January 1985

Report No. STAN-CS-85-1037
Also numbered: HPP-84-4

Expressiveness and Language Choice

by

Jock MacKinlay
Michael R. Genesereth

Department of Computer Science

Stanford University
Stanford, CA 94305

Stanford Heuristic Programming Project
Report No. HPP 84-4

January 4, 1985

Expressiveness and Language Choice

by

Jock MacKinlay, and Michael R. Genesereth

Computer Science Department
Stanford University
Stanford, California 94304

Expressiveness and Language Choice[1]

Jock Mackinlay
Michael R. Genesereth

Abstract

Specialized languages are often more appropriate than general languages for expressing certain information. However, specialized languages must be chosen carefully because they do not allow **all** sets of facts to be stated. This paper considers the problems associated with choosing among specialized languages. Methods are presented for determining that a set of facts is expressible in a language, for identifying when additional facts are stated accidentally, and for choosing among languages that can express a set of facts. This research is being used to build a system that automatically chooses an appropriate graphical language to present a given set of facts.

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. Such languages often make facts easier to express and easier to understand.

When an information presentation system [15] 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 determine automatically when a specialized language is appropriate.

Some languages have the property that when some collections of facts are stated explicitly, additional facts are stated **implicitly.**[2] 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.

---

[1] **This work was supported in part by grant N00014-K-0004 from the Office of Naval Research. An earlier version of this paper was presented at AAAI-83, Austin, Texas.** **The address of the authors is: Computer Science Department; Stanford University, CA 94305**

[2] **Note that implicit facts are not conventional [8] or conversational [6] implicature because the implicit facts are actually stated in the message. They are implicit only in the sense that they have not been stated explicitly.**

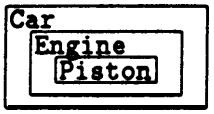

Figure 1: Some "part of" facts

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:

Figure 2: Some "next to" facts

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

A language's expressiveness is the major criterion for choosing a language to state **a** given set of facts: a language that cannot express the facts should not be used. However, additional criteria are needed to choose among languages that are sufficiently expressive for a set of facts. Two of these criteria are how easy it is to state the facts in the languages and how easy it is to perceive the facts once they are stated. In this paper, we consider all three of these **criteria.** Section 2 describes how messages and facts are related by the conventions of a language and specifies when a fact is stated by a message. Section 3 **specifies** when **a** set of facts is expressible in a language. It also describes how to identify implicit facts so that they can be checked for correctness. Section 4 considers the problem of choosing among languages **that** are sufficiently expressive for a set of facts. Finally, Section **5** discusses **the related** research.

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. The process of understanding messages involves identifying them in the world and determining their meaning.

In this paper, we describe the world and the messages it contains with

predicate calculus formulas.[3] For example, we describe the nesting of two of the rectangles in the first diagram with the formula **Inside(Piston-Rec, Engine-Rec)**. We say that this formula is true by writing

$$Satisfied(\texttt{Inside(Piston-Rec,Engine-Rec)}, \texttt{Fig1}).$$

The constant Figl, a situation [10], represents a complete state of the world that includes Figure 1. The situation argument allows us to decribe more than one arrangement of the world. A formula that describes a message is called a ***message formula.***

2.1 Stating Facts in Messages

A *language* is a set of conventions that a speaker and hearer have for constructing and interpreting messages.[4] We can describe a language by specifying the correspondence between facts about the world and the message formulas that represent them. For example, the fact **PartOf(Piston,Engine)** is paired with the message formula **Inside(Piston-Rec,Engine-Rec)**. A fact is stated if the corresponding message formula is true:

Definition *1 A fact* **f is stated** *with language* **1** *in situation s if the associated message formula* $\bar{\mathbf{f}}$ *is satisfied by* **s**:[5]

$$Stated(f, \mathtt{l}, \mathtt{s}) \iff Satisfied(\bar{\mathbf{f}}, \mathtt{s}).$$

We use the overbar notation to represent the binary function from a language **1** and a fact **f** to the corresponding message formula $\bar{\mathbf{f}}$. For notational simplicity, we have omitted **1** from the notation because it can always be determined from the surrounding context. Some languages have regularities that make it possible to describe this function recursively. For example, the language used in Figure **1** can be described with the following equation:

$$\overline{\texttt{PartOf}(x, y)} = \texttt{Inside}(\bar{x}, \bar{y}).$$

The variables in this equation are universally quantified outside of the bar and range over expressions. If a fact includes variables of its own, they must be quantified within the bar notation.

