

English as a Macro Language & Programming Environment for Lisp



Henry Lieberman and Hugo Liu

MIT Media Lab

Cambridge, Mass. USA

<http://www.media.mit.edu/~lieber>

The crazy dream: Programming in English



Programming languages are too difficult for human beings

People can express procedures in English (or other natural language)

Why not program in natural language?

That was the original idea behind Cobol

(whereas Fortran tried to make programming like math)

Natural Language Programming and Lisp



Lisp community has the best tradition of always looking for new, higher-level programming ideas

Lisp well suited for natural language analysis

Lisp as a target language for translation from natural language to code

Well, what makes you think you can do it?



That was then, this is now

Natural language processing technology has vastly improved in the last decades

Nobody's asked the question lately, can we do it now?

Ambiguity is your friend



Isn't English hopelessly ambiguous?

Ambiguous, yes; hopeless, no!

Conventional programming forces *premature commitment* to representation

Common Sense resolves ambiguity *when necessary*

Interaction resolves ambiguity *when necessary*

New resources for Common Sense Reasoning

Open Mind Common Sense knowledge base

ConceptNet Semantic Net

MontyLingua/LangUtils (Eslick, Wed. 2pm)

© PEN MIND *Teaching computers the stuff we all know*
Are you an artificial intelligence researcher? Download Open Mind today!

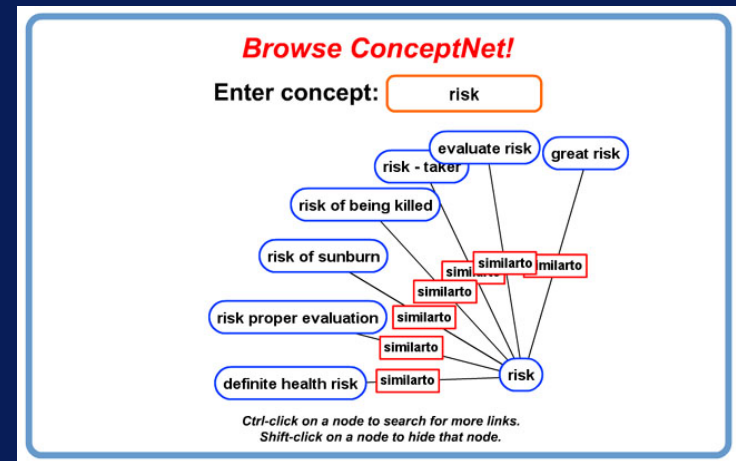
Welcome Tessa, to Open Mind! You have entered [124 items](#)

Search: [Other Activities!](#) · [Information](#) · [Preferences](#) · [Logout](#)

Open Mind

Search Results for wedding

Author	Knowledge
havasi	A bride wears a wedding gown
havasi	A bride and a groom are married in a wedding
Kohane	You can use a wedding ring to marry
jkasunic	going for a haircut is for a wedding
skoerber	Things that are often found together are: wedding gown, bouquet, bride, veil, groom



CMU Natural Programming Study



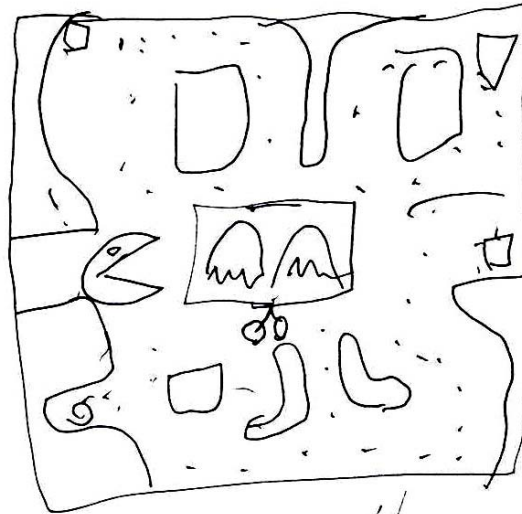
John Pane and Brad Myers studied 5th graders' description of Pac-Man

Also college students' descriptions of spreadsheet programs

Findings, e.g. Nobody talks about loops


Designed HANDS programming language

Childrens' description of Pac-Man



The yellow dots
are set through a maze
and Pac-man. The chomping
yellow ~~circle~~ circle has to collect them.
There are blinking big yellow
dots that allow you to eat
the ghosts (☹).

