Expressiveness of Languages[1]

Jock Mackinlay
Michael R. Genesereth

Computer Science Department
Stanford University
Stanford, California 94305

Abstract

Specialized languages are often a good choice for expressing a set of facts. However, many specialized languages are limited in their expressive power. This paper presents methods for determining when a set of facts is expressible in a language. Some specialized languages have the property that when some collections of facts are stated explicitly, additional facts are stated implicitly. A set of facts should not be stated in such a language unless these implicit facts are correct. This paper presents an algorithm for identifying implicit facts so that they can be checked for correctness. Criteria are also presented for choosing between languages that are sufficiently expressible for a set of facts. This research is being used to build a system that automatically determines when a specialized language is appropriate. It is also relevant to system designers who wish to use specialized languages.

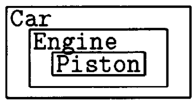
1. Introduction

Specialized languages are used in everyday life as well as in the development of computer software. Common examples include maps, geometry diagrams, and organization charts. General languages, such as predicate calculus, can express a broader range of facts than more specialized languages. However, specialized languages have distinct advantages in efficiency, clarity, or parsimony for certain information.

In an information presentation system [Zdybel 81], it is desirable to use specialized languages for clear, succinct presentation of information to the user. When an information presentation system acts as the user interface for a representation system or database system, it is often expected to present arbitrary collections of information. In such circumstances, taking advantage of specialized languages requires that the presentation system be able to automatically determine when a specialized language is appropriate.

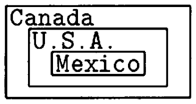
[1]This work was supported in part by grant N00014-K-0004 from the Office of Naval Research.

Many languages have the property that when some collections of facts are stated explicitly, additional facts are stated implicitly. We call such languages *implicit languages.* For example, in the following diagram, the placement of the engine rectangle inside the car rectangle states that an engine is part of a car. Similarly, the placement of the piston rectangle inside the engine rectangle states that a piston is part of an engine.



The diagram also states implicitly that a piston is part of a car because the piston rectangle is contained (indirectly) in the car rectangle.

When choosing an implicit language to express facts, one must make sure that the implicit facts are correct. If the nesting of rectangles represents the relation "next to" instead of the relation "part of", the following diagram states that Canada is next to the U.S.A. and the U.S.A. is next to Mexico:



It also states implicitly that Canada is next to Mexico. Although the two explicit facts are correct, this implicit fact is not. Therefore, this rectangle language is inappropriate for expressing facts about the adjacency of countries.

This paper examines one component involved in the selection of a language: expressiveness. Section 2 describes how messages and facts are related by the conventions of a language and when a fact is stated by a message. Section 3 specifies when a set of facts is expressible in a language. Section 4 describes how to check implicit facts for correctness and presents some criteria for choosing between languages that are sufficiently expressive for a set of facts.

2. Messages and Facts

A *message* is an arrangement of the world intended to convey meaning. Stacks of children's blocks on a table, puffs of smoke in the sky, and spots of ink on a page can all be messages. A *language* is a set of conventions that a speaker and hearer have for constructing and interpreting messages.[2] The process of understanding messages involves identifying them in the world and determining their meaning. Intuitively, the first step is the syntactic interpretation of the message; the second step is the semantic interpretation.

We describe the world and the messages it contains with predicate calculus formulas. For example, the relation `Inside` can be used to describe the nesting of the rectangles in the first diagram in this paper. The formula $\texttt{Inside(}\boxed{\texttt{Piston}}\texttt{,}\boxed{\texttt{Engine}}\texttt{)}$ describes the nesting of two of the rectangles.[3]

We use predicate calculus because it is sufficiently expressive to describe interesting languages. Any formalism with similar characteristics could have been used. The results in this paper do not depend in any direct way on this choice. Variables in predicate calculus formulas are written in lower case. All free variables are universally quantified. Quotes are used around predicate calculus formulas to represent them in propositions.