[3] Variables begin with lower case while constants begin with upper case. Free variables are universally quantified.

This definition of "language" is similar to Winograd's: "a system intended to communicate ideas from a speaker to a hearer" [14].

This definition is related in spirit to Pylyshyn's [11] *Semantic Interpretation Function* (SIP). 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.

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 existence of one block anywhere above another represents a "**next** to" fact between the objects associated with those blocks. Call this language STACK. When Canada is the block C and U.S.A. is the block U, the following stack, which is described by **Above(Canada,U.S.A.)**, states the fact **NextTo(Canada,U.S.A.)** in STACK:

The schema in (1) describes when facts are stated in STACK.

$$Stated(\mathbf{NextTo(x,y),STACK,s}) \Longleftrightarrow Satisfied(\mathbf{Above}(\bar{x},\bar{y}),\mathbf{s}) \quad (1)$$

Example: Layered tree. A diagram consisting of nodes and arcs can be used to express facts. Figure 3 lists a set of facts describing the constraints on class scheduling and the 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 message in Figure 4 states these facts in a layered tree language called **LAYERTREE**.

```
Prereq(FundAI,AIProg)          Concur(FundMTC,FundCS)
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 3: Facts about Classes and Quarters

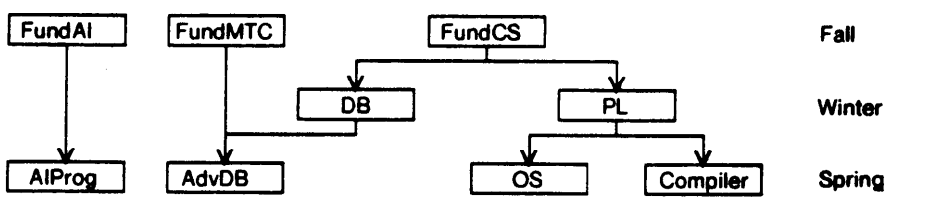

Figure 4: Prerequisites and Class Schedule in **LAYERTREE**

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 schema in (2) describes how **LAYERTREE** facts are stated.

$$\begin{aligned} &\mathit{Stated}(\texttt{Prereq(x,y),LAYERTREE,s}) \Longleftrightarrow \\ &\quad \mathit{Satisfied}(\texttt{Connected}(\bar{x},\bar{y}),\texttt{s}) \\ &\mathit{Stated}(\texttt{Concur(x,y),LAYERTREE,s}) \Longleftrightarrow \\ &\quad \mathit{Satisfied}(\texttt{SameLayer}(\bar{x},\bar{y}),\texttt{s}) \\ &\mathit{Stated}(\texttt{Qtr(x,y),LAYERTREE,s}) \Longleftrightarrow \\ &\quad \mathit{Satisfied}(\texttt{HorzLabel}(\bar{x},\bar{y}),\texttt{s}) \end{aligned} \tag{2}$$

Example: Predicate calculus. Strings displayed on a terminal can be used to express facts. The language PC (for Predicate Calculus) is an example. The schema in (3) describes when facts are stated in **PC.**

$$\mathit{Stated}(\texttt{f,PC,s}) \Longleftrightarrow \mathit{Satisfied}(\texttt{OnTerminal}(\bar{\texttt{f}}),\texttt{s}) \tag{3}$$

Example: The world. The world can be used as a language. If WORLD denotes this language, the schema in (4) describes when facts are stated in this language. Note that there is no bar in this schema because **f is its own message** formula. This language might be used by a robot to decide that it does not have to explicitly store a fact in its representation system. If the fact is satisfied by the current state of the world, it is already stated using **WORLD.** For example, the exact position of a door does not have to be remembered because the robot can always go and look at it.

$$\mathit{Stated}(\texttt{f,WORLD,s}) \Longleftrightarrow \mathit{Satisfied}(\texttt{f,s}) \tag{4}$$

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 describe these constraints formally.

Example: Stacks of ***blocks.*** The axioms[6] in (5) describe the relation **Above among** blocks. The first three **axioms are anti-reflexivity, anti-symmetry,** and transitivity. The last two axioms assert that blocks are stacked in columns:

[6] **These axioms are similar to Montague's meaning postulates [2]. However, we are describing the physical properties of the world rather than placing restrictions on possible models.**

if two blocks are in the same stack because they are above (or below) a block, one must be above (or below) the other.

$$
\begin{aligned}
&\neg \mathbf{Above(x,x)} \\
