Notation, Technology and Ideas

How an old tune went global and how natural-language speech applications can be built and deployed on the web

Richard Frost
School of Computer Science
University of Windsor

CSC 2011 Windsor

Example I
An old tune going global

- **Pachelbel composed the "Canon" (late 1600s)**

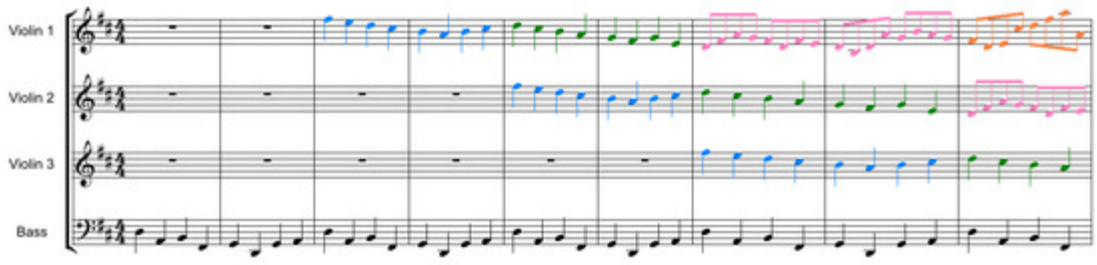
  http://www.youtube.com/watch?v=8Af372EQLck

- **Jerry C (Chang) re-arranged the Canon for electric guitar "Canon Rock" (2005)**

  http://www.youtube.com/watch?v=by8oyJztzwo

- **A youtube user, Impeto, spliced together 39 excerpts of musicians playing Jerry C's arrangement and called it the "Ultimate Canon Rock" (2007)**

  http://www.youtube.com/watch?v=dMWI_5NujBw



How did it go Global?

- Electric guitar (1930's)
- The Web (Tim Berners-Lee 1990's)
- Guitar TAB reborn (1940's)

```
----------------------|----------------------|----------------------
--19~--------17~------|--15~--------14~------|--12~--------10~------
----------------------|----------------------|----------------------
----------------------|----------------------|----------------------
----------------------|----------------------|----------------------
----------------------|----------------------|----------------------
```