2.1. Stating Facts in Messages

A language relates facts and messages. For example, the fact `PartOf(Piston,Engine)` is paired with the message $\texttt{Inside(}\boxed{\texttt{Piston}}\texttt{,}\boxed{\texttt{Engine}}\texttt{)}$. Thus, a fact `f` is stated in a language `L` if the corresponding message `m` is satisfied by the world:

Definition 1: `Stated(f,L)` $\Longleftrightarrow$ `Satisfied(m)`.[4]

[2]This definition of "language" is similar to Winograd's: "a system intended to communicate ideas from a speaker to a hearer" [Winograd 71].

[3]A rectangle around a symbol is used to denote the rectangle in a diagram that corresponds (given a language) to what that symbol represents. We use this specialized notation for clarity only; it can be replaced with a functional notation in accordance with predicate calculus syntax.

[4]This definition is related in spirit to Pylyshyn's [Pylyshyn 75] *Semantic Interpretation Function* (SIF). He correctly observed that there are many possible interpretations for a collection of objects in the world. The particular interpretation depends on the SIF that is being used. However, our approach can be used in a computer system to reason about a language.

Prereq(FundMTC,AdvDB)	Qtr(FundCS,Fall)
Prereq(FundCS,DB)	Concur(DB,PL)
Prereq(FundCS,PL)	Qtr(PL,Winter)
Prereq(DB,AdvDB)	Concur(AIProg,AdvDB)
Prereq(PL,OS)	Concur(AdvDB,OS)
Prereq(PL,Compiler)	Concur(OS,Compiler)
Concur(FundAI,FundMTC)	Qtr(Compiler,Spring)

Figure 1: Facts about Classes and Quarters

When a language is defined so that a fact can be stated by more than one message, the formula `m` is a disjunction of clauses describing the various possible messages.

Example: Stacks of blocks. A stack of children's blocks can be used to express facts. Suppose that a speaker and hearer agree that the placement of block $\boxed{\texttt{x}}$ above block $\boxed{\texttt{y}}$ represents the fact `NextTo(x,y)`.[5] Call this language `STACK`. When block $\boxed{\texttt{C}}$ represents `Canada` and block $\boxed{\texttt{U}}$ represents the `U.S.A.`, the following stack states the fact `NextTo(Canada,U.S.A.)`:

$$\begin{array}{c}\boxed{\texttt{C}}\\ \boxed{\texttt{U}}\end{array}$$

In predicate calculus, this message is described by $\texttt{Above(}\boxed{\texttt{C}}\texttt{,}\boxed{\texttt{U}}\texttt{)}$.

In most languages the relationship between messages and facts is fairly stylized; `STACK` is no exception. Schema (1) describes when facts are stated in `STACK`:

$$\begin{array}{l}\texttt{Stated("NextTo(x,y)",STACK)} \Longleftrightarrow \\ \qquad \texttt{Satisfied("Above(}\boxed{\texttt{x}}\texttt{,}\boxed{\texttt{y}}\texttt{)")}\end{array} \quad (1)$$

Example: Layered tree. A diagram consisting of nodes and arcs can be used to express facts. Figure 1 lists a set of facts that describe constraints on class scheduling and prerequisite relationships among several computer science classes. `Prereq` means that one class is a prerequisite for another, `Concur` means that two classes may be taken concurrently, and `Qtr` means that a class is given in a particular quarter. The diagram in Figure 2 is a message that states these facts in a layered tree language called `LAYERTREE`.

[5]The placement of a square around a symbol is used to denote a children's block. This notation is similar to the rectangle notion introduced earlier.