&\mathbf{Above(x,y)} \Rightarrow \neg \mathbf{Above(y,x)} \\
&\mathbf{[Above(x,y) \wedge\ Above(y,z)]} \Rightarrow \mathbf{Above(x,z)} \\
&\mathbf{[Above(x,y) \wedge Above(x,z) \wedge\ y \neq z]} \Rightarrow \\
&\qquad \mathbf{[Above(y,z) \vee Above(z,y)]} \\
&\mathbf{[Above(y,x) \wedge\ Above(z,x) \wedge\ y \neq z]} \Rightarrow \\
&\qquad \mathbf{[Above(y,z) \vee Above(z,y)]}.
\end{aligned}
\tag{5}
$$

Example: Layered tree diagrams. The axioms in (6) describe the predicates **SameLayer and Connected. SameLayer is symmetric** and **transitive. Connected is transitive. HorzLabel is unconstrained.**

$$
\begin{aligned}
&\mathbf{SameLayer(x,y)} \Rightarrow \mathbf{SameLayer(y,x)} \\
&\mathbf{[SameLayer(x,y) \wedge SameLayer(y,z)]} \Rightarrow \\
&\qquad \mathbf{SameLayer(x,z)} \\
&\mathbf{[Connected(x,y) \wedge\ Connected(y,z)]} \Rightarrow \\
&\qquad \mathbf{Connected(x,z)}
\end{aligned}
\tag{6}
$$

3 Expressiveness

This section presents a formal **definition** of expressiveness by specifying which sets of facts are expressible in a language. Intuitively, a set of facts f s is ***expressible*** in a language **1 if there exists a situation that states every fact in f s.** The **difficulty** with this intuition is that every situation that **states f s** might also state additional incorrect facts. Thus our formal definition of expressiveness requires that the situation state exactly the facts in f s. Later, we relax this definition when the additional facts turn out to be correct.

3.1 Expressible Sets of Facts

Definition 2 *A* **set** of facts f **s is expressible** ***in language*** 1 ***if there exists a situation*** **s** ***that states every fact in*** f **s** ***and does not state any other fact:***

$$
\begin{aligned}
&\mathit{Expressible}(\mathbf{fs}, \mathbf{1}) \Longleftrightarrow \\
&\qquad 3\ \mathbf{s}.\ (\mathbf{[\forall f \in fs}.\mathit{Stated}(\mathbf{f,1,s})] \wedge \\
&\qquad\qquad \mathbf{[\forall f \notin fs}.\neg \mathit{Stated}(\mathbf{f,1,s})]).
\end{aligned}
$$

The **two conjuncts** in Definition 2 can be used to identify when a set of facts is not expressible in a language.

The first conjunct asserts that **fs** is not expressible when 1 does not associate a message formula with some fact in f **s**. A **naïve** test of expressibility is to check that each fact has an associated message formula. However,

the first conjunct identifies two additional cases where f **s** is not expressible. First, a message formula associated with a fact in the set might not be satisfied by any possible situation. Second, two or more facts might conflict in such a way that they cannot be stated simultaneously. The first difficulty can be avoided by excluding unsatisfiable message formulas from the definition of the language. However, the second difficulty is unavoidable because messages are part of the world and must conform to physical laws.

Example: Message ***not*** possible. Some facts are associated with message formulas that cannot be satisfied. For example, *Stated*(**NextTo(Canada,** Canada), STACK, **s)** would be stated by **Above(**$\overline{\text{Canada}}$**,**$\overline{\text{Canada}}$**)**, but the Above relation is anti-reflexive.

Example: Messages ***conflict***. Two or more facts may be impossible to state 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.

The second conjunct in Definition 2 limits the situations to ones that do not include additional facts. These additional facts are the ***implicit facts*** in a message. The second conjunct rejects languages that have implicit facts because the implict facts might be incorrect. However, in some cases these implicit facts are correct. The next subsection presents an algorithm for identifying these implicit facts so that they can be checked for correctness.

Example: Incorrect implicit facts. When block ❑ is associated with Mexico, the following stack states the set

