Programming Descriptive Analogies By Example

Henry Lieberman

Visible Language Workshop
Media Laboratory
Massachusetts Institute of Technology

Cambridge, Mass. USA
Electronic mail [Internet]: Henry@AI.AI.MIT.Edu

Abstract

This paper describes a system for "programming by analogy", called *Likewise*. Using this new approach to interactive knowledge acquisition, a programmer presents specific examples and points out which aspects of the examples are "slippable" to more general situations. The system constructs a general rule which can then be applied to "analogous" examples. Given a new example, the system can then construct an analogy with the old example by trying to instantiate new descriptions which correspond to the descriptions constructed for the first example. If a new example doesn't fit an old concept exactly, a concept can be generalized or specialized incrementally to make the analogy go through. Midway between "programming by example" and inductive inference programs, Likewise attacks the more modest goal of being able to communicate to the computer an analogy which is already understood by a person. Its operation on a typical concept learning task is presented in detail.

A program that can be taught analogies is a step towards intelligent computing

We see the world through analogy-colored glasses. Analogy pervades every aspect of human thought, so computers will need to be able to perform analogies themselves if they are ever to make any progress in becoming intelligent. On the road to creating computers that can make analogies themselves lies the more modest goal of having a computer that can at least *understand* an analogy presented to it by a person. This follows McCarthy's famous dictum about "advice takers": Before a computer can figure out how to perform a task by itself, you have to at least be able to t*ell it how* to perform that task [McCarthy 68].

In the case of analogies, the test of whether a computer "understands" an analogy or not takes a specific form: Once the analogy has been presented, can the machine apply the analogy in new example situations where it is appropriate? Can the machine be made to generalize the analogy as the need to encompass new examples arises? This paper describes a proposal to create a new interactive computer environment for "programming by analogy", which I hope will allow a machine to understand analogies in this way. The system is not complete, but is currently being implemented on Symbolics 3600 Lisp Machines.

Foremost are the *user interface* issues involved in creating a system for analogical learning. These issues have been insufficiently addressed by previous AI work in analogy. We introduce in this paper a *graphical notation* for analogy, using icons to represent descriptions and links to represent relations, suitable for interactive manipulation by a graphical editor. These constitute a graphical semantic network browser [Barber 83], [Ciccarelli 85].

We are concerned with the cognitive tasks facing the user in interacting with an analogy system -- how is the user going to decide what to tell the system next? Much of analogical learning is incremental, so it is important to provide operations for incrementally adding examples, incrementally adding descriptions, or incrementally generalizing or specializing analogies. These incremental operations are more easily provided in a graphical editor than by interfaces requiring typing assertions textually in a logic-like notation. Immediate visual feedback must be provided by the interface for every incremental operation.

We will also consider *reasoning* issues raised by analogical thinking. Analogical inferences are not logically sound, so some non-deductive mechanism for making analogical inferences must be provided. We take the approach of harnessing *meta-level reasoning* to perform analogical inference.

This also brings up the issue of the nature of definitional knowledge. In most systems, definitions are considered fixed, immutable things, and most questions deal with what kind inferences can be drawn from a set of definitions. Analogical reasoning works the other way -- it is known what inferences should be made, and the issue is how to manipulate the definitions so that the inferences work out correctly. In short, ask not what you can do with your definitions, ask what your definitions can do for you.

An example of learning analogies by example

The methodology is best explained, of course, by considering an example of how it would work. This example is taken from [Hofstadter 85] and [Hofstadter et al. 80], where Hofstadter uses it as an example of how "definitions" of concepts can "slip" in the face of the need to apply them to new examples.

Let's say we would like to teach Likewise the concept of a "First Lady". We will introduce the concept using a particular example of a First Lady, namely Nancy Reagan. By presenting statements and descriptions about the specific case Nancy Reagan, we can teach the system to generalize to statements and descriptions about the concept of First Lady in general.

But there are many different possible ways of generalizing the example of Nancy Reagan. A simple way of defining a "First Lady" is to say that the First Lady is the wife of the President of the United States. What if we ask "Who is the First Lady of France?". The straightforward, literal, deductive answer is: Nobody, because France is not the United States. But analogical reasoning does permit a positive answer. The real question is: does anyone play a role with respect to France analogous to the role that Nancy Reagan plays with respect to the United States?

Does that mean that Nancy Reagan is not a First Lady because George Bush is now President? Could there be a "First Lady of IBM"? It all depends. Each interpretation can made plausible, depending on how far you want to "stretch" the analogy with the original example. Our aim is to make a system that lets a user communicate such analogies to a descriptive knowledge base, driven by the need to make inferences in concrete examples.

Omega is a language of descriptions, good for representing analogies

We now introduce our notation. We use the description system Omega [Attardi and Simi 81], which is an object-oriented semantic knowledge base of descriptions. Omega has

- *Individual descriptions*, names of specific constants such as `Nancy-Reagan`, which describes the woman who is the wife of the President.

- *Instance descriptions* describing instances of general concepts, such as `(a President)`.

- *Attributions* which describe some aspect of an instance description, for example, `(a President (with Country America) (with Home White-House))`.

- *A relation* `is`, which says that one description describes another. `(America is (a Country))` says that the individual concept `America` is a particular instance of the general concept `Country`. `((a First-Lady) is (a Woman))` says that every instance of the concept `First Lady` also fits the concept `Woman`.

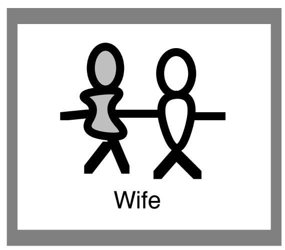
- *Variables*, denoted by `=`, such as `=Country`, and with restrictions `(=Country whichIs (not (a Communist-Country))`

- Logical *connectives*, `and`, `or`, `not`, etc.

Individual description

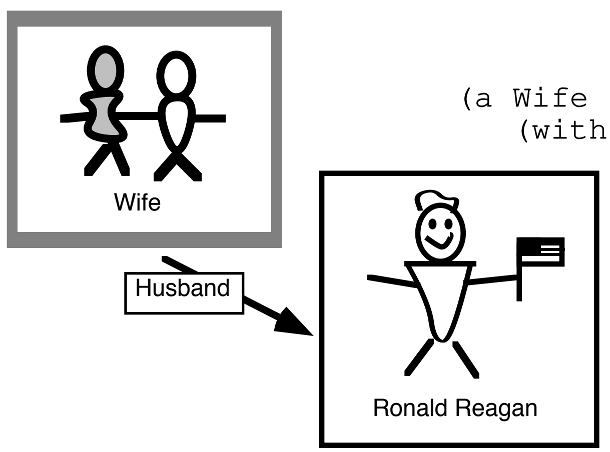

```
Nancy-Reagan
```

Instance description



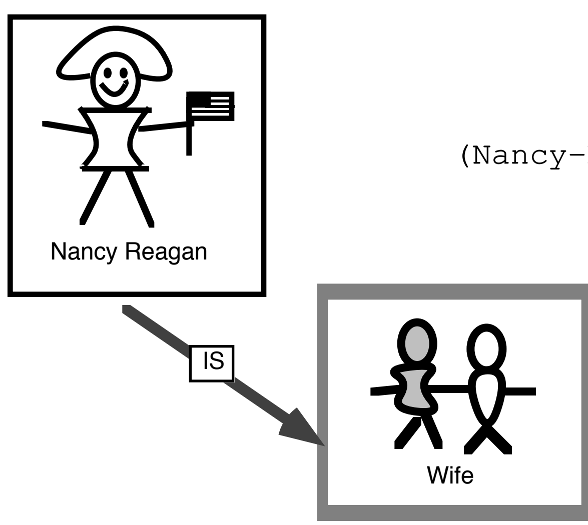

```
(a Wife)
```

Description with attribute




```
(a Wife
   (with Husband
         Ronald-Reagan))
```

Assertion




```
(Nancy-Reagan is (a Wife))
```

While Omega is especially well suited to our purposes here, it might be possible to construct analogous systems for other declarative, "frame-based" description languages, or a "logic programming" language like Prolog.