The messages for this language can be described with three predicates: Connected(x,y), SameLayer(x,y) and HorzLabel(x,y). Connected(x,y) means that node x is connected to node y, SameLayer(x,y) means that x and y are on the same layer of the diagram, and HorzLabel(x,y) means that y labels the layer that contains x. The following schema describes how LAYERTREE facts are stated:

$$\begin{aligned}
&\texttt{Stated("Prereq(x,y)",LAYERTREE)} \iff \\
&\quad\texttt{Satisfied("Connected(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)")} \\
&\texttt{Stated("Concur(x,y)",LAYERTREE)} \iff \\
&\quad\texttt{Satisfied("SameLayer(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)")} \\
&\texttt{Stated("Qtr(x,y)",LAYERTREE)} \iff \\
&\quad\texttt{Satisfied("HorzLabel(}\boxed{\texttt{x}}\texttt{,Name(y))")}
\end{aligned} \tag{2}$$

Example: Predicate calculus. Strings displayed on a terminal can be used to express facts. The language PC (for Predicate Calculus) is an example. If the function Wff maps well-formed formulas to strings that represent them, the following schema describes when sentences in PC are stated:

$$\begin{aligned}
&\texttt{Stated(f,PC)} \iff \\
&\quad\texttt{Satisfied("OnTerminal(Wff(f))")}
\end{aligned}$$

Example: The world. The world can be used as a language. If WORLD denotes this language, Stated(f,WORLD) $\iff$ Satisfied(f) describes when facts are stated in this language.

2.2. Constraints on Messages

The physical properties of the world constrain the messages of a given language. For example, it is not possible for two blocks to be mutually above each other. The predicates that are used to describe messages can also be used to construct formal descriptions of these constraints.

Example: Stacks of blocks. The axioms in (3) describe the relation Above among blocks. The first three axioms are anti-reflexivity, anti-symmetry, and transitivity. The last two axioms state that a block is not above another unless it is directly overhead. Thus, if two blocks are in the same stack because they are above (or below) a block, one must be above (or below) the other.

$$\begin{aligned}
&\neg\,\texttt{Above(}\boxed{\texttt{x}},\boxed{\texttt{x}}\texttt{)} \\
&\texttt{Above(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)} \Rightarrow \neg\,\texttt{Above(}\boxed{\texttt{y}},\boxed{\texttt{x}}\texttt{)} \\
&[\texttt{Above(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)} \wedge \texttt{Above(}\boxed{\texttt{y}},\boxed{\texttt{z}}\texttt{)}] \Rightarrow \\
&\quad\texttt{Above(}\boxed{\texttt{x}},\boxed{\texttt{z}}\texttt{)} \\
&[\texttt{Above(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)} \wedge \texttt{Above(}\boxed{\texttt{x}},\boxed{\texttt{z}}\texttt{)} \wedge \boxed{\texttt{y}} \neq \boxed{\texttt{z}}] \Rightarrow \\
&\quad[\texttt{Above(}\boxed{\texttt{y}},\boxed{\texttt{z}}\texttt{)} \vee \texttt{Above(}\boxed{\texttt{z}},\boxed{\texttt{y}}\texttt{)}] \\
&[\texttt{Above(}\boxed{\texttt{y}},\boxed{\texttt{x}}\texttt{)} \wedge \texttt{Above(}\boxed{\texttt{z}},\boxed{\texttt{x}}\texttt{)} \wedge \boxed{\texttt{y}} \neq \boxed{\texttt{z}}] \Rightarrow \\
&\quad[\texttt{Above(}\boxed{\texttt{y}},\boxed{\texttt{z}}\texttt{)} \vee \texttt{Above(}\boxed{\texttt{z}},\boxed{\texttt{y}}\texttt{)}]\,.
\end{aligned} \tag{3}$$

Example: Layered tree diagrams. The axioms in (4) describe the predicates SameLayer and Connected. SameLayer is symmetric and transitive. Connected is transitive. HorzLabel is unconstrained.