$$\{\texttt{NextTo(Canada,U.S.A.)},\texttt{NextTo(U.S.A.,Mexico)}\}.$$

The incorrect fact **NextTo(Canada,Mexico)** is also stated implicitly because block ❑ is above block [M].

$$c^{C}1$$
$$c^{U}1$$
$$c^{M}1$$

Example: Correct implicit fac ta. The facts about classes and quarters listed in Figure 3 are not expressible in **LAYERTREE** because the diagram in Figure 4 includes many implicit facts. These implicit facts, listed in Figure 5, are correct. Furthermore, these additional facts would be useful to someone being presented the original facts.

Definition 2 is a sharp definition of expressiveness: a set of facts is not expressible unless precisely that set can be stated. This raises an interesting question about conjunctive facts. Namely, is *Stated*(**p∧q,** 1, **s)** true whenever *Stated*(**p,** 1 **,s)** and *Stated*(**q,l,s)** are true? The answer depends on the definition of the language. Some languages (for example, STACK) do not

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,AdvProg)	Concur(OS,AdvDB)
	Concur(Compiler,OS)

Figure 5: Implicit Facts of Figure 2

have message formulas associated with conjunctive facts, in which case $p \wedge q$ is not stated. The set $\{p, q, p \wedge q\}$ is not expressible in these languages. However, many languages are defined so that $p \wedge q$ is implicitly stated when p and q are stated, in which case $p \wedge q$ is stated. The set $\{p, q, p \wedge q\}$ is expressible in these languages, but the set $\{p, q\}$ is not expressible. Finally, some languages (for example, predicate calculus) are defined so that $p \wedge q$ must be stated explicitly, in which case $p \wedge q$ is not stated. In these languages, both sets ($\{p, q\}$ and $\{p, q, p \wedge q\}$) are expressible.

Definition 2 allows us touseanautomatic deduction system to determine 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. Transitive axioms and other recursive axioms can be handled using rewriting techniques [13]. In general, a depth limit can be used to force termination.

Example: Expressibility algorithm. The proof in Figure 6 shows that the set {Prereq(FundCS,DB), Prereq(DB,AdvDB)} is not expressible in **LAYERTREE**. Steps d and e use the expressibility definition to convert the facts to be stated into their associated message formulas. Step f combines the transitivity axiom for Connected with these message formulas to conclude that $\overline{\text{FundCS}}$ is connected to $\overline{\text{AdvDB}}$. However, step g concludes that $\overline{\text{FundCS}}$ is ***not*** connected to $\overline{\text{AdvDB}}$ by converting the negative assumption of step c into the negation of the corresponding message formula. Since this conclusion directly contradicts the conclusion of step f, the set of facts is not expressible.

Assumptions

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

Proof

d. Connected($\overline{\text{FundCS}}$,$\overline{\text{DB}}$)	a,(2)
e. Connected($\overline{\text{DB}}$,$\overline{\text{AdvDB}}$)	b,(2)
f. Connected($\overline{\text{FundCS}}$,$\overline{\text{AdvDB}}$)	d,e,(6)
g. ¬ Connected($\overline{\text{FundCS}}$,$\overline{\text{AdvDB}}$)	c,(2)
h. ∅	f,g

Figure 6: Proof that a Set Is Not Expressible

3.2 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 s* for a set of facts f s is a minimal expressible set** of facts that contains **f s.** Formally, **f s* denotes the relation ImpCl between a language 1, a set of facts f s, and an implicit closure f s*. For notational simplicity, we have omitted the language name** from the notation because it can always be determined from the surrounding context.

Definition 3 ***The set*** **of facts f s*** is ***an*** **implicit closure of** ***the set*** **f s** ***if there*** **is** ***no smaller*** **expressible** ***set that contains*** **f s:**