Example II
Natural Language Speech Processing

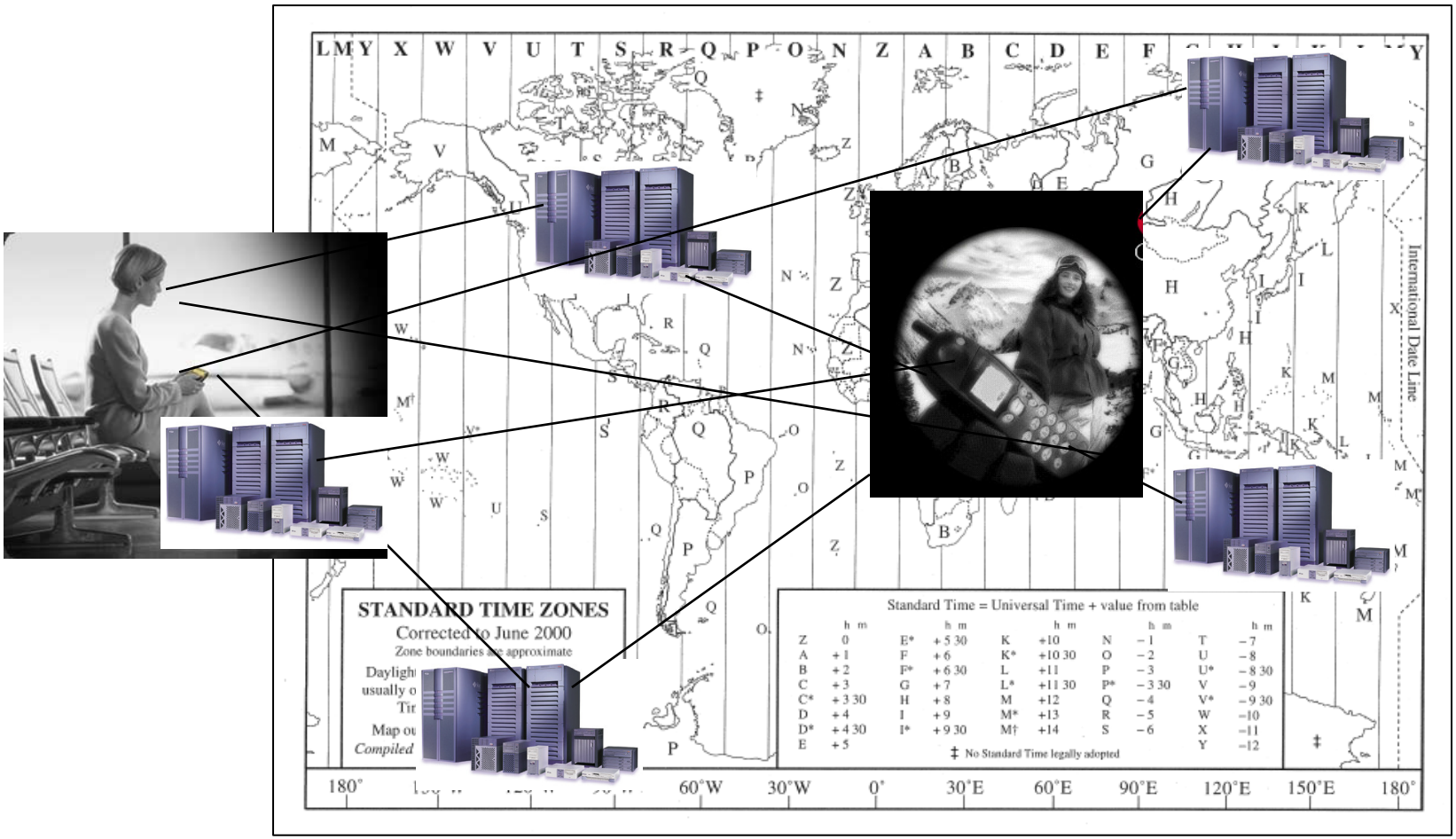
- **Leibniz had an idea that "The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right." (1685).**

- **Richard Montague, UCLA Department of Philosophy, created a formal denotation of a subset of English – natural language => λ-expressions (late 1960's/early70's).**