$$\begin{aligned}
&\texttt{SameLayer(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)} \Rightarrow \texttt{SameLayer(}\boxed{\texttt{y}},\boxed{\texttt{x}}\texttt{)} \\
&[\texttt{SameLayer(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)} \wedge \texttt{SameLayer(}\boxed{\texttt{y}},\boxed{\texttt{z}}\texttt{)}] \Rightarrow \\
&\quad\texttt{SameLayer(}\boxed{\texttt{x}},\boxed{\texttt{z}}\texttt{)} \\
&[\texttt{Connected(}\boxed{\texttt{x}},\boxed{\texttt{y}}\texttt{)} \wedge \texttt{Connected(}\boxed{\texttt{y}},\boxed{\texttt{z}}\texttt{)}] \Rightarrow \\
&\quad\texttt{Connected(}\boxed{\texttt{x}},\boxed{\texttt{z}}\texttt{)}
\end{aligned} \tag{4}$$

3. Expressiveness

A fact f is *expressible* in a language L if it is consistent with the world for f to be stated in L. Two complications arise when extending this definition to stating sets of facts. First, it might be impossible to state two facts simultaneously in L. Second, every message that states all the facts might also state additional incorrect facts. Therefore, we say that a set of facts F is expressible in L if exactly those facts (and no more) can be stated simultaneously in L:

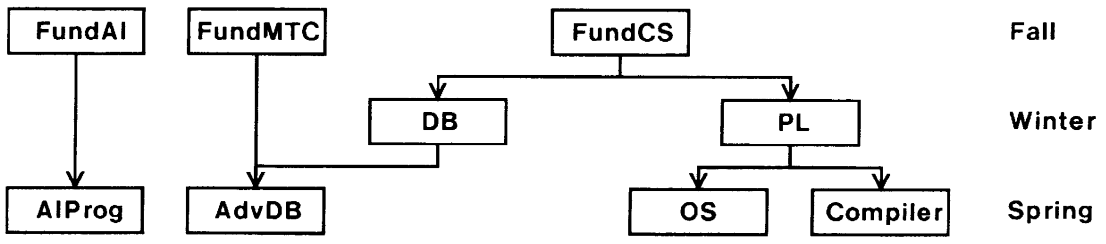


Figure 2: Prerequisite and Class Schedule in LAYERTREE

Definition 2: `Expressible(F,L)` $\Longleftrightarrow$

    Consistent([∀f∈F Stated(f,L)] ∧
        [∀f∉F ¬Stated(f,L)])

The first clause in Definition 2 might not be satisfied for three reasons: there is no message in the language that corresponds to one of the facts, the message that corresponds to one of the facts cannot be stated in the world, and two of the messages cannot be stated simultaneously because they conflict with each other.

Example: No message. A fact is not expressible in a language when there is no message to represent that fact. For example, {p ∨ q} is not expressible in STACK because there is no convention in STACK for representing disjunctions with a stack of blocks. This means that Stated("p ∨ q",STACK) is false.

Example: Message not possible. Sometimes the message that represents a fact cannot be achieved. For example, Stated("NextTo(Canada,Canada)",STACK) is equivalent to Above($\boxed{C}$,$\boxed{C}$), and the Above relation is anti-reflexive.

Example: Messages conflict. Sometimes two or more facts cannot be stated simultaneously because their messages conflict. For example, it is impossible to state both NextTo(Canada,U.S.A.) and NextTo(U.S.A.,Canada) in STACK because the Above relation is anti-symmetric.

For some languages, the only messages that state certain sets of facts also state additional facts implicitly. These additional facts are the *implicit facts* in the message. The second clause in Definition 2 excludes implicit facts because they might be incorrect. However, in some cases these implicit facts are correct. In Section 4 we present an algorithm that can be used to identify these implicit facts so that they can be checked for correctness.