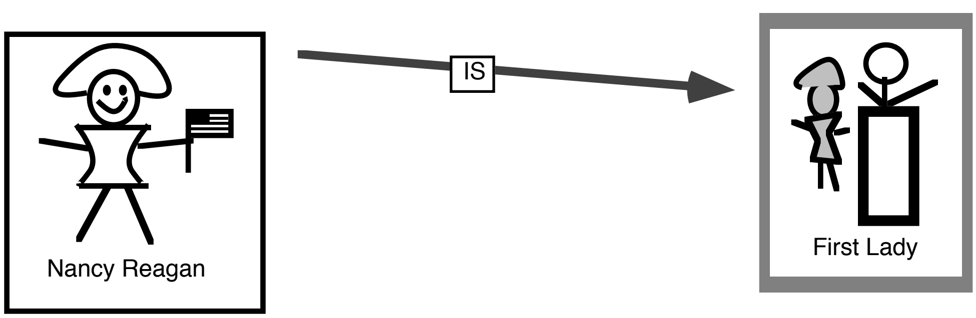
The concept of a First Lady is explained by describing Nancy Reagan

If you know the concept First Lady, try to answer this question quickly, without "thinking": *What is it* about Nancy Reagan that makes her a First Lady?

Before the "official definition" of a First Lady even comes to mind, the following answer might pop into your head: Nancy is a First Lady *because she's married to Ronald Reagan*. Whatever the eventual criterion for deciding whether someone is a First Lady, we will all agree that Nancy Reagan serves as a prototypical example.

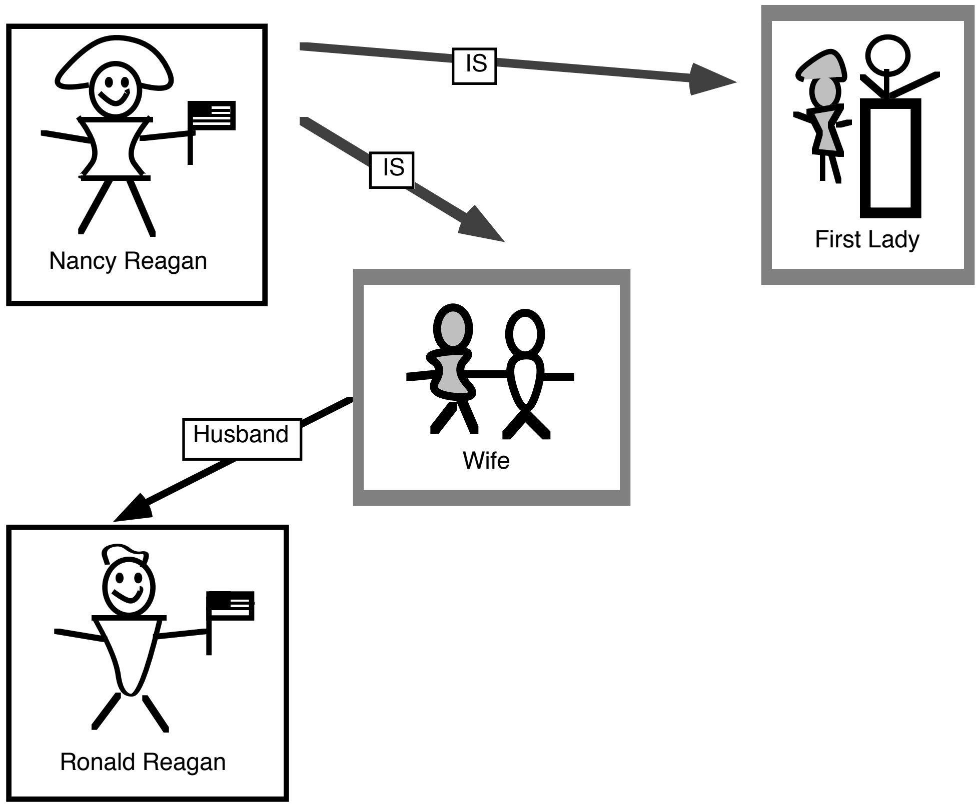
To generalize a concept from examples, we must have some way of determining what features of a particular situation are important in general, and which are just accidental properties of a particular example. Instead of requiring Likewise to guess what the user intended in presenting the Nancy Reagan example, we will ask the user to instruct the system on how to make generalizations, by pointing out what each description is an example of.

We begin simply by asserting that Nancy is a First Lady.




```
(Nancy-Reagan is (a First-Lady))
```

It is certainly also true that




```
(Nancy-Reagan is (a Wife (with Husband Ronald-Reagan)))
```

But there is something special about this second fact. The fact that Nancy is married to Ronald is the reason *why* Nancy is the First Lady.

Analogical reasoning is a meta-level reasoning technique

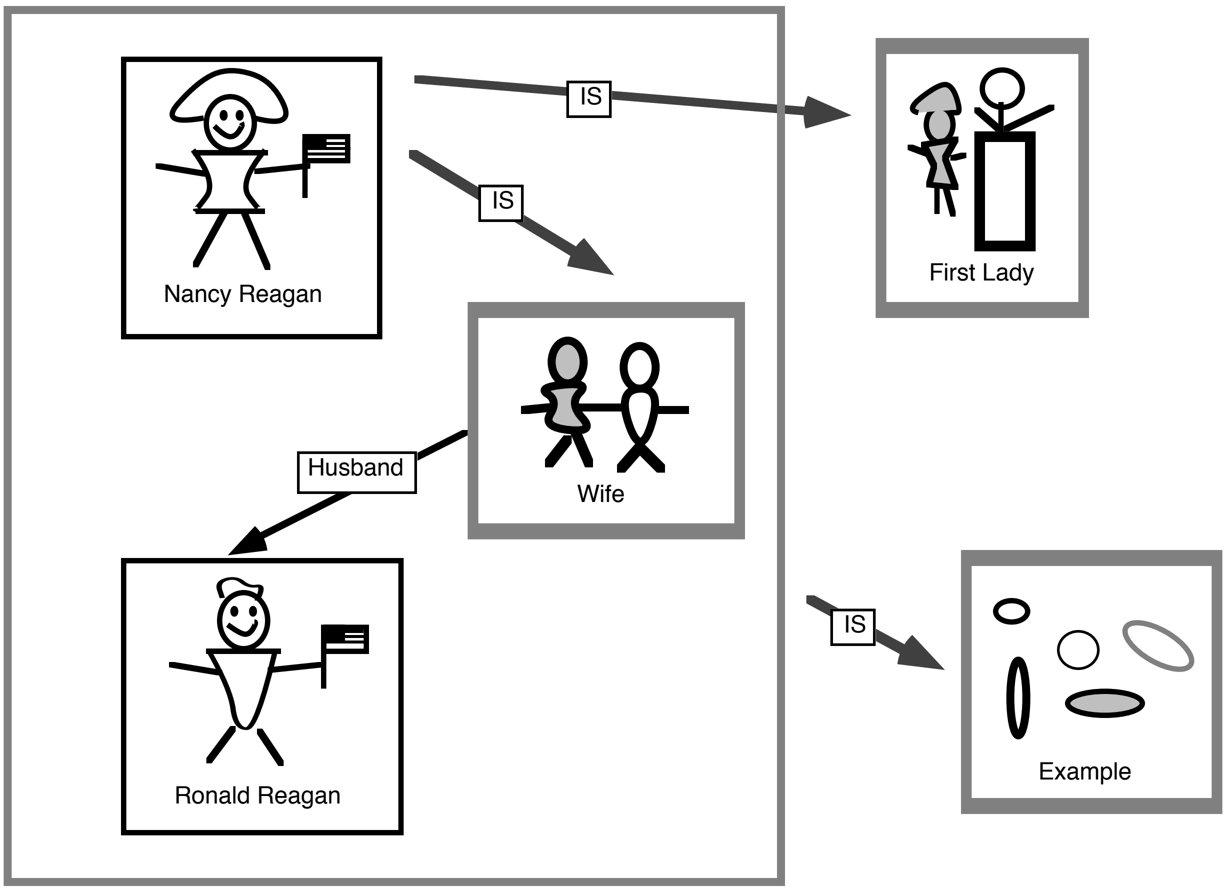
Most knowledge representation systems can represent specific facts, using assertions about individual constants. They can also represent more abstract statements by rules containing variables which can be replaced by individuals in specific cases. But few systems currently represent the *relationship* between abstract descriptions and concrete individuals. For reasoning by analogy, the representation of the relationship between concrete examples and abstract descriptions is essential.

