> *I am an enthusiast, but not a crank in the sense that I have some pet theories as to the proper construction of a flying machine. I wish to avail myself of all that is already known and then, if possible, add my mite to help on the future worker who will attain final success.*
>
> —Wilbur Wright

Genifer 3.0 white paper

YKY (甄景贤)

May 18, 2015

Abstract

Introduces the Genifer3 logic engine.

1 Background

Genifer3 is "neo-classical", logic-based AI similar to OpenCog and NARS. Genifer4 attempts to transition logic to a vector space setting and employ continuous iterative methods, but its details have not been worked out yet.

Propositional logic concerns with propositions ("sentences") which can be assigned *truth values* and which have no *internal structure*. For example the formula $P \wedge Q$. Propositional logic is isomorphic to Boolean algebra. Its proof problem is called SAT (**satisfiability**) and is famously NP-complete. **Resolution** is a proof algorithm for propositional logic.

Predicate logic gives formulas internal structure via predicates such as `loves(john, mary)`. Predicate logic allows to have patterns such as `loves(X, Y)` where X and Y are variables. Logic rules containing patterns are matched to facts via the **unification** algorithm; This is known as *pattern-matching*. For example, this rule defines “grandfather”:

`grandfather(X,Z) ← father(X,Y) ∧ father(Y,Z)`

The proof procedure for predicate logic combines **unification** with **resolution**. This procedure is undecidable in the worst case — it follows because first-order logic is Turing universal and the *halting problem* dictates that such a proof procedure must be undecidable.

Genifer3 is different from classical logic in the following ways:

- fuzzy-probabilistic truth values
- elimination of variables

Fuzzy-probabilistic inference is very straightforward and routine, but it requires the Bayesian **belief propagation** algorithm to satisfy probability laws.

Eliminating variables in predicate logic is desirable because the use of variables is *unnatural* in human reasoning. Humans think of “grandfather” as the father of father (expressed as $f \circ f$ in relation algebra), instead of using variables as in `grandfather(X, Z)`. Also, the use of variables requires complicated *substitution management* during proof (this is explained in the classic text *Structure and Interpretation of Computer Programs*, but we don't need to bother). The following diagrams illustrate the linkage of variables within formulas in predicate logic:

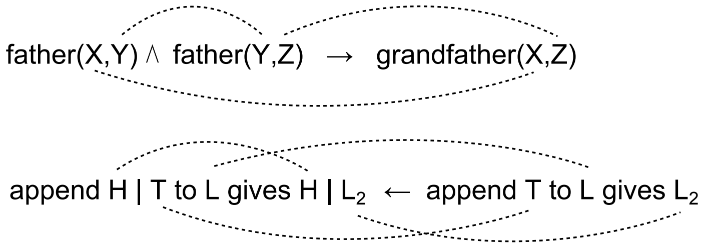
father(X,Y) ∧ father(Y,Z) → grandfather(X,Z)
append H | T to L gives H | L2 ← append T to L gives L2

Figure 1: Linkage of variables in predicate logic: the grandfather rule and the recursive definition of APPEND in PROLOG.

Combinatory logic is an attempt to eliminate all uses of variable binding in logic; **Relation algebra** is a restriction of combinatory logic to relational operations. Genifer3 uses a simplified form of relation algebra.

The bottleneck of AI is in the **learning algorithm**, which will be addressed below.

2 Logic

A **formula** in Genifer3 is simply a concatenation of atomic **concepts**:

$c_1 \cdot c_2 \cdot \ldots \cdot c_n$.

For example: `john · loves · mary`.

Since there are no variables in Genifer, *generalization* is achieved via the **subsumption** relation $\subseteq$, for example:

`cats` $\subseteq$ `animals`.

This allows to deduce, for example, from `humans are mortal` to `socrates is mortal`.

Logic formulas can be **facts** or **rules**; The difference is that rules are of the form:

`pre-condition` $\rightarrow$ `post-condition`.

As an example consider how one can deduce the new fact:

`john · grandfather = paul` (john's grandfather is paul)

from the given facts:

```
john · father = pete
pete · father = paul.
```

This may be achieved via the rule $father \circ father = grandfather$, and the substitution of terms by their equals, but I'm not sure if the substitution operation should or could be reduced to some other logic primitives.