Example: Implicit facts—incorrect. When block $\boxed{M}$ represents Mexico, the following stack states the set {NextTo(Canada,U.S.A.),NextTo(U.S.A.,Mexico)}. The incorrect fact NextTo(Canada,Mexico) is also stated implicitly because block $\boxed{C}$ is above block $\boxed{M}$.

$$\begin{array}{c}\boxed{C}\\ \boxed{U}\\ \boxed{M}\end{array}$$

Example: Implicit facts—correct. The facts about classes and quarters listed in Figure 1 are not expressible in LAYERTREE because the diagram in Figure 2 includes many implicit facts. These implicit facts, listed in Figure 3, are correct. Furthermore, these additional facts would be useful to someone being presented the original facts.

Definition 2 is the basis of an algorithm that determines whether a given collection of facts is expressible in a language. Given a set of facts, assume that these facts are stated and all other facts are not stated. The facts will be expressible if these assumptions are consistent with a description of the world. An automatic deduction technique such as resolution is used to determine if these assumptions and the axioms describing the world are consistent. Although there is no guarantee that the deduction will terminate, transitive axioms and other recursive axioms can be handled using techniques described in [Smith 84]. In general, a depth limit can be used to force termination.

Example: Expressibility algorithm. The proof in Figure 4 shows that the set {Prereq(FundCS,DB), Prereq(DB,AdvDB)} is not expressible in LAYERTREE. The transitive axiom for the relation Connected in (4) is combined with the two positive assumptions to conclude that $\boxed{\text{FundCS}}$ is connected to $\boxed{\text{AdvDB}}$. However, the negative assumption that PreReq(FundCS,AdvDB) is *not* stated leads to a contradiction.

4. Choosing a Language

Expressibility (Definition 2) can be used as a criterion for choosing a language in which to state a given collection of facts: a language should not be used if the facts are not expressible in that language.

Prereq(FundCS,AdvDB)	Qtr(FundAI,Fall)
Prereq(FundCS,OS)	Qtr(FundMTC,Fall)
Prereq(FundCS,Compiler)	Qtr(DB,Winter)
Concur(FundAI,FundCS)	Qtr(AIProg,Spring)
Concur(AIProg,OS)	Qtr(AdvDB,Spring)
Concur(AIProg,Compiler)	Qtr(OS,Spring)
Concur(AdvDB,Compiler)	Concur(FundMTC,FundAI)
Concur(FundCS,FundAI)	Concur(FundCS,FundMTC)
Concur(OS,AIProg)	Concur(AdvDB,AIProg)
Concur(Compiler,AdvDB)	Concur(PL,DB)
Concur(Compiler,AdvDB)	Concur(OS,AdvDB)
	Concur(Compiler,OS)

Figure 3: Implicit Facts of Figure 2

Assumptions

a. Stated("PreReq(FundCS,DB)",LAYERTREE)
b. Stated("PreReq(DB,AdvDB)",LAYERTREE)
c. ¬Stated("PreReq(FundCS,AdvDB)",LAYERTREE)

Proof

d.	Connected($\boxed{\text{FundCS}}$, $\boxed{\text{DB}}$)	a,(2)
e.	Connected($\boxed{\text{DB}}$, $\boxed{\text{AdvDB}}$)	b,(2)
f.	[Connected($\boxed{x}$, $\boxed{y}$) ∧ Connected($\boxed{y}$, $\boxed{z}$)] ⇒ Connected($\boxed{x}$, $\boxed{z}$)	(4)
g.	Connected($\boxed{\text{FundCS}}$, $\boxed{\text{AdvDB}}$)	d,e,f
h.	¬Connected($\boxed{\text{FundCS}}$, $\boxed{\text{AdvDB}}$)	c,(2)
i.	Contradiction	f, h

Figure 4: Proof that a Set Is Inexpressible

In this section we address two problems with this criterion. First, Definition 2 excludes languages in which messages state additional facts. As the class scheduling example suggests, this restriction can be relaxed when the additional facts are correct. This is particularly relevant for implicit languages, in which the additional facts are stated without any additional cost. Second, this criterion does not indicate how to choose between two languages that are sufficiently expressible for a set of facts.