In order to be able to reason about examples, we introduce explicitly the concept of an `Example` as an Omega description. Rules which perform analogical inference can be driven by exemplification relationships.

The key idea is implementing the reasoning mechanism for analogy is that being an example is a special kind of meta-level reasoning construct. An example is just like an ordinary fact, but it has special importance. It can activate heuristic rules that are not logically valid, but "jump to conclusions" that are pragmatically useful.

In Omega, we state an ordinary fact using the `is` statement. This says that one description describes another description. The ^ notation is a meta-level "quotation" device. It allows us to attach a description to the *statement* itself, making a "statement about a statement". Descriptions that describe statements are meta-level descriptions.

We introduce an explicit description of this reason, by describing the `is` statement concerning the `Wife` relation as an example. This is a meta-level assertion that reads as "The statement that '*Nancy Reagan is a wife with husband Ronald Reagan*' is an example". Graphically, we encase the whole is statement in the same kind of box that surrounds a description.



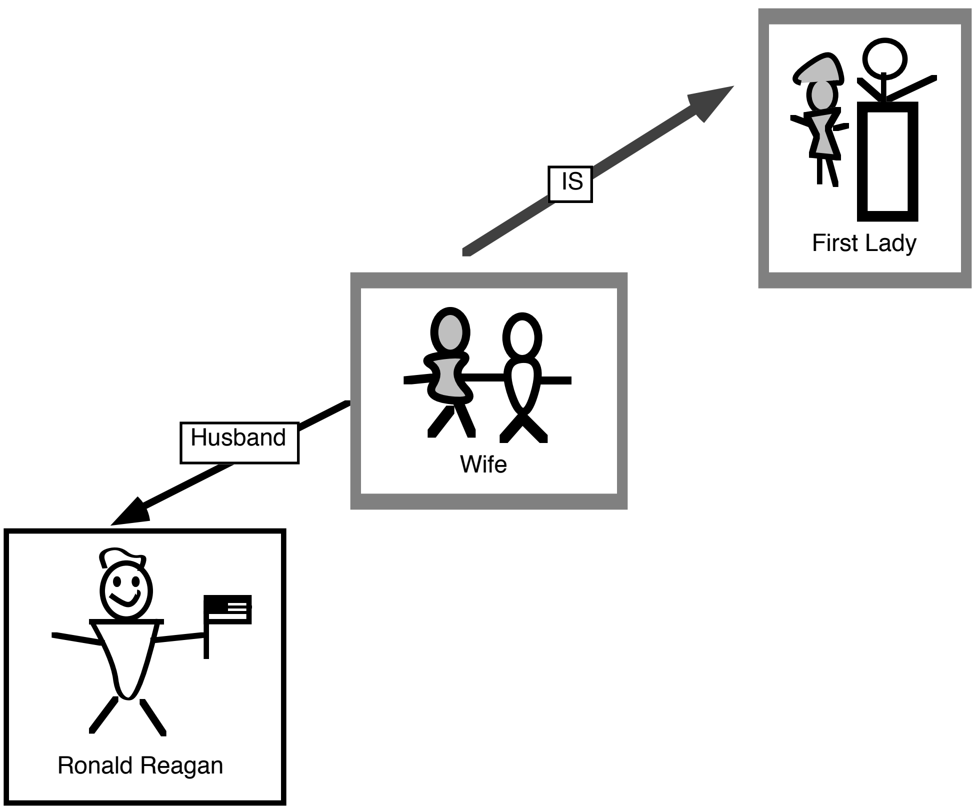

```
(^(Nancy-Reagan is (a Wife (with Husband Ronald-Reagan)))
 is
 (an Example))
```

Because Nancy is being used in this context as an example of a certain kind of wife, anything said about Nancy specifically is likely to be illustrative of that kind of wife in general. The *Concept Analogy Rule* says roughly that "If something is an example of a more general concept, than whatever is true of the example illustrates a truth about the general concept as well".

The Concept Analogy Rule:

```
(((^(=Example is =Generalization) is (an Example))
  and
  (=Example is =Example-Description))
 implies
 (=Generalization is =Example-Description))
```

In the case of Nancy Reagan, the `Generalization` is her description as a wife, and the `Example-Description` is the characterization as a First Lady. In this case, the Concept Analogy Rule means that the system can automatically make a generalization. Here it is:




```
((a Wife (with Husband Ronald-Reagan)) is (a First-Lady))
```

This says, "Any wife whose husband is Ronald Reagan is a First Lady". True enough. If Reagan had two wives [imagine that!], his other wife would be a First Lady, too. While this is correct, strictly speaking, it is not very useful, and certainly not the generalization we wanted the system to learn.

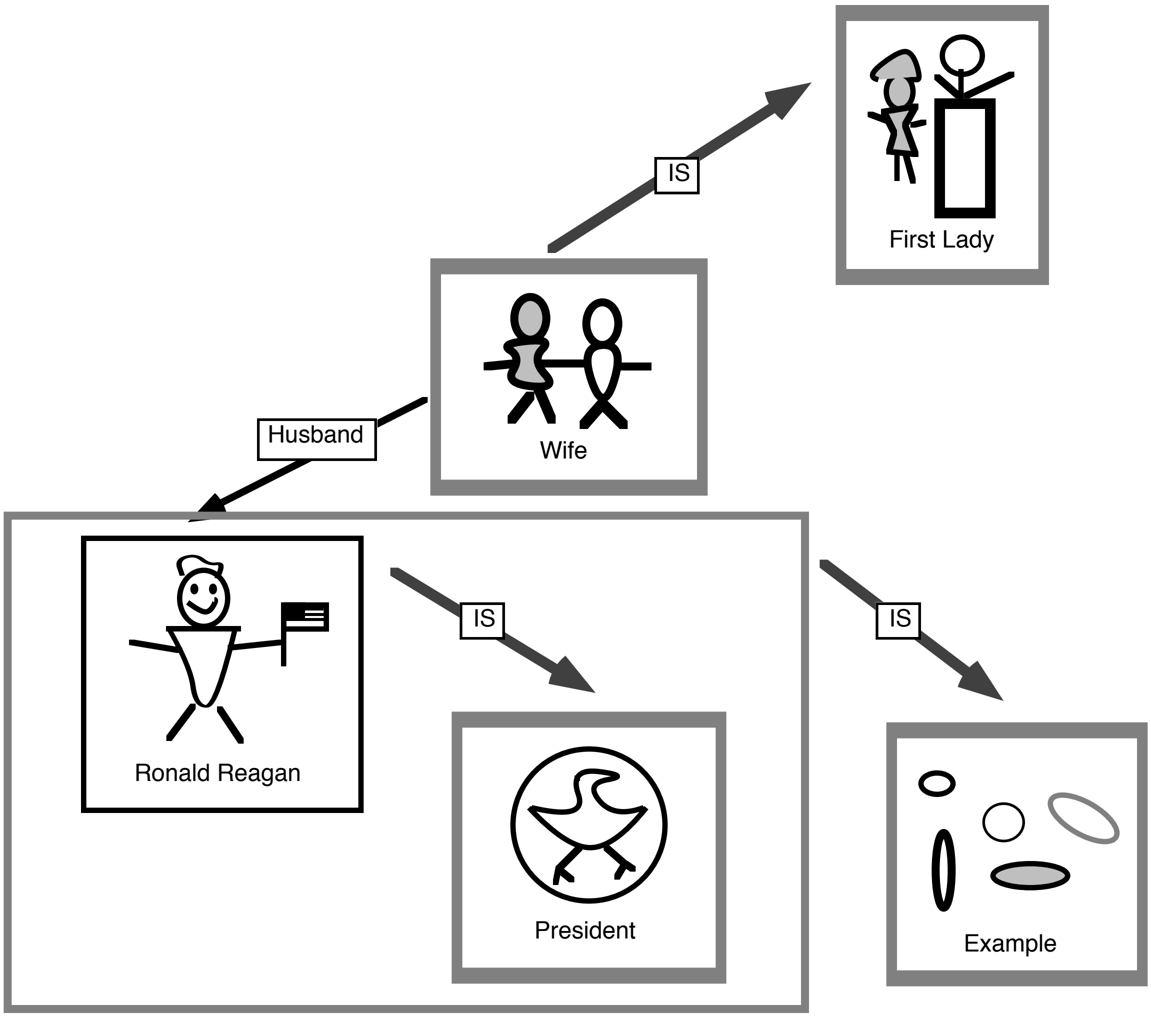
The problem is that it isn't *only* the fact that Nancy is married to Ronald that makes her the First Lady, but also the fact that Ronald is President. Nancy's role here is only as an *example* of a First Lady. We could just as well have chosen Martha Washington or Jackie Kennedy. Just as Nancy is only an example of a First Lady, Ronald is only an example of a President, and had we chosen Martha Washington as our example, George Washington would have played the role that Ronald plays. We want to remove the specific fact that it is Ronald who is the President from consideration.