But what can you do if you can only *partially* understand English?



Need to make the interaction *fail-soft*

Rely on *dialogue* to resolve ambiguities and errors

Maybe don't eliminate code entirely

Emphasize *correspondence* between language and code

Metafor



An “outliner” for code as a *brainstorming* tool

Helps you discover and manage the overall structure but doesn't fill in all the details

- 1 English as a Macro Language & Programming Environment for Lisp
 - Henry Lieberman and Hugo Liu
 - MIT Media Lab
 - Cambridge, Mass. USA
 - <http://www.media.mit.edu/~lieber>
- 2 The crazy dream:
Programming in English
 - Programming languages are too difficult for human beings
 - People can express procedures in English (or other natural language)
 - Why not program in natural language?
 - That was the original idea behind Cobol
 - (whereas Fortran tried to make programming like math)

Programmatic Semantics



English to “scaffolding” code

Roughly,

Turns nouns into classes and/or instances;

Verbs into functions

Adjectives into instance variables

Resolves anaphora

Untangles conditional and loops

Propagates context

Metafor Interface



```
File
```

[user] (01:28:32) Pacman is a character who loves to run through a maze and eat dots.
[agent] (01:28:32) Ok. I created a new agent Pacman that is a kind of character agent. I added the ability for Pacman to run, which can be through a maze. I added the ability for Pacman to eat. A dot is something which can be eaten.
[user] (01:28:53) Whenever Pacman eats a dot, it disappears and he wins a point.
[agent] (01:28:53) Ok. Now we are going to think about Pacman eating... When Pacman eats a dot, the dot disappears and Pacman wins a point. ...Now that we are done thinking about Pacman eating, we are going to resume thinking about `__main__`.

When Pacman is running through the maze, if a ghost catches him, then he loses and the game is over.

DEICTIC STACK: [('Pacman', ['he', 'him', 'his', 'her', 'she', 'hers', 'who']), ('character', ['it']), ('maze', ['it']), ('dots', ['they', 'them']), ('dot', ['it']), ('dot', ['it']), ('Pacman', ['he', 'him', 'his', 'her', 'she', 'hers', 'who']), ('dot', ['it']), ('point', ['it'])]

DIR: ['__main__.Pacman', '__main__.dot']

CODETREE: [['__main__', 'FunctionT

```
def __main__():
    class Pacman(character):
        def run(maze):
            pass

        def eat(dot):
            dot.disappear()
            Pacman.win(point)

        def win(point):
            pass

    class dot:
        def disappear():
            pass
```

Target domain: MOOs (Programmable text adventure games)



Game itself is an interactive narrative

MOOs allow programmable objects & characters

```
Miranda gives you a hug  
Mouse says, "I'm here to hug you!"  
Mouse hugs Miranda  
Mouse says, "I made a mistake"
```

```
Stacy is a frendly killerwhale.  
She has Brown eyes. Her tail has a rash.  
1 script on Stacy:  
on flap this number "times"  
set flapped to number  
if flapped > 5 times  
    emote "blinks her eyes"  
endif  
end
```