$$\text{ImpCl}(fs, fs^*, l) \iff fs \subseteq fs^* \wedge \textit{Expressible}(fs^*, l) \wedge \neg[\exists x.\ fs \subseteq x \subset fs^* \wedge \textit{Expressible}(x, l)]$$

The set difference **f s*-f s** describes the implicit **facts that are stated when f s* is used to state f s. If all the implicit facts are correct, the implicit language can be used to state f s. If f s is expressible, it is its own implicit** closure.

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:

c^{C}l	c^{C}l
c^{U}l	c^{M}l
c^{M}l	c^{U}l

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

The algorithm for determining if a set of facts is expressible can be modified to produce an algorithm for generating implicit closures. In the expressibility algorithm, we assumed that the facts not in the set were ***not*** stated. However, this 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 assuming that the corresponding fact is an implicit fact. This invalidates 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. Of course, sometimes there will not be a negative assumption that can be reversed. In this case, the set of facts is not expressible.

Example: Generating an implicit closure. The proof in Figure 6 can be used to generate the implicit closure of {**Prereq(FundCS , DB)** , **Prereq(DB,AdvDB)**}. Since we used ¬*Stated*(**Prereq(FundCS,AdvDB)**, LAYERTREE, **s**) to derive the contradiction, the implicit closure is the original set plus **Prereq(FundCS,AdvDB)**.

4 Choosing Among Languages

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 be used only if the facts are expressible in that language. However, other criteria are needed to choose among languages that ***are*** sufficiently expressive for a set of facts. This section describes two fundamental criteria for choosing among sufficiently expressive languages: the cost of constructing messages and the cost of interpreting messages. We illustrate these criteria by comparing the diagram in Figure 4, which is written in LAYERTREE, with the diagram in Figure 7, which is written in the language NETWORK. Since both these diagrams state the same facts, expressiveness cannot be used to choose between NETWORK and LAYERTREE.

NETWORK uses **labelled** arcs to represent relations. The schema in (7) describes when facts are stated in NETWORK. The predicate **LabelledArc(n,m,arc)** means that node **n** is connected to node **m** by a sequence of arcs that have the label arc.

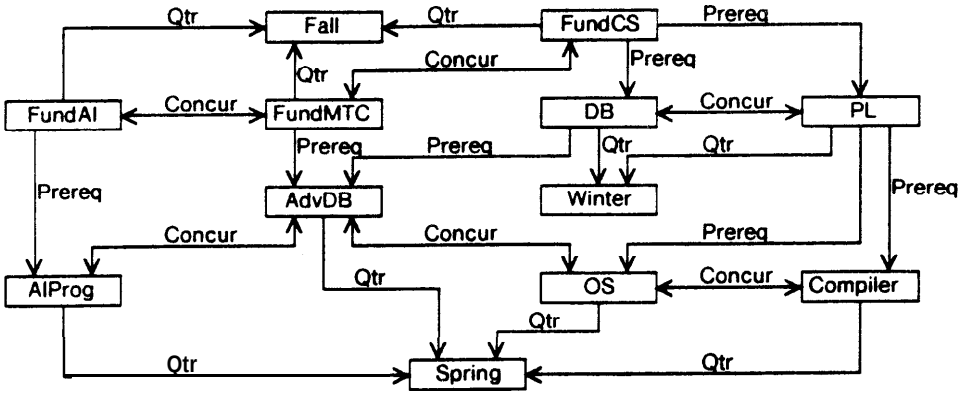

Figure 7: Prerequisites and Class Schedule in **NETWORK**

$$
\begin{aligned}
&\mathit{Stated}(\mathtt{Prereq(x,y),NETWORK,s}) \Longleftrightarrow \\
&\quad \mathit{Satisfied}(\mathtt{LabelledArc}(\bar{x},\bar{y},\mathtt{Prereq}),\mathtt{s}) \\
&\mathit{Stated}(\mathtt{Concur(x,y),NETWORK,s}) \Longleftrightarrow \\
&\quad \mathit{Satisfied}(\mathtt{LabelledArc}(\bar{x},\bar{y},\mathtt{Concur}),\mathtt{s}) \\
&\mathit{Stated}(\mathtt{Qtr(x,y),NETWORK,s}) \Longleftrightarrow \\
&\quad \mathit{Satisfied}(\mathtt{LabelledArc}(\bar{x},\bar{y},\mathtt{Qtr}),\mathtt{s})
\end{aligned}
\tag{7}
$$

LabelledArc satisfies the following **transitivity axiom:**

$$
\begin{aligned}
&[\mathtt{LabelledArc(x,y,arc)} \wedge \mathtt{LabelledArc(y,z,arc)}] \Rightarrow \\
&\quad \mathtt{LabelledArc(x,z,arc)}
\end{aligned}
$$

4.1 Construction Cost

A set of facts can be stated in a language by arranging the world so that all of the message formulas associated with those facts are satisfied. We represent the effort expended to construct a message by the cost function **ConCost($\bar{f}$,s)**, which maps message formulas to their corresponding construction costs. The situation variable **s** is included because the current arrangement of the world influences the construction cost.