3 Actions

We have to pay the price of eliminating logic variables; One way to do that is via the use of "memory registers".

Recall that a Turing machine consists of a tape memory and a set of finite states:

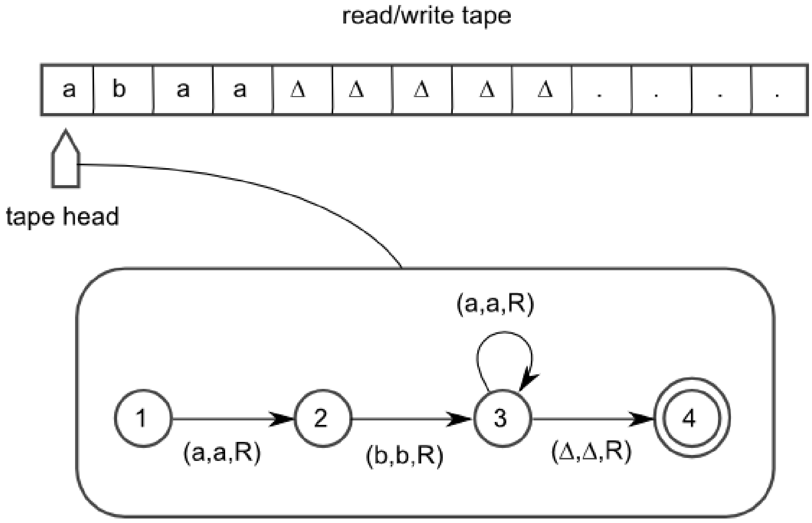

Figure 2: Turing machine that accepts aba*.

A logic without variables may be too weak (in the sense that it is no longer Turing-universal). If we equip the logic with a memory tape (or registers), it would naturally become Turing-universal again.

More concretely, consider this classical logic formula:
`raining → grass is wet`
and we can introduce a new kind of fomulas containing **actions**:
`if register A is '0' → write '1' into register B`.
We can even use data structures such as linked lists and trees instead of simple registeres. In Genifer 3.0 I am using lists.

4 Learning

The goal of learning is to discover a bunch of logic formulas to *explain* the world. To explain means to *deduce*.

Learning is achieved through **induction**, that is, to induce **rules** from **facts**.

In logical notation:
$KB \cup H \models E$
where KB is the **knowledge base** or background knowledge, H is the new

hypothesis (to be learned), E is the set of examples or new experience that needs to be explained, and $\models$ denotes **entailment**.

Inductive learning is a search inside the possible space of logic formulas; This search space is huge and our search would be much faster if the space is a lattice, ie, endowed with a **general-to-specific order**. In logic, such an order may be given by:

1. one concept being more **general** than another concept, for example:
   `animals` $\supseteq$ `dogs`

2. adding **conjunctions**, for example:
   `wear-glasses` $\wedge$ `has-long-hair`

In Genifer 4 I attempt to switch the setting to continuous space and apply gradient descent, but I am still facing a few unsolved obstacles. Genifer 3 does not need continuous space, the gradient is no use, so we could simply "fuck it" (use a genetic algorithm), which is much easier to do. Easy does not necessarily mean inferior; For example, in machine learning competitions, people find out that the most efficient classifiers are based on the theoretically simple **decision tree** method, rather than the theoretically complex support vector machines. Perhaps, genetic algorithm is sufficient? :)

Acknowledgments

I am heavily indebted to Pei Wang [2] [3] and Ben Goertzel [1] for their seminal contributions to AGI. To Abram Demski and Russell Wallace – we spent years exploring many ideas in logic. Also thanks to Matt Mahoney and other participants on the AGI mailing list. William Taysom, Seh, and Joseph Cheung helped implement the code.

References

[1] Goertzel, Pennachin, and Geisweiller. Building better minds: engineering beneficial general intelligence, 2011.

[2] Wang. *Rigid Flexibility - The Logic of Intelligence.* Springer applied logic series, 2006.

[3] Wang. *Non-axiomatic logic: a model for intelligent reasoning.* World Scientific (in press), 2013.