Narrative stances



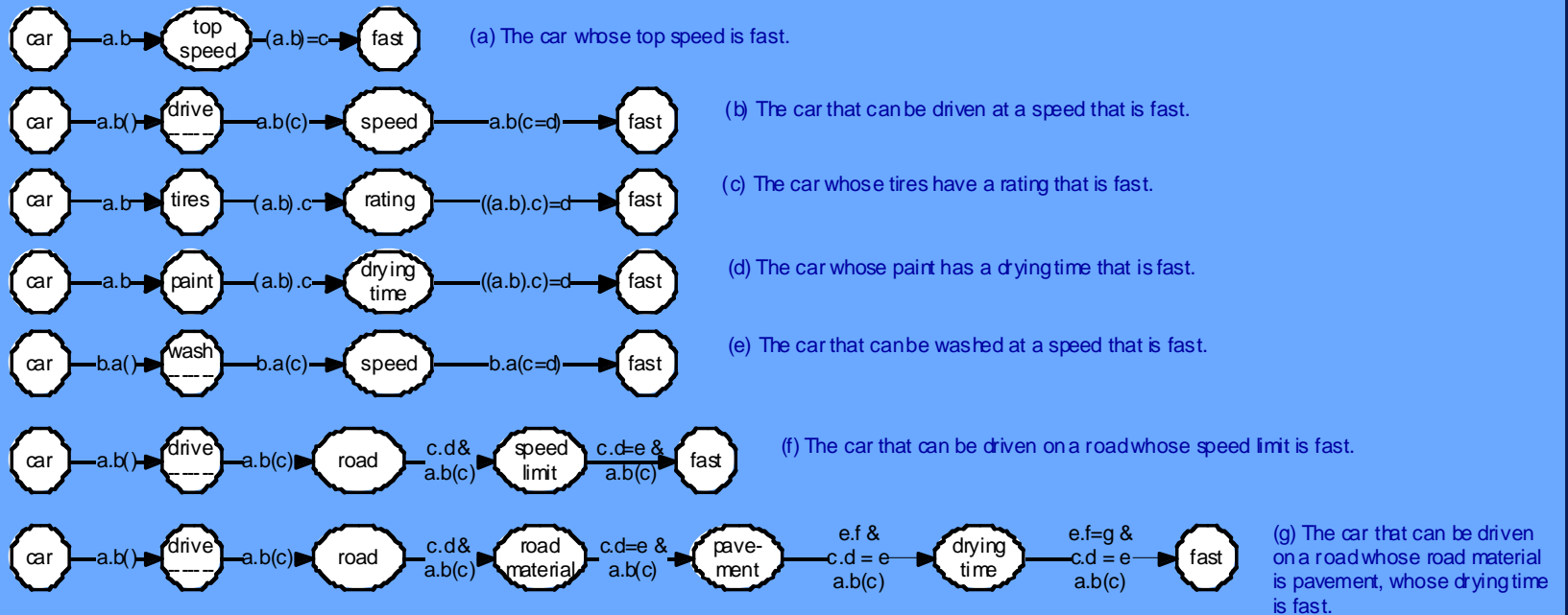
I want to make a bar with a customer. (1st person programmer)

There is a customer in the bar. (3rd person objective)

I am a customer sitting on a stool. (1st person program)

The bartender said, “Here is a customer” (3rd person subjective)

Bubble Semantics



Refactoring



There is a bar. (atom)

The bar contains two customers. (unimorphic list)

It also contains a waiter. (unimorphic wrt. persons)

It also contains some stools. (polymorphic list)

The bar opens and closes. (class / agent)

The bar is a kind of store. (inheritance class)

Some bars close at 6pm. (subclass or instantiatable)

Dialog for disambiguating



User: We're going to program a game, Pac-Man.

System: OK. I know a few things about how to make games. We need to decide what the playing field looks like, design some characters for the game, how the users are going to control the characters, and how the scoring system is going to work. OK?

U: Sure. The yellow dots are set through a maze, and Pac-Man, the chomping yellow circle, has to collect them.

S: You want the playing field to be a grid of squares, each containing a yellow dot, right? (System displays a grid of squares with yellow dots). To make it a maze, we need to have some walls. Can you draw some walls for me?

Dialog for disambiguating



S: OK. What does it mean for Pac-Man to "collect" a dot?

U: Whenever Pac-Man is on a square with a dot, he eats it.

That means that the dot is gone and the player gets another point.

S: OK, now I don't understand what "eats" means.

Can you explain it?

U: When you draw Pac-Man, you alternate between a picture of Pac-Man with his mouth open and a picture of Pac-Man with his mouth closed

English as a programming language?



The “Holy Grail” of easy to use interfaces for programming

Natural language technology has improved to the point that some understanding of programming concepts can be done

Good interaction design can make natural language interfaces “fail-soft”

Maybe Cobol wasn't such a bad idea after all!