Be Supported By Geometry Alone", DAI Discussion Paper No 53, 1987.

[Smithers 87b] "Artificial Intelligence and Product Creation: An AI-Based View", DAI Research Paper No 342, 1987.

Tim Smithers
March 1988