This calls for another generalization, from `Ronald` to `(a President)`. If the system already has a knowledge base about Ronald, it can display the possible generalizations of Ronald, so that if one of them is appropriate, we can just choose it, rather than having to re-input it. We may know that `(Ronald-Reagan is (a Movie-Actor))`, `(Ronald-Reagan is (a Rancher))`, etc. but by marking the relationship to `(a President)` as `(an Example)`, we are singling out that one as important for this application.

Analogies can be made about attributes as well as concepts

So far, the Concept Analogy Rule allows us to make analogies only from examples that appear as the first argument to an `is` statement. But what about objects that, like Ronald Reagan in our example, appear as the value of attributes of descriptions?

To generalize on these, we need an Attribute Analogy Rule [analogous to the Concept Analogy Rule]. This rule will let us conclude from the meta-level statement




```
(^(Ronald-Reagan is (a President)) is (an Example))
```

that Ronald can be abstracted out of the corresponding statement about First Ladies.

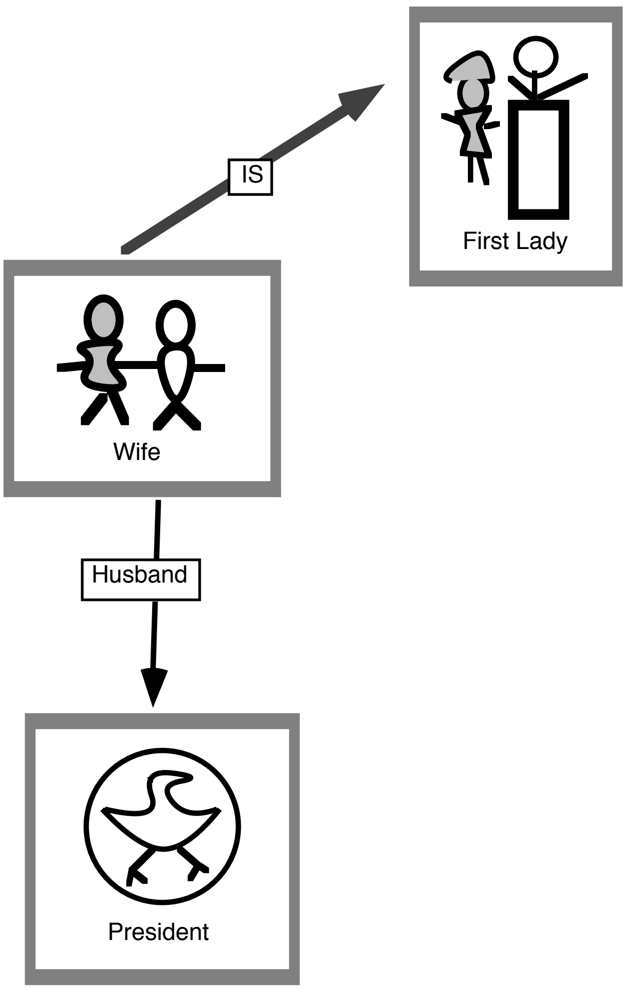
The Attribute Analogy Rule says "If an example can be generalized, descriptions containing that example as an attribute can also be generalized".

The Attribute Analogy Rule:

```
(^(=Example is =Generalization) is (an Example))
 implies
 (^((a =Concept (with =Attribute =Example))
    is
    (a =Concept (with =Attribute =Generalization)))
  is
```

```
    (an Example)))
```

In the case of Ronald Reagan, the meta-statement that he exemplifies a President means that we can generalize from the wife of Ronald to the wife of a President, using the Attribute Analogy Rule.




```
((a Wife (with Husband (a President)) is (a First-Lady))
```

or, "Anyone who is the wife of a President is a First Lady", which, I'm sure you'll agree, is much better.

Tying the notion of First Lady to a particular country makes it more specific

We can even sharpen the notion of First Lady by noting that there is only one First Lady of any country, so maybe a First Lady should be associated with a particular country.

```
(Nancy-Reagan is (a First-Lady (with Country America)))
```