4.1. Using Implicit Languages

Due to the implicit properties of a language, it is often necessary to state more facts than are desired. An *implicit closure* F^* for a set of facts F is a minimal expressible set of facts that contains F. The set difference F^*-F describes the implicit facts that are stated when F^* is used to state F. If all the implicit facts are correct, the implicit language can be used to state F.

Definition 3 shows the relation ImpCl between a set of facts F and an implicit closure F^*. If F is expressible, it is its own implicit closure.

Definition 3: $\forall$F,F^*ImpCl(F,F^*,L) $\iff$
F⊆F^*∧ Expressible(F^*,L)∧
¬ [∃X F⊆X⊂F^*∧ Expressible(X,L)]

Note that ImpCl may not be a function. For example, there are two implicit closures in STACK for the set {NextTo(Canada,U.S.A.),NextTo(Canada,Mexico)}. The following stacks describe these two messages:

$$\begin{array}{cc} \boxed{C} & \boxed{C} \\ \boxed{U} & \boxed{M} \\ \boxed{M} & \boxed{U} \end{array}$$

The first states the implicit fact NextTo(U.S.A., Mexico), while the second states NextTo(Mexico, U.S.A.).

An algorithm for generating the implicit closures is produced by modifying the algorithm used in the last section to determine if a set of facts is expressible. In that algorithm, we assumed that the facts in the set were stated and all the other facts were not stated. However, the negative assumption does not hold for implicit facts. When a contradiction is derived while trying to prove that a set of facts is expressible, we can reverse any negative assumption that was used in the derivation by making the corresponding fact an implicit fact. This will invalidate that particular derivation. When every contradiction is invalidated by placing a fact in the implicit closure, the implicit closure is guaranteed to be expressible because it is consistent with the world. If there is more than one negative assumption that can be reversed to invalidate a contradiction, the alternatives generate different implicit closures. If there are no negative assumptions to be reversed, the set of facts is not expressible.

Example: Generating an implicit closure. The proof in Figure 4 can be used to generate the implicit closure of {Prereq(FundCS,DB),Prereq(DB,AdvDB)}. Since we used ¬Stated("Prereq(FundCS,AdvDB)", LAYERTREE) to derive the contradiction, the implicit closure is the set {Prereq(FundCS,DB), Prereq(DB, AdvDB), Prereq(FundCS, AdvDB)}.

4.2. Choosing Between Languages

There are many criteria for choosing between languages that are sufficiently expressive for a set of facts. For example, one presentation might be more desirable than another because it is:

- smaller
- easier to draw
- in the expected style
- more pleasing
- more dramatic

Developing a precise criterion from each of these examples is beyond the scope of this paper. However, the concepts developed in this paper can be used to suggest how an information presentation system might choose among languages.

We first consider a criterion based on the cost of constructing messages, and then we consider one based on the cost of perceiving messages. Note that the first two examples in the previous list focus on the construction cost, while the rest focus on the perception cost.

The cost of constructing a message is equivalent to the cost of stating the corresponding facts. Under the criterion of construction cost, implicit languages are preferred over other languages because the implicit facts are stated without additional cost. The *implicit kernel* for a set of facts is the smallest subset that can be stated so that its implicit closure contains all of the facts. The cost of stating a set of facts is the cost of stating the facts in its implicit kernel. Definition 4 shows the relation `ImpKer` between a set of facts `F` and its implicit kernel `K`.