Construction cost can be used as a criterion for choosing **among** languages: choose the language with the lowest construction cost. For an arbitrary language, the construction cost might be astronomical. For example, launching the space shuttle would be an expensive way to represent America's patriotism. However, most languages are designed to have reasonable construction costs.

A simple calculation of the cost of stating a set of facts is to sum the costs of stating each fact individually. However, this simple calculation does

not work for all languages. Stating one fact changes the world, thereby changing the cost of stating additional facts. Hence, any general construction cost calculation must check all orderings in which the facts might be stated. Therefore, a general calculation is much less efficient than the simple summation.

The simple summation cannot be used with implicit languages because the stating of the explicit facts reduces the cost of stating the implicit facts to zero. However, the cost can be calculated by determining the cost of stating the set of explicit facts. We call this set the ***explicit kernel.*** Formally, the explicit kernel for a set of facts is the smallest subset that can be stated such that its implicit closure contains all the facts. Definition 4 defines the relation **ExpKer** between a set of facts **f s** and its explicit kernel **k.**

Definition 4 ***The*** **explicit kernel** ***for a set of facts*** **f s** ***is the smallest subset*** **k** ***such that*** k^* ***contains*** **f s:**

$$\mathbf{ExpKer(fs,k,l)} \iff \mathbf{k \subseteq fs \subseteq k^* \wedge \neg [\exists x.x \subset k \ A \ fs \subseteq x^*]}$$

The **NETWORK** diagram shown in Figure 7 has a larger explicit kernel than the **LAYERTREE** diagram shown in Figure 4. Figure 5 lists the implicit facts of the **LAYERTREE** diagram in two columns. The facts in the right column are stated explicitly in the **NETWORK** diagram by arcs and arrowheads. The facts in the left column are the implicit facts **in the NETWORK diagram.**

Since NETWORK and LAYERTREE are both tree languages, we assume a constant, identical cost for stating facts in these languages. Therefore, their construction cost is directly proportional to the size of their explicit kernels, and **LAYERTREE** is more desirable than **NETWORK** for stating the facts in Figure 3.

4.2 Interpretation Cost

In this section, we consider another criterion for choosing among sufficiently expressive languages: interpretation cost. Interpretation cost is more significant than construction cost. Most languages are designed so that messages are easy to construct. However, the cost of interpreting messages in specialized languages depends on the facts that are stated.

We represent the cost of interpreting a message with a cost function **IntCost ($\bar{f}$)** that maps a message formula into its corresponding interpretation cost. For construction costs, we compared two messages by comparing the construction cost for the entire set of facts. However, perception of a specific fact does not necessarily require the interpretation of the entire message. Some facts might be more important than others, making it important that

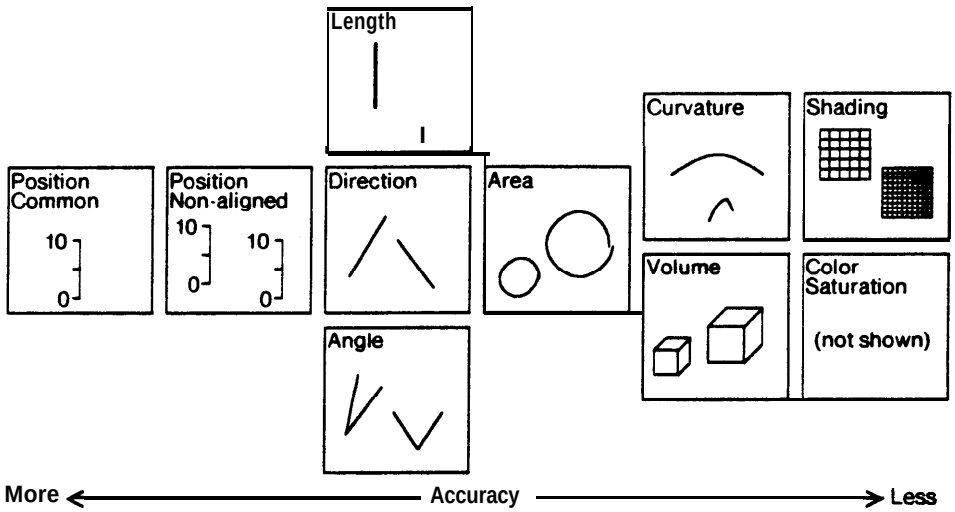

Figure 8: Elementary Perceptual Tasks

they have low interpretation costs. A more realistic calculation is to average the interpretation costs so that a few high costs do not bias the comparison. Another possibility is to calculate a weighted sum of the interpretation costs, where the weights represent the importance of the corresponding facts. We assume that the facts in the LAYERTREE and NETWORK diagrams are of equal importance.

