

Call by Meaning

Hesam Samimi

Yoshiki Ohshima

Alessandro Warth



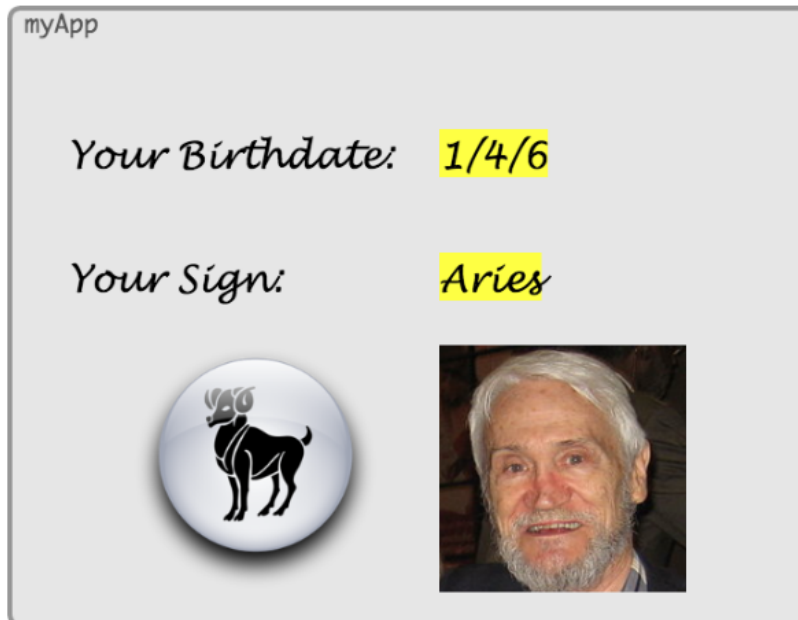
Chris Deaton

Todd Millstein



UCLA

Code View



Code

```
run: function (userInput) {
  // parse the input
  this.userDateArray = this.getDate(userInput)
  if (this.userDateArray) {
    // convert into date object
    this.userDate = this.dateArrayToDate(this.userDateArray)
    // compute zodiac sign
    this.userSign = this.getZodiacSign(this.userDate)
    canvas.getElementByName('signName').text = this.userSign
    // get a url to img of zodiac sign
    this.signImgUrl = this.getZodiacSignImg(this.userSign)
    // get a uri to img of a CS person with that sign
    this.csPersonImgUrl = this.getCSPersonImg(this.userSign)
    canvas.getElementByName('signImg').acyclicSet('src', this.signImgUrl)
    canvas.getElementByName('csPersonImg').acyclicSet('src', this.csPersonImgUrl)
  }
}
```

Values

```
name: "myApp"
userDateArray: [6,3,1]
userDate: "1906-04-01T07:00:00.000Z"
userSign: "Aries"
signImgUrl: "Aries.png"
csPersonImgUrl: "cs-wesley-clark.png"
```

Looking, Looking, ...

The image shows a Google search interface. At the top, the address bar displays the URL `https://www.google.com/#q=javascript+string+to+date`. Below the address bar is a row of application icons including Apps, G, P, G Cal, Wflowy, G, Y, G, News, A LA, Cal, and sketchpad1. The Google logo is on the left, and the search bar contains the text "javascript string to date". To the right of the search bar is a microphone icon. Below the search bar are navigation links: Web (highlighted with a red underline), News, Shopping, Videos, Maps, More (with a dropdown arrow), and Search tools. The search results section shows "About 10,300,000 results (0.43 seconds)". The first result is titled "javascript - Converting string to date in js - Stack Overflow" in purple, with the URL `stackoverflow.com/questions/5619202/converting-string-to-date-in-js` and a dropdown arrow. The snippet below the title reads: "Apr 11, 2011 - How can I convert a **string to date** in js? var st = "**date** in some format" ... The best you can do is use the ISO format: YYYY-MM-DD or ... var st ...". The second result is titled "Date.parse() - JavaScript | MDN" in blue, with the URL `developer.mozilla.org > ... > Date` and a dropdown arrow. The snippet below the title reads: "Aug 20, 2014 - The **Date.parse()** method parses a **string** representation of a **date**, and returns the number of milliseconds since January 1, 1970, 00:00:00 ...". The third result is titled "Date - JavaScript | MDN" in blue, with the URL `https://developer.mozilla.org/.../JavaScript/.../...` and a dropdown arrow.