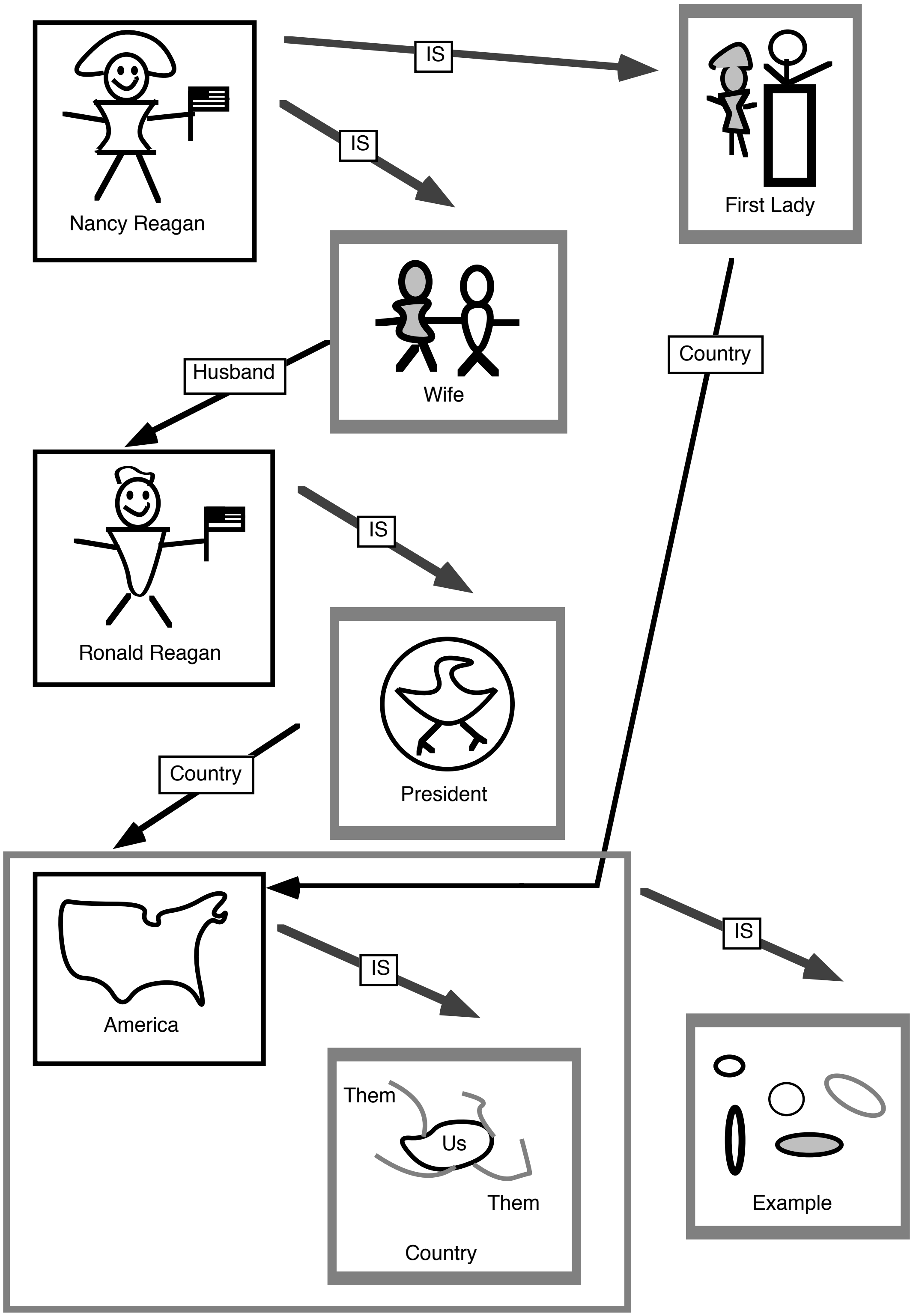
But it wouldn't be right to say that

```
(Nancy-Reagan is (a First-Lady (with Country Russia)))
```

so we had better make sure that the country that the husband is President of is the same country that the wife is the First Lady of. Connecting them both to the same description `America` represents a constraint that the values of those attributes be identical.

```
(^(Ronald-Reagan is (a President (with Country America)))
 is
 (an Example))
```

The above formulation only allows us to have American First Ladies. By saying that America is an example of a country, we can bring over the notion of First Lady to other countries. Again, we have to abstract out the constant `America`, since there's nothing important in this example about America, except that it illustrates the idea of associating a First Lady with a country.


Nancy Reagan
IS
IS
First Lady
Wife
Husband
Country
Ronald Reagan
IS
President
Country
America
IS
IS
Them
Us
Them
Country
Example

```
(^(America is (a Country)) is (an Example))
```

This generalization requires two applications of the Attribute Analogy Rule.

```
(^((a President (with Country America))
   is
   (a President (with Country (a Country))))
 is
 (an Example))

(^((a Wife (with Husband (a President (with Country America))))
   is
   (a Wife (with Husband (a President (with Country (a Country))))
 is
 (an Example))
```

And now the Concept Analogy Rule applies.

```
((a Wife (with Husband
               (a President (with Country
                                  (=C whichIs (a Country))))))
 is
 (a First-Lady (with Country =C)))
```

or, "A wife whose husband is the President of a country is the First Lady of that country".

Analogies between countries lead to analogies between First Ladies

Once we have learned a definition of First Lady, this knowledge can now be applied to another example. Introducing a precise technical notion of analogy, we will say that two descriptions are *analogous* when both descriptions are examples of the same concept, and all their attributions are analogous. In this sense, then,

```
(France is-analogous-to America)
```

because they are both examples of `(a Country)`, that is

```
(((^France is (a Country)) is (an Example))
 and
 ((^America is (a Country)) is (an Example)))
```

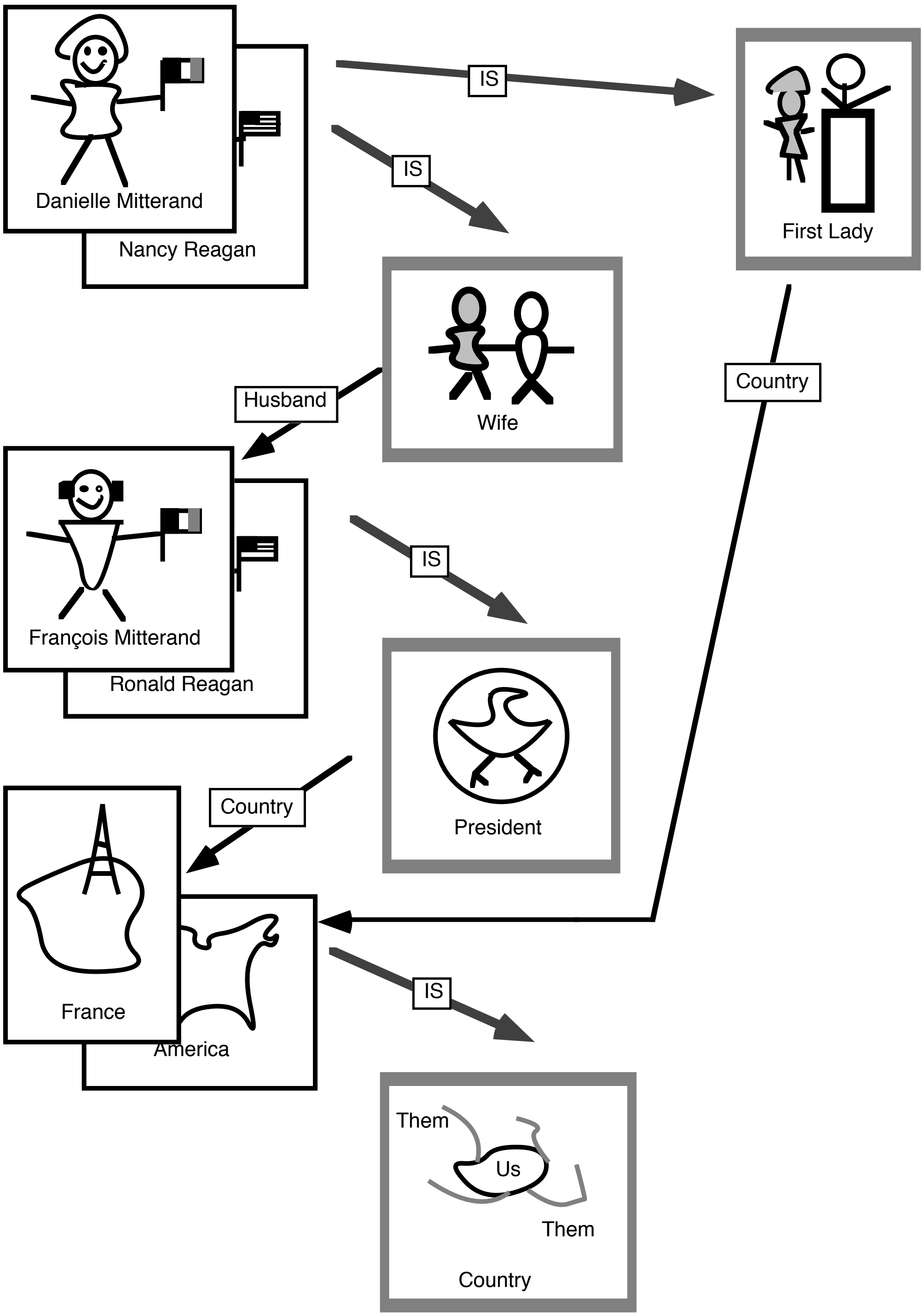
We introduce the description `France`, placing it a position on the screen near the displayed representation of `America`.