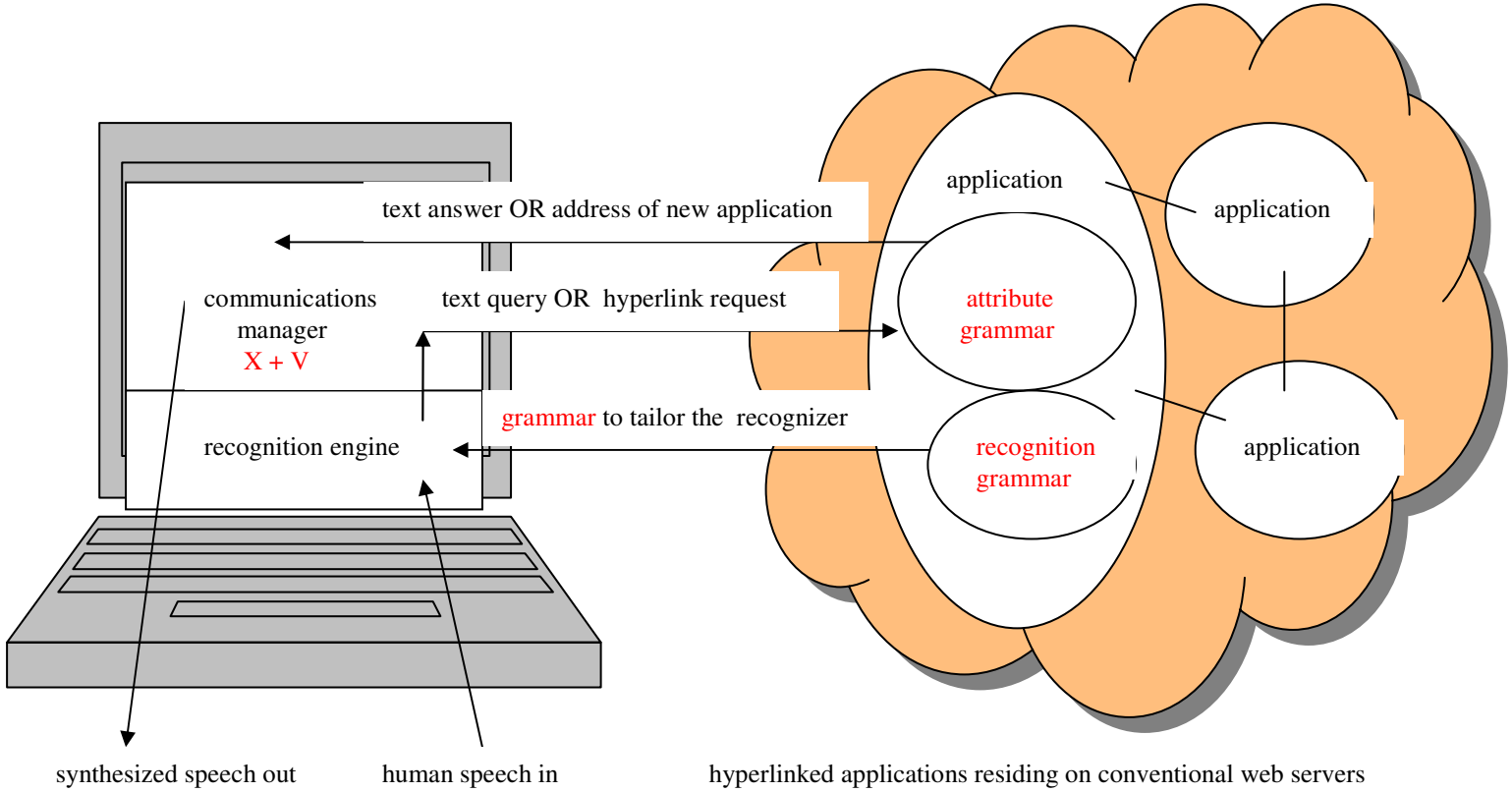
- **The SpeechWeb project - computer implementation of Montague's ideas + speech (1990's to present).**

SPEECHWEB DEMO

SpeechWeb - The Developer/User Perspectives



The SpeechWeb Architecture



Notation & technologies used in SpeechWeb

- **IBM's plugin speech recognizer is tailored by a context-free grammar (Chomsky 1950's).**
- **The natural-language processors are constructed as executable specifications of attribute-grammars (Knuth 1960's).**
- **The natural language semantics: Variation of Montague's NL semantics (1970's) developed in the λ-calculus (Church 1930's), and implemented in set-theory (Cantor 1874).**
- **LRRP Architecture uses common web communication protocols & a web script written in X+V (IBM ~2004).**

Objective – to facilitate construction,
deployment and access to natural-language speech applications

Recognition: Context-Free Grammars Guide Search

```
<question>      = does <termphrase> <verbphrase>;

<termphrase>    = <propernoun>   | <determiner> <noun>;

<propernoun>    = mars | jupiter | phobos | etc.

<determiner>    = a | every | the

<noun>          = moon | person | planet etc

<verphrase>     = <intrans_verb>
                | <transitive_verb> <termphrase>

<intrans_verb> = spin | etc.

etc.
```

Query Evaluators: Executable Specifications of Attribute Grammars

```
<question>    = does <termphrase> <verbphrase>
                question↑ANS = g [termphrase↑TVAL,
                                  verbphrase↑VVAL]
                etc.
<termphrase>  = <propernoun>
                termphrase↑TVAL = propernoun↑PNVAL
                | <determiner> <noun>
                termphrase↑TVAL = h [determiner↑DVAL,
                                     noun↑NVAL]
```

QUESTION: what are the VALS, and the functions g and h, etc. That is, what is the semantic theory?