Web News Shopping Videos Maps More Search tools

About 10,300,000 results (0.43 seconds)

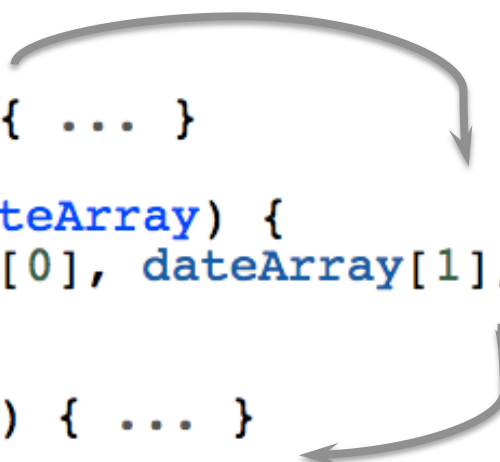
javascript - Converting string to date in js - Stack Overflow
`stackoverflow.com/questions/5619202/converting-string-to-date-in-js`
Apr 11, 2011 - How can I convert a **string to date** in js? var st = "**date** in some format" ...
The best you can do is use the ISO format: YYYY-MM-DD or ... var st ...

Date.parse() - JavaScript | MDN
`developer.mozilla.org > ... > Date` Mozilla Developer Network
Aug 20, 2014 - The **Date.parse()** method parses a **string** representation of a **date**, and returns the number of milliseconds since January 1, 1970, 00:00:00 ...

Date - JavaScript | MDN
`https://developer.mozilla.org/.../JavaScript/.../...` Mozilla Developer Network

Hmmm, some glue code...

```
getDate: function (dateText) { ... }  
  
dateArrayToDate: function (dateArray) {  
    return new Date(dateArray[0], dateArray[1], dateArray[2])  
}  
  
getZodiacSign: function (date) { ... }
```



The diagram illustrates the flow of data between three functions. A curved arrow originates from the `dateText` parameter of the `getDate` function and points to the `dateArray` parameter of the `dateArrayToDate` function. Another curved arrow originates from the `dateArrayToDate` function and points to the `date` parameter of the `getZodiacSign` function, indicating a sequential data flow.

Moral of Story

#1

*We develop by composing
existing software*

Moral of Story

#2

Lots of stuff
on the Internet!

Moral of Story

#3

Finding

Understanding

Using

Composing

Maintaining

Moral of Story

Finding

manual!

Understanding

anti-automation!

Using

non-scalable!

Composing

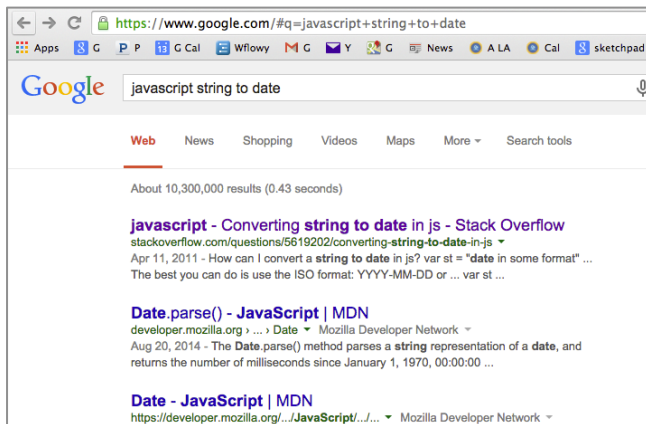