To recognize that facts are stated in the presentation languages **LAYERTREE** and NETWORK, a person must determine whether the predicates Connected, **SameLayer**, **HorzLabel**, and **LabelledArc** are true. The interpretation cost for message formulas involving these predicates could be derived from a theory of human graphical perception. Although a confirmed theory does not exist, Cleveland and McGill [3] have proposed a tentative one. We can use their tentative theory to suggest how a cost function might be determined. This approximate calculation is sufficient to show that LAYERTREE is easier to interpret than NETWORK.

Cleveland and McGill identify ten elementary perceptual tasks that people must accomplish to extract quantitative information from graphical languages. They hypothesize an ordering of the perceptual tasks on the basis of accuracy. Figure 8 lists these tasks from more to less accurate. They also present the results of two experiments that partially confirm this ordering. One experiment compared judgements of position along a common scale to judgements of length. The other compared judgements of position along a common scale to judgements of angle. Both experiments showed that people were more accurate at judging position along a common scale.

Determining the truth of **SameLayer** and HorzLabel predicates simply involves the perception of position along a common scale. However, Connected and **LabelledArc** do not fit directly into their theory. Cleveland and McGill are primarily interested in the presentation of quantitative information, which does not normally involve perceiving that two graphic objects are connected. Therefore, we add another perceptual task to their list of perceptual task: the perception of connected objects. Furthermore, we assume that people are as accurate at judging connection as position along a common scale.[7]

Given these assumptions, we can develop a rough estimate of the interpretation costs for messages in **LAYERTREE** and NETWORK. In **LAYERTREE**, facts are stated by position and connection. **SameLayer** and HorzLabel are both instances of position on a common scale. Connected requires more than one primitive perceptual task to determine its truth value: the connection perceptual task does not include judging the direction of the connection, which requires the additional perceptual effort of noticing the arrowhead's location. We assume that Connected is twice as hard to perceive as **SameLayer** and HorzLabel. Thus, the estimate of the interpretation cost for LAYERTREE is 49 (see Figure 9). **In NETWORK,** all the facts are stated with **LabelledArc**, which also involves more than one primitive perceptual task. In addition to perceiving the connection and the arrowhead, a person must perceive the label on the arc. Optimistically[8], we assume that **labelled** arcs have an interpretation cost of 3. Thus, the estimate of the interpretation cost for **NETWORK** is 117. The table in Figure 9 summarizes both calculations of interpretation cost and shows that the **LAYERTREE** language is better than the **NETWORK** language for the facts in Figure 3.

5 Related Research

In artificial intelligence, most of the related research on specialized languages has been done on representation languages for problem solving. Genesereth has proposed a representation system that allows and even encourages the use of multiple specialized representation languages [5]. In fact, any criterion for choosing presentation languages can also be used to evaluate specialized

[7] **Connection is easier to judge than the quantitative value of a position. However, we are primarily interested in qualitative positions (such as "above" and "below"), which appear to be as easy to perceive as connection.**

[8] **Intuitively, the perception of the transitivity of connections becomes more difficult as the number of connections increase. This suggests that connection perception in NETWORK is more costly than in LAYERTREE because there are more connections. However, even with this optimistic cost assumption, NETWORK is found to be more difficult to interpret than LAYERTREE.**

