**What to Represent?**

Let us first consider what kinds of knowledge might need to be represented in AI systems:

**Objects**

-- Facts about objects in our world domain. *e.g.* Guitars have strings, trumpets are brass instruments.

**Events**

-- Actions that occur in our world. *e.g.* Steve Vai played the guitar in Frank Zappa's Band.

**Performance**

-- A behavior like *playing the guitar* involves knowledge about how to do things.

**Meta-knowledge**

-- knowledge about what we know. *e.g.* Bobrow's Robot who plan's a trip. It knows that it can read street signs along the way to find out where it is.

Thus in solving problems in AI we must represent knowledge and there are two entities to deal with:

**Facts**

-- truths about the real world and what we represent. This can be regarded as the *knowledge level*

**Representation of the facts**

which we manipulate. This can be regarded as the *symbol level* since we usually define the representation in terms of symbols that can be manipulated by programs.

We can structure these entities at two levels

**the knowledge level**

-- at which facts are described

**the symbol level**

-- at which representations of objects are defined in terms of symbols that can be manipulated in programs (see Fig. [5](http://www.cs.cf.ac.uk/Dave/AI2/node33.html#figfacts))

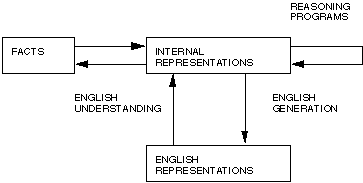


Fig [5](http://www.cs.cf.ac.uk/Dave/AI2/node33.html#figfacts) Two Entities in Knowledge Representation

English or natural language is an obvious way of representing and handling facts. Logic enables us to consider the following fact: *spot is a dog* as *dog(spot)*We could then infer that all dogs have tails with: tex2html_wrap_inline7154 :*dog(x)* tex2html_wrap_inline7156 *hasatail(x)* We can then deduce:

*hasatail(Spot)*

Using an appropriate backward mapping function the English sentence *Spot has a tail can be generated.*

The available functions are not always one to one but rather are many to many which is a characteristic of English representations. The sentences *All dogs have tails* and *every dog has a tail* both say that each dog has a tail but the first could say that each dog has more than one tail try substituting teeth for tails. When an AI program manipulates the internal representation of facts these new representations should also be interpretable as new representations of facts.

Consider the classic problem of the mutilated chess board. Problem In a normal chess board the opposite corner squares have been eliminated. The given task is to cover all the squares on the remaining board by dominoes so that each domino covers two squares. No overlapping of dominoes is allowed, can it be done. Consider three data structures

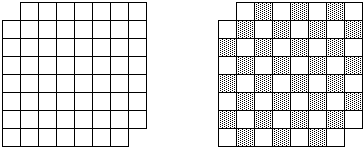


Fig. [3.1](http://www.cs.cf.ac.uk/Dave/AI2/node33.html#figchecker) Mutilated Checker

the first two are illustrated in the diagrams above and the third data structure is the number of black squares and the number of white squares. The first diagram loses the colour of the squares and a solution is not east to see; the second preserves the colours but produces no easier path whereas counting the number of squares of each colour giving black as 32 and the number of white as 30 yields an immediate solution of NO as a domino must be on one white square and one black square, thus the number of squares must be equal for a positive solution.

**Using Knowledge**

We have briefly mentioned where knowledge is used in AI systems. Let us consider a little further to what applications and how knowledge may be used.

**Learning**

-- acquiring knowledge. This is more than simply adding new facts to a knowledge base. New data may have to be *classified* prior to storage for easy *retrieval, etc.*. *Interaction* and *inference* with existing facts to avoid redundancy and replication in the knowledge and and also so that facts can be updated.

**Retrieval**

-- The representation scheme used can have a critical effect on the *efficiency* of the method. Humans are very good at it.

Many AI methods have tried to model human (see lecture on distributed reasoning)

**Reasoning**

-- Infer facts from existing data.