Likewise now tries to instantiate a description network for France which is isomorphic to the description network involving America and Nancy Reagan. For every individual description in the first network, Likewise tries either to find in the knowledge base, or failing that, asks the user to provide, descriptions which fill analogous roles. For example, since Ronald Reagan is the President of America, we need an analogous description for the President of France.

Given the knowledge that

```
(Danielle-Mitterand is (a Wife (with Husband Francois-Mitterand)))
(Francois-Mitterand is (a President (with Country France)))
```

the description system can automatically conclude


Danielle Mitterand
Nancy Reagan
IS
First Lady
IS
Wife
Husband
Country
François Mitterand
Ronald Reagan
IS
President
Country
France
America
IS
Them
Us
Them
Country

```
(Danielle-Mitterand is (a First-Lady (with Country France)))
```

by analogy with Nancy Reagan.

Adding new examples may require generalizing concepts

The driving force for creation and modification of concepts is the need to understand new examples by making analogies with older examples. Often, one example is not sufficient to illustrate a concept. We may want a concept to apply to any one of several examples, and it must be general enough to capture what all the examples have in common.

People tend to generalize concepts incrementally from examples. Starting with one example, certain features are abstracted out of the example to create a general concept. Then, another example is encountered which may not fit in the original version of the concept, but is similar enough to the first that an *analogy* can be made between the two examples.

It may be necessary to *slip* some aspects of the original definition in order to get it to fit the new example, while continuing to apply equally well to the old example. This slipping often involves making concepts more general or abstract, travelling higher in the inheritance hierarchy, so that they can apply to a larger number of specific cases.

Let's now consider making an analogy between America and England, asking "Who is the First Lady of England?". The idea is that we will use Likewise and Omega to set up a description network for England whose structure parallels the one for America. But it won't be exactly parallel -- the differences between the two situations will force some reconceptualization of what it means to be a First Lady.

[Of course, it isn't really correct here to say "England", because the independent country we are talking about is really named "The United Kingdom of Great Britain [England, Scotland, and Wales] and Northern Ireland". South Americans are annoyed when North Americans refer to the United States of America simply as "America". But that statements about "England" or "America" are understandable despite the lack of precision is precisely the point. Analogical understanding always picks the closest "reasonable" concept.]

Looking back at the description network we have set up for `America`, we notice there are three roles that the description `America` plays.

° As an example of `(a Country)`.

° As the `Country` of a `First-Lady`.

° As the `Country` of `(a President)`.

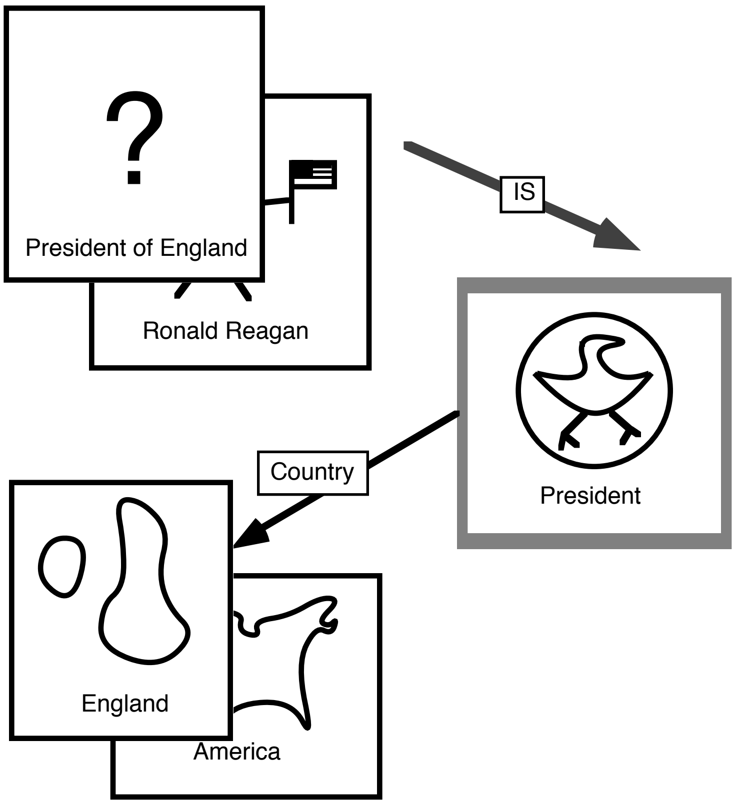
Likewise can make an attempt to automatically instantiate the same roles for England as we set up for America. That `England` is an example of `(a Country)` is trivial.

The description

```
(a First-Lady (with Country England))
```

gives England the role of being the `Country` of `(a First-Lady)`.

Now, all we need is




```
(a President (with Country England))
```

When analogies don't fit exactly, try to generalize

Oops -- England doesn't have a President at all! If the knowledge base already has enough knowledge about the government of England, it can detect this as a contradiction. Otherwise, the user can notice the inconsistency and point it out to the system.

A Prime Minister may not be a President, but a Prime Minister is *almost* a President, if we let the concepts slip into each other. A Prime Minister is *analogous* to a President in our example just like England is analogous to America. We can make these two descriptions analogous by finding what they share in common.

We already know that America and England both exemplify the concept `(a Country)`, so they are analogous. The question now becomes, "What concept generalizes `(a President)` and `(a Prime-Minister)` in the same way as `(a Country)` generalizes `America` and `England`"?

If England and America are analogous because they are both examples of countries, a Prime Minister and a President are analogous because they are both examples of rulers or leaders of analogous countries.

```
(^((a President) is (a Leader)) is (an Example))

(^((a Prime-Minister) is (a Leader)) is (an Example))
```

Likewise now infers

```
((a President) is-analogous-to (a Prime-Minister))
```

Our previous statement about first ladies is now generalized to

```
((a Wife (with Husband (a Leader)))
 is
 (a First-Lady))
```

[along with the corresponding statement involving naming the country].

Since

```
(^(Ronald-Reagan is (a Leader (with Country America)))
 is
 (an Example))
```

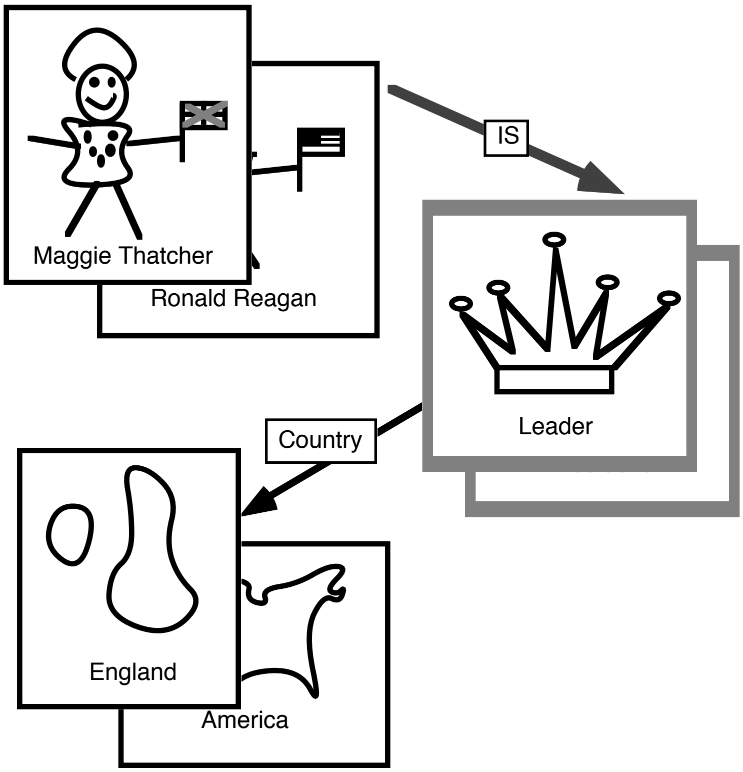
who is analogous to Ronald? Likewise needs somebody to play Ronald's role in the case of England. If we happen to have facts about English government, we know that

```
(Maggie-Thatcher is (a Prime-Minister (with Country England)))
```

and since

```
(^((a Prime-Minister) is (a Leader)) is (an Example))
```

it's true that




```
(Maggie-Thatcher is (a Leader (with Country England)))
```