Naïve natural-language semantics ⟦ ⟧

⟦Mars⟧ = the entity e_mars

⟦spins⟧ = the set of things that spin = spin_set

⟦Mars⟧ ⟦spins⟧ = True if e_mars ∈ spin_set

What is the meaning of "and" in the following?

(⟦Mars⟧ ⟦and⟧ ⟦Venus⟧) ⟦spin⟧

What is the meaning of "every" so that we can use the same meaning of ⟦and⟧ in the following:

((⟦every⟧ ⟦moon⟧) ⟦and⟧ ⟦Venus⟧) ⟦spin⟧

Not easy

For a better semantics, we need a new notation

An introduction to the λ-calculus

(x - y) can be thought of as a function of x (which is increasing) or a function of y (which is decreasing) or a function of x and y, or y and x.

λx(x - y) is the name of the function which takes one input n and which returns (n - y).

e.g. λx(x - y) 5 => (5 - y)

Now "λx(x - y) is increasing"

λxλy (x - y) is the name of the function which takes two inputs n and m and which returns (n - m).

e.g. λxλy(x - y) 5 8 => λy(5 - y) 8 => (5 - 8)=> -3

A Breakthrough - Montague's (1970's) approach to natural-language semantics (simplified)

⟦ Mars⟧ **⟦spins⟧**

= λp(p e_mars) **spins_pred**

=> spins_pred e_mars

=> True

Where:

spins_pred e = True if e has the property that it spins. spins_pred has type **entity -> bool**

Note **⟦Mars⟧** is of type (entity -> bool) -> bool

More of Montague

$[\![\textbf{every}]\!] \quad =$

$$\lambda p \lambda q\ \forall x (p\ x \rightarrow q\ x)$$

Example Use of ⟦**every**⟧

(⟦every⟧ ⟦moon⟧) ⟦spins⟧

= (λpλq ∀x(p x → q x) moon_pred) spins_pred

=> λq ∀x(moon_pred x → q x) spins_pred

=> ∀x(moon_pred x → spins_pred x)

=> True (if all things that are moons spin)

A note on types

The type of ⟦every moon⟧ = the type of ⟦Mars⟧

λq ∀x(moon_pred x → q x):: = λp(p e_mars)::

They are both in the syntactic category of termphrases and their meanings both are of type

(entity -> bool) -> bool

They are semantically as well as syntactically interchangeable

A polymorphic meaning for “and”

$[\![\mathbf{and}]\!] =$

λsλt (λr(s r & t r))

Takes two functions and creates a new function

Example use of ⟦and⟧

⟦Mars and Venus spin⟧

=> (⟦and⟧ ⟦Mars⟧ ⟦Venus⟧) ⟦spin⟧

=> (λsλt (λr(s r & t r)) λp(p e_mars) λp(p e_venus)) spins_pred

=>> λr(λp(p e_mars) r & λp(p e_venus) r) spins_pred

=> λp(p e_mars) spins_pred & λp(p e_venus) spins_pred

=> spins_pred e_mars & spins_pred e_venus

=> True & True

=> True

The power of λ-Calculus
Calculating the meaning of transitive verbs

What is the denotation of "discovered" in

⟦Hall discovered Phobos⟧ (Montague left it undefined)

We developed the answer by solving the following
(just as we solve sq (x * 2) = 36 for x)

⟦Hall⟧ (⟦discovered⟧ ⟦Phobos⟧)
= discover_pred (e_hall, e_phobos)