If a system on only knows:

* Miles Davis is a Jazz Musician.
* All Jazz Musicians can play their instruments well.

If things like *Is Miles Davis a Jazz Musician?* or *Can Jazz Musicians play their instruments well?* are asked then the answer is readily obtained from the data structures and procedures.

However a question like *Can Miles Davis play his instrument well?* requires reasoning.

The above are all related. For example, it is fairly obvious that learning and reasoning involve retrieval *etc*.

**Properties for Knowledge Representation Systems**

The following properties should be possessed by a knowledge representation system.

**Representational Adequacy**

-- the ability to represent the required knowledge;

**Inferential Adequacy**

- the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

**Inferential Efficiency**

- the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

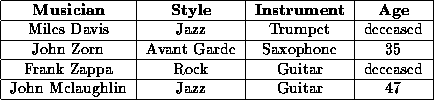
**Acquisitional Efficiency**

- the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

### Simple relational knowledge

The simplest way of storing facts is to use a relational method where each fact about a set of objects is set out systematically in columns. This representation gives little opportunity for inference, but it can be used as the knowledge basis for inference engines.

* Simple way to store facts.
* Each fact about a set of objects is set out systematically in columns (Fig. [7](http://www.cs.cf.ac.uk/Dave/AI2/node37.html#figrelate)).
* Little opportunity for inference.
* Knowledge basis for inference engines.

    
**Figure:** Simple Relational Knowledge

We can ask things like:

* Who is dead?
* Who plays Jazz/Trumpet etc.?

This sort of representation is popular in database systems.

### Inheritable knowledge

Relational knowledge is made up of objects consisting of

* attributes
* corresponding associated values.

We extend the base more by allowing inference mechanisms:

* Property inheritance
  + elements inherit values from being members of a class.
  + data must be organised into a hierarchy of classes (Fig. [8](http://www.cs.cf.ac.uk/Dave/AI2/node38.html#figinherit)).

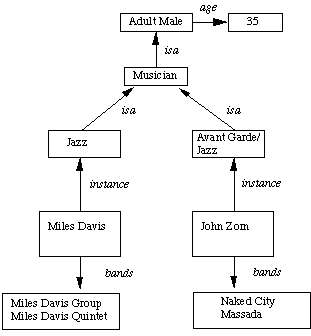


Fig. [8](http://www.cs.cf.ac.uk/Dave/AI2/node38.html#figinherit) Property Inheritance Hierarchy

* Boxed nodes -- objects and values of attributes of objects.
* Values can be objects with attributes and so on.
* Arrows -- point from object to its value.
* This structure is known as a slot and filler structure, semantic network or a collection of frames.

The algorithm to retrieve a value for an attribute of an instance object:

1. Find the object in the knowledge base
2. If there is a value for the attribute report it
3. Otherwise look for a value of instance if none fail
4. Otherwise go to that node and find a value for the attribute and then report it
5. Otherwise search through using isa until a value is found for the attribute.

**Inferential Knowledge**

Represent knowledge as *formal logic*:

*All dogs have tails* tex2html_wrap_inline7154 : *dog(x)* tex2html_wrap_inline7156 *hasatail(x)* Advantages:

* A set of strict rules.
  + Can be used to derive more facts.
  + Truths of new statements can be verified.
  + Guaranteed correctness.
* Many inference procedures available to in implement standard rules of logic.
* Popular in AI systems. *e.g* Automated theorem proving.

### Procedural Knowledge

Basic idea:

* Knowledge encoded in some procedures
  + small programs that know how to do specific things, how to proceed.
  + e.g a parser in a natural language understander has the knowledge that a noun phrase may contain articles, adjectives and nouns. It is represented by calls to routines that know how to process articles, adjectives and nouns.

Advantages:

* Heuristic or domain specific knowledge can be represented.
* Extended logical inferences, such as default reasoning facilitated.
* Side effects of actions may be modelled. Some rules may become false in time. Keeping track of this in large systems may be tricky.