Since England is already analogous to America and Prime Ministers to Presidents,

```
(^(Maggie-Thatcher is (a Leader (with Country England)))
 is
 (an Example))

(Ronald-Reagan is-analogous-to Maggie-Thatcher)
```

Likewise now tries to start filling in analogies for the roles played by Ronald Reagan in the case of Maggie Thatcher. One role of Nancy is as the wife of Ronald, so Likewise tries to create a description for

```
(a Wife (with Husband Maggie-Thatcher))
```

and fails, because Maggie is a woman and can't be a husband.

Finding Maggie Thatcher's "wife" requires generalization

Like the case where we had to find a way to think of a Prime Minister as a President, we have to find a sense in which Maggie can be thought of as a "husband" to perform the analogy. We need to find a generalization of the concept `Husband` such that it applies to Maggie as well as Ronald.

If we want the system to recognize that Maggie can't be a husband, we have to tell it the facts of life.

```
((a Wife) is (a Woman))
((a Husband) is (a Man))

((a Woman) is (a Person))
((a Man) is (a Person))
((a Woman) is (not (a Man)))
((a Man) is (not (a Woman)))
```

Since we already know that `Person` is a generalization of the concepts `Man` and `Woman`, we can create a new concept `Spouse` which, by analogy, generalizes the concepts `Husband` and `Wife`. The variables `=Hubby` and `=Sweetie` capture the fact that both husband and wife are partners in a marriage.

```
(^((a Wife (with Husband =Hubby))
   is
   (a Spouse (with Partner =Hubby)))
 is
 (an Example))

(^((a Husband (with Wife =Sweetie))
   is
   (a Spouse (with Partner =Sweetie)))
 is
 (an Example))
```

```
(Nancy-Reagan is (a Spouse (with Partner Ronald-Reagan)))
```

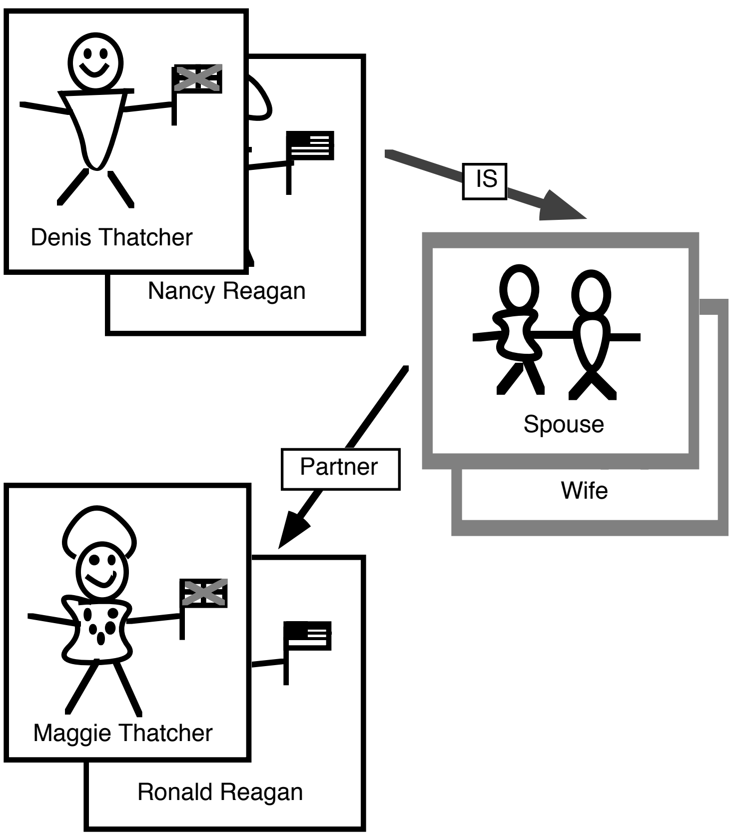
leads Likewise to look for the analogous description

```
(a Spouse (with Partner Maggie-Thatcher))
```

which has a higher probability of success than looking for Maggie's wife.

The First Lady of England is Denis Thatcher

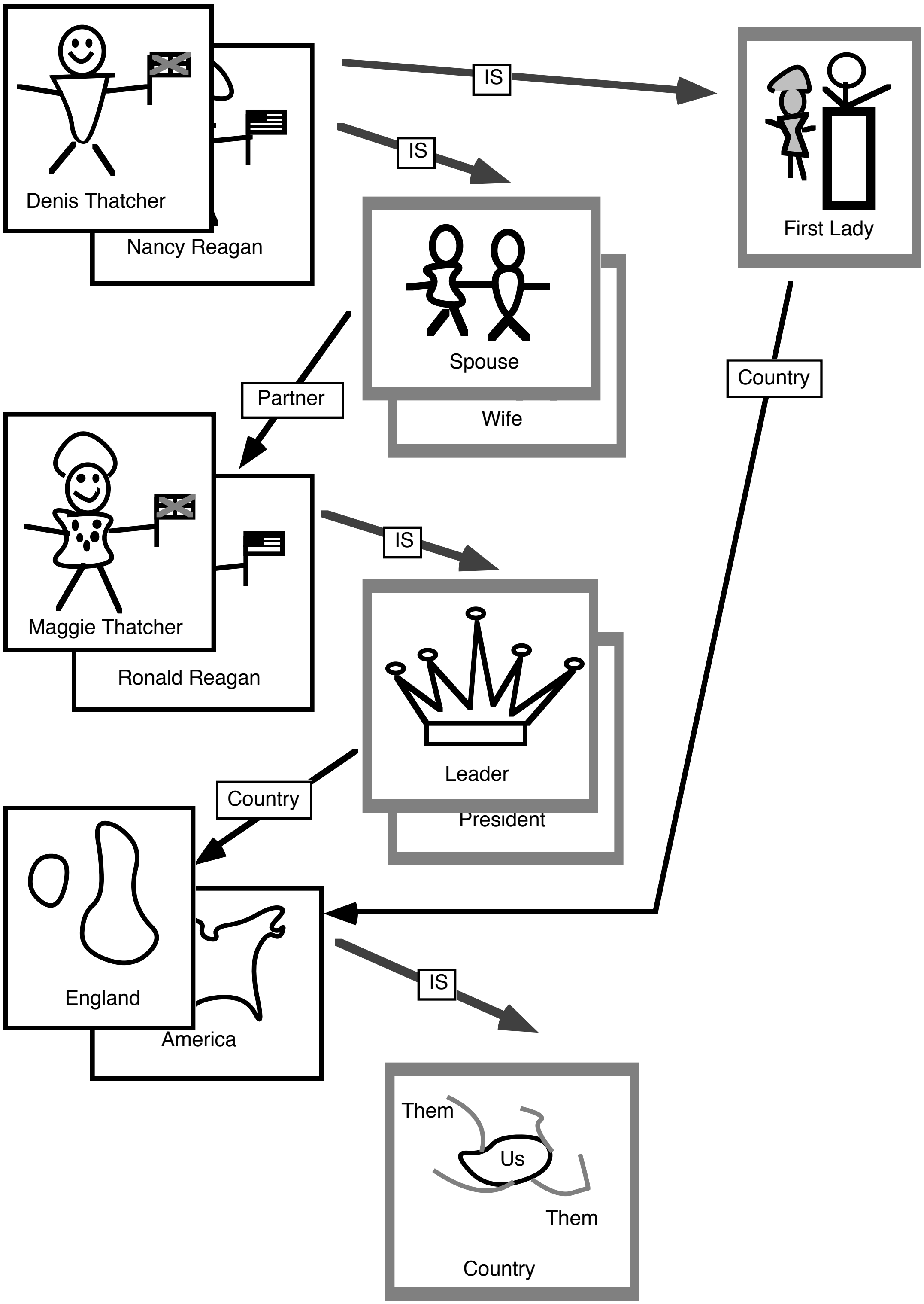
Either we retrieve or supply the information




```
(Denis-Thatcher is (a Husband (with Wife Maggie-Thatcher)))
```

instantiating Denis as Maggie's spouse.