		LAYERTREE	NETWORK
Prereq	(10 facts)	20	30
Concur	(20 facts)	20	60
Qtr	(9 facts)	9	27
Total	(39 facts)	49	117

Figure 9: Interpretation Costs for LAYERTREE and NETWORK

representation languages. In particular, implicit languages are desirable representation languages because the implicit facts need not be stated explicitly.

Implicit languages are related to the intuitive concept of direct or analogical representations [1]. 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 [12]. 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 [7]. 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 [4]. 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, Gelemter 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 theory of expressiveness for languages. This theory can be used to determine 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 and has presented an algorithm for generating implicit closures. Finally, the paper has discussed

how to choose an appropriate language in which to express some facts. This research is currently being used to construct an information presentation system that can automatically choose specialized graphical languages for presenting information [9].

Acknowledgements

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

References

1. **BARR,** A., **AND FEIGENBAUM,** E. (Eds.) ***The Handbook of Artificial Intelligence 1.*** William Kaufmann Inc. (1981), 200-206.

2. **DOWTY,** D. R., **WALL,** R. E., **AND PETERS,** ***S. Introduction to Montague Semantics.*** D. Reidel Publishing Company, Boston, MA, (1981),224-225.

3. **CLEVELAND,** W. S., **AND MCGILL,** R. Graphical preception: theory, experimentation, and application to the development of graphical methods. ***Journal of the American Statistical Association. 79, 387,*** (September 1984), 531-554.

4. **GELERNTER,** H . Realization of a geometry-theorem proving machine. In Feigenbaum, E., and Feldman, J. (Eds.) ***Computera and Thought.*** McGraw-Bill, (1963), 134-152.

5. **GENESERETH,** M . R . Metaphors and models. ***Proceedings AA AI 80.*** Stanford University, CA, (August 1980), 208-211.

6. **GRICE,** H. P. Logic and conversation. In Davidson, D., and Harman, ***G.*** (Eds.) ***The Logic of Grammar.*** Dickinson Publishing Co., Encino, CA, (1975), 64-75.

7. HAYES, P . J . Some problems and non-problems in represention theory. ***Proceedings AISB Summer Conference.*** (1974), 63-79.

8. **KARTTUNEN, L. AND PETERS,** S . Conventional implicature. In Oh C., and Dinneen, D. A. (Eds.) ***Syntax and Semantics 11.*** Academic Press, New York, (1979), l-56.

9. **MACKINLAY,** J. Intelligent presentation: the generation problem for user interfaces. Report HPP-83-34, Computer Science Department, Stanford University, CA, (1983).

10. **McCarthy, J . and Hayes,** P. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer B., and Michie D. (Eds.) ***Machine Intelligence 4***. Edinburgh University Press, (1969), 463-502.

11. **Pylyshyn,** Z . W . Representation of knowledge: non-linguistic forms. do we need images and analogues? Proceedings ***TINLAP*** 75. Massachusetts, (June 1975), 174- 177.

12. **Sloman,** A. Interactions between philosophy and artificial intelligence: the role of intuition and non-logical reasoning in intelligence. ***Artificial Intelligence. 2,*** (1971), **209-225.**

13. **Smith, D.** E., **and Genesereth,** M . R . Controlling recursive inference. Report HPP-84-6, Computer Science Department, Stanford University, CA, (1984).

14. **W inograd,** T . Procedures as a representation for data in a computer program for understanding natural language. PhD Thesis, MIT, MA, (1971).

15. **Zdybel,** F . , **Greenfeld, N., Yonke, M., and Gibbons J . An** information presentation system. Proceedings ***IJCAI 81***. Vancouver, (August 1981), 978-984.