Disadvantages:

* Completeness -- not all cases may be represented.
* Consistency -- not all deductions may be correct.

e.g If we know that Fred is a bird we might deduce that Fred can fly. Later we might discover that Fred is an emu.

* Modularity is sacrificed. Changes in knowledge base might have far-reaching effects.
* Cumbersome control information.

# *Logic Knowledge Representation*

*We briefly mentioned how logic can be used to represent simple facts in the last lecture. Here we will highlight major principles involved in knowledge representation. In particular*predicate logic*will be met in other knowledge representation schemes and reasoning methods.*

*A more comprehensive treatment is given in the third year*Expert Systems*course.****Symbols used****The following standard logic symbols we use in this course are:*

*For all tex2html_wrap_inline7176*

*There exists tex2html_wrap_inline7174*

*Implies tex2html_wrap_inline7156*

*Not tex2html_wrap_inline7182*

*Or tex2html_wrap_inline7184*

*And tex2html_wrap_inline7186*

### An example

Consider the following:

* Prince is a mega star.
* Mega stars are rich.
* Rich people have fast cars.
* Fast cars consume a lot of petrol.

and try to draw the conclusion: Prince's car consumes a lot of petrol.

So we can translate Prince is a mega star into: mega\_star(prince) and Mega stars are rich into: *tex2html_wrap_inline7176*m: mega\_star(m)*tex2html_wrap_inline7156*rich(m)

Rich people have fast cars, the third axiom is more difficult:

* Is cars a relation and therefore car(c,m) says that case *c* is m's car. **OR**
* Is cars a function? So we may have car\_of(m).

Assume cars is a relation then axiom 3 may be written: *tex2html_wrap_inline7176*c,m: car(c,m)*tex2html_wrap_inline7186*rich(m)*tex2html_wrap_inline7156*fast(c).

The fourth axiom is a general statement about fast cars. Let consume(c) mean that car *c* consumes a lot of petrol. Then we may write: *tex2html_wrap_inline7176*c:*tex2html_wrap_inline7204*fast(c)*tex2html_wrap_inline7206*m:car(c,m)*tex2html_wrap_inline7156*consume(c)*tex2html_wrap_inline7210*.

**Is this enough?** **NO!** -- Does prince have a car? We need the car\_of function after all (and addition to car): *tex2html_wrap_inline7176*c:car(car\_of(m),m). The result of applying car\_of to *m* is *m*'s car. The final set of predicates is: mega\_star(prince) *tex2html_wrap_inline7176*m: mega\_star(m)*tex2html_wrap_inline7156*rich(m) *tex2html_wrap_inline7176*c:car(car\_of(m),m). *tex2html_wrap_inline7176*c,m: car(c,m)*tex2html_wrap_inline7186*rich(m)*tex2html_wrap_inline7156*fast(c). *tex2html_wrap_inline7176*c:*tex2html_wrap_inline7204*fast(c)*tex2html_wrap_inline7206*m:car(c,m)*tex2html_wrap_inline7156*consume(c)*tex2html_wrap_inline7210*. Given this we could conclude:consume(car\_of(prince)).

### Isa and instance relationships

Two attributes isa and instance play an important role in many aspects of knowledge representation.

The reason for this is that they support property inheritance.

**isa**

-- used to show class inclusion, e.g. isa(mega\_star,rich).

**instance**

-- used to show class membership, e.g. instance(prince,mega\_star).

From the above it should be simple to see how to represent these in predicate logic.

**Applications and extensions**

* First order logic basically extends predicate calculus to allow:
  + functions -- return *objects* not just TRUE/FALSE.
  + equals predicate added.
* Problem solving and theorem proving -- large application areas.
* STRIPS robot planning system employs a first order logic system to enhance its means-ends analysis (GPS) planning. This amalgamation provided a very powerful heuristic search.
* Question answering systems.