Now that we have discovered Denis Thatcher, the final link in the network has been found, and we are finally left with



(Denis-Thatcher is (a First-Lady (with Country England)))

```
(Denis-Thatcher is-analogous-to Nancy-Reagan)
```

As a result, Likewise has generalized the definition of First Lady to read

```
((a Spouse (with Partner
                 (a Leader (with Country
                                 (=C whichIs (a Country))))))
 is
 (a First-Lady (with Country =C)))
```

or, "A First Lady is a spouse of a Leader of a country". This is our generalized notion of First Lady.

There's more than one way to choose a First Lady

This is, of course not the only possible answer to the question "Who is the First Lady of England". We could have chosen Queen Elizabeth, leading to a generalization of "highest-ranking woman" without violating the femaleness of the First Lady concept. Or perhaps Lady Diana. Each of these would have a unique, though plausible explanation. The point is not so much to come up with the "correct" answer in any sense, but to have a means of conveying whatever conceptualization the user desires.

One possible way to deal with multiple generalizations and multiple definitions is to use a "*truth maintenance*" or *reason maintenance* system. We can tag each generalization inferred by the system with the exemplification assumption `(=Example is (an Example))` that gave rise to it. For example, the reason for assuming that Ronald is an example of a President is to infer a First Lady definition that is not dependent upon Ronald. Then, if we wanted to change the generalization inferred, it could be traced back to one or more responsible exemplification assertions.

Negative examples serve to restrict concepts

In addition to the technique illustrated above of using positive examples to force generalization of concepts, the opposite technique is also important -- using *negative* examples to force *specialization* of concepts. If a concept is found to be too general, an example can be given of some situation which does not fit the description, and the notion can be adjusted down the inheritance hierarchy
until it excludes the exceptional case while still encompassing all the positive examples [Mitchell 83].

If we wanted to restrict the notion of First Lady to mean only *current* holders of that title, we could give Nancy Reagan as a negative example, and Barbara Bush as a positive example. A negative example fits the "wife of President" description but is excluded from the notion First Lady. The system would notice this contradiction and try to resolve it by finding or requesting a description that specialized the First Lady notion to be consistent with the set of examples.

Beyond Likewise: programs which make analogies automatically

Again, I should stress that Likewise does not *automatically* create generalizations; it is *not* an "inductive inference" program or an "automatic programmer". It cannot, by itself, induce descriptions in the manner of Winston's learning program [Winston 70]. Inductive inference capability would relieve the user from the burden of pointing out many trivial analogies which seem "obvious" to people. But the problem is complicated by the fact that is rarely only one "best" analogy. Heuristics like choosing the most specific generalization are at best only guesses. This is why we have chosen to focus on the *process* of analogy construction and the exploration of alternatives rather than finding the "right" answer.

Likewise can also be viewed as a step towards an *interface* for programs which do perform automatic inductive generalization. Programs in the vein of Winston's and Mitchell's learners still need an effective way of communicating with human users about complex knowledge domains. The best way to hook up an inductive inference program is to bestow it with the ability to automatically designate `is` statements as examples as needed to make analogies go through.

A plausible near-term scenario is to have a system that can work in both a "manual" mode, as described in this paper, or an "automatic" mode through the use of automatic generalization programs. The system could display the result of its inferences, and the user could have the opportunity to edit out any unwanted generalizations, or choose from a set of possibilities if the program could not make up its mind.

Likewise should be very useful in aiding a programmer in creating and debugging descriptive knowledge bases, because it accepts information in the form in which people like to provide it -- by making analogies based on specific examples. Likewise combines the process of stating abstract rules and testing them in specific cases into one activity, providing immediate, incremental visual feedback.

Acknowledgments

This research has been supported by a grant from Hewlett-Packard. The Visible Language Workshop is supported in part by additional grants from NYNEX, DARPA, IBM, and Xerox.

Maria Simi and Giuseppe Attardi were very helpful in formulating the Concept and Attribute Analogy Rules in Omega. Douglas Hofstadter was instrumental in starting me to think about the issues discussed here, and supplied the "First Lady" example. I would like to thank Jon Amsterdam, Laura Gould, Carl Hewitt, Douglas Hofstadter, Kenneth Kahn, David Rogers, and Luc Steels for their helpful comments on previous versions of this paper.

Bibliography

[Attardi and Simi 86] Giuseppe Attardi and Maria Simi. A Description Oriented Logic for Building Knowledge Bases. Proceedings of the IEEE Special Issue on Knowledge Representation, Vol. 74, No. 10, October 1986.

[Attardi and Simi 81] Giuseppe Attardi and Maria Simi. Semantics of Inheritance and Attributions in the Description System Omega. Proceedings of IJACI 81, Vancouver, B.C., Canada, August, 1981.

[Barber 83] Gerald Barber. Supporting Organizational Problem Solving with a Workstation. *ACM Transactions on Office Information Systems* (July 1983).

[Ciccarelli 85] Eugene C. Ciccarelli. Presentation Based User Interfaces. Ph.D. Th., Massachusetts Institute of Technology, 1985.

[Gentner 80] Dedre Gentner. The Structure of Analogical Models in Science. Tech. rep. 4451, Bolt, Beranek and Newman, July, 1980.

[Hofstadter 82] Douglas Hofstadter. Artificial Intelligence: Subcognition as Computation. Technical Report 132, Indiana University, November, 1982.

[Hofstadter 85] Douglas Hofstadter. *MetaMagical Themas.* Basic Books, 1985.

[Hofstadter 80] Douglas Hofstadter, Gray Clossman, Marsha Meredith. Shakespeare's Plays Weren't Written By Him, But By Someone Else With The Same Name. Technical Report 96, Indiana University, July, 1980.

[Iba 79] Glenn Iba. Learning Disjunctive Concepts From Examples. AI Memo 548, MIT Artificial Intelligence Laboratory, September 1979.

[Lieberman 82] Henry Lieberman. Designing Interactive Systems From the User's Viewpoint. In *Integrated Interactive Computer Systems*, P. Degano and E. Sandewall, Ed., North Holland, 1982.

[Lieberman 82] Henry Lieberman. Constructing Graphical User Interfaces by Example. Graphics Interface Conference, Toronto, Ontario, Canada, May, 1982.

[Lieberman 84] Henry Lieberman. Seeing What Your Programs Are Doing. *International Journal of Man-Machine Studies 21*, 4 (October 1984).

[McCarthy 68] John McCarthy. Computer Programs With Common Sense. In *Semantic Information Processing*, Marvin Minsky, Ed., MIT Press, 1968.

[Mitchell 83] Tom Mitchell, Paul Utgoff, Ranan Bannerji. Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics. In *Maching Learning*, R. Michalski, J. Carbonell, T. Mitchell, Ed., Tioga, 1983.

[Winston 70] Patrick H. Winston. Learning Structural Descriptions from Examples. Ph.D. Th., Massachusetts Institute of Technology, 1970.