Definition 4: $\forall F, K;\ \mathrm{ImpKer}(F,K,L) \iff$
$K \subseteq F \subseteq K^* \wedge \neg\, [\exists X\ X \subset K\ \wedge F \subseteq X^*]$

Example: Comparing layered trees and trees with labeled arcs. The language **ARCTREE**, which is based on labeled arcs, is an alternative to the **LAYERTREE** language for expressing the facts listed in Figures 1 and 3. The diagram in Figure 5 shows how these facts are stated in **ARCTREE**. •

The following schema describes when facts are stated in **ARCTREE**. The predicate `LabArc(n,m,l)` means that node `n` is connected to node `m` by a sequence of arcs that have label `l`.

$$\begin{aligned}&\texttt{Stated("Prereq(x,y)",ARCTREE)} \iff\\ &\quad\texttt{Satisfied("LabArc(}\boxed{\texttt{x}}\texttt{,}\boxed{\texttt{y}}\texttt{,PreReq)")}\\ &\texttt{Stated("Concur(x,y)",ARCTREE)} \iff\\ &\quad\texttt{Satisfied("LabArc(}\boxed{\texttt{x}}\texttt{,}\boxed{\texttt{y}}\texttt{,Concur)")}\\ &\texttt{Stated("Qtr(x,y)",ARCTREE)} \iff\\ &\quad\texttt{Satisfied("LabArc(}\boxed{\texttt{x}}\texttt{,}\boxed{\texttt{y}}\texttt{,Qtr)")}\end{aligned} \tag{5}$$

`LabArc` satisfies the following transitivity axiom:

$$\begin{aligned}&[\texttt{LabArc(}\boxed{\texttt{x}}\texttt{,}\boxed{\texttt{y}}\texttt{,l)} \wedge \texttt{LabArc(}\boxed{\texttt{y}}\texttt{,}\boxed{\texttt{z}}\texttt{,l)}] \Rightarrow\\ &\quad\texttt{LabArc(}\boxed{\texttt{x}}\texttt{,}\boxed{\texttt{z}}\texttt{,l)}\end{aligned}$$

Recall that Figure 3 lists the implicit facts in the **LAYERTREE** diagram (Figure 2) of the facts listed in Figure 1. The facts on the left side of Figure 3 are the implicit facts in the **ARCTREE** diagram. The facts listed on the right side are stated explicitly in Figure 5 by the arcs drawn between class nodes and quarter nodes, and by the arrowheads on the left side of the concurrent arcs. Therefore, the **ARCTREE** implicit kernel is larger than the **LAYERTREE** kernel. Since both languages are tree languages, it is reasonable to assume that the cost of stating facts in them is identical. Thus the **LAYERTREE** language is more economical.

The cost of perceiving messages can also be used as a criterion for choosing between languages. The cost of perceiving messages in the world depends on the nature of the messages. The **LAYERTREE** and **ARCTREE** diagrams are described with different predicates. Because a person looking at these diagrams must ascertain that these predicates are true, the cost of perceiving facts in these diagrams is directly proportional to the cost of determining the truth value of these predicates. By inspection, it is clear that the predicate `LabArc` is more difficult to perceive than `Connected`, `SameLayer`, or `HorzLabel` because the label must be read. This means that the cost of perceiving facts in the **LAYERTREE** diagram is lower than the cost of perceiving the same facts in the **ARCTREE** diagram.

5. Related Work

Genesereth has proposed a representation system that allows and even encourages the use of multiple specialized representation languages [Genesereth 80]. Any criterion for choosing presentation languages can also be used to evaluate specialized representation languages. Implicit languages, in particular, are desirable representation languages because the implicit facts need not be stated explicitly.