=> λp(p e_hall) (⟦discovered⟧ λp(p e_phobos))
= discover_pred (e_hall, e_phobos)

One solution to this is:

⟦discovered⟧ = λz z(λxλy discover_pred(y,x))

A summary of meanings

$[\![\textbf{Mars}]\!] = \lambda p(p\ e\_mars)$

$[\![\textbf{spin}]\!] = spin\_pred$

$[\![\textbf{moon}]\!] = moon\_pred$

$[\![\textbf{every}]\!] = \lambda p \lambda q\ \forall x(p\ x \rightarrow q\ x)$

$[\![\textbf{a}]\!] = \lambda p \lambda q\ \exists x(p\ x\ \&\ q\ x)$

$[\![\textbf{and}]\!] = \lambda s \lambda t\ \ \lambda r(s\ r\ \&\ t\ r)$

$[\![\textbf{discovered}]\!] = \lambda z\ z(\lambda x \lambda y\ discover\_pred(y,x))$

Result is a Fully Compositional Semantics

- **Words/phrases of the same syntactic category have the same semantic type.**
- **The composition rule is always simple function application.**
- **The semantics covers a large sub-set of classical first-order English.**

```
does (((every moon) $and (every planet)) spin)

how many moons that orbit a red planet were discovered
                        by the person who discovered Nereid

which planet is orbited by no moon
```

- **The meaning of words can be defined in terms of other words.**

$$[\![\texttt{discoverer}]\!] = [\![\texttt{person who discovered a thing}]\!]$$

BUT it is not computationally efficient!!

Convert to set notation to achieve computational efficiency

$(\llbracket \text{every} \rrbracket \llbracket \text{moon} \rrbracket) \llbracket \text{spins} \rrbracket$ => $\forall x(\text{moon\_pred } x \rightarrow \text{spins\_pred})$

which is computationally inefficient at best.

Change all predicates to sets and modify all denotations accordingly:

$(\llbracket \text{every} \rrbracket \quad \llbracket \text{moon} \rrbracket) \quad \llbracket \text{spins} \rrbracket$

=> $(\lambda p \lambda q\ (p \subseteq q)$ moon_set) spins_set

(which has linear complexity wrt the shortest list if the sets are ordered)

Note: now $\llbracket \text{Mars} \rrbracket = \lambda p\ e\_mars \in p$

(different to naïve semantics)

Result is an Easily Implementable Efficient Compositional Semantics

- **λ-calculus semantics is easily implemented in a higher-order pure functional programming language (Miranda, Haskell etc.)**
- **The NL interpreter can be constructed as an executable specification of an attribute grammar.**
- **The Speech front end can be easily created with X+V and CFGs.**

***voilà :* we have the SpeechWeb**

A problem

We cannot accommodate prepositional phrases such as:

`Did Hall discover Phobos with a telescope`

Proposed Solution: Change the notation.

This time change the way in which we represent the world.

Accommodating propositional phrases

⟦Hall⟧ ((⟦discovered⟧ ⟦Phobos⟧) ⟦with⟧ (⟦a⟧ ⟦telescope⟧))
=> discover_pred (e_hall, e_phobos) ?????

Naïve solution: discover_pred' (subject,object,implement)

What about "Did Hall discover Phobos in 1877?"

Our proposed solution: change the underlying semantic model to be based on binary-relationships = triples

e_hall	subject	disc#123
e_phobos	object	disc#123
e_45	implement	disc#123
e_45	member	set_telescopes
1877	year_of	disc#123
disc_#123	member	set_discoveries
U.S, Naval OBS	location	disc#123

The Message

The notation (e.g. guitar TAB, context-free grammars, X + V, attribute-grammars, λ-calculus, set-theory, etc.) is as important as the technology – it affects the way we think, solve problems, communicate ideas, and implement solutions using technology.

Think about the notation FIRST
when trying to solve computational problems!