Implicit languages are related to the intuitive con-

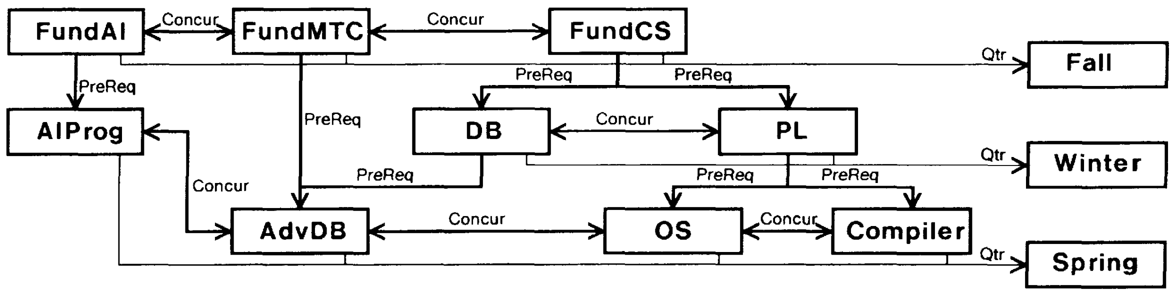


Figure 5: Prerequisite and Class Schedule in **ARCTREE**

cept of direct or analogical representations [Barr 81]. An analogical representation, such as a map, has a structure that directly reflects the world it represents. Sloman has argued for the importance of analogical representations, which he contrasts with "Fregean" representations like predicate calculus [Sloman 71]. His definition of analogical representation consists of an informal collection of examples and a philosophical discussion. We believe that Sloman is incorrect in asserting that analogical representations are dramatically different from the more formal representations used in artificial intelligence. Critiquing Sloman, Hayes has argued for the unity of analogical representations and formal logic languages [Hayes 74]. This paper is a step toward this unity.

Implicit languages have been used in the design of many software systems. One of the earliest uses of an implicit language was Gelernter's Geometry–Theorem Proving Machine [Gelernter 63]. It used a diagram of the problem to help control the search for a proof. The diagram implicitly stated many common facts about geometry. Of course, Gelernter had to be careful that the diagram did not state incorrect facts:

> "If a calculated effort is made to avoid spurious coincidences in the figure, one is usually safe in generalizing any statement in the formal system that correctly describes the diagram."

6. Conclusion

This paper has presented a collection of axioms for describing the expressiveness of languages. These axioms can be used to compute whether a given set of facts is expressible in a language. The paper has also extended these results to implicit languages, in which additional facts may be stated implicitly, including an algorithm for generating implicit closures. Finally, the paper has discussed ways to use these axioms to choose a language in which to express some facts. This research is currently being used to construct an information presentation system that can automatically choose specialized languages for presenting information [Mackinlay 83].

Acknowledgements

We wish to thank David Smith and Polle Zellweger for their incisive comments on drafts of this paper.

References

Barr, A. and E. Feigenbaum, editors. *The Handbook of Artificial Intelligence, Volume 1.* William Kaufmann Inc., 1981, 200-206.

Gelernter, H. "Realization of a Geometry-Theorem Proving Machine." In E. Feigenbaum and J. Feldman, editors. *Computers and Thought.* McGraw-Hill, 1963, 134-152.

Genesereth, M. R. "Metaphors and Models." *Proc. AAAI 80.* Stanford University, August 1980, 208-211.

Hayes, P. J. "Some Problems and Non-Problems in Represention Theory." *Proc. AISB Summer Conference,* 1974, 63-79.

Mackinlay, J. "Intelligent Presentation: The Generation Problem for User Interfaces." Report HPP-83-34, Computer Science Department, Stanford University, 1983.

Pylyshyn, Z. W. "Representation of Knowledge: Non-linguistic Forms. Do We Need Images and Analogues?" *Proc. TINLAP 75.* Massachusetts, June 1975, 174-177.

Sloman, A. "Interactions Between Philosophy and Artificial Intelligence: The Role of Intuition and Non-Logical Reasoning in Intelligence." *Artificial Intelligence* 2 (1971) 209-225.

Smith, D. E. and M. R. Genesereth, "Controlling Recursive Inference." Report HPP-84-6, Computer Science Department, Stanford University, 1984.

Winograd, T. "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language." PhD Thesis, MIT, 1971.

Zdybel, F., N. Greenfeld, M. Yonke, and J. Gibbons. "An Information Presentation System." *Proc. IJCAI 81.* Vancouver, August 1981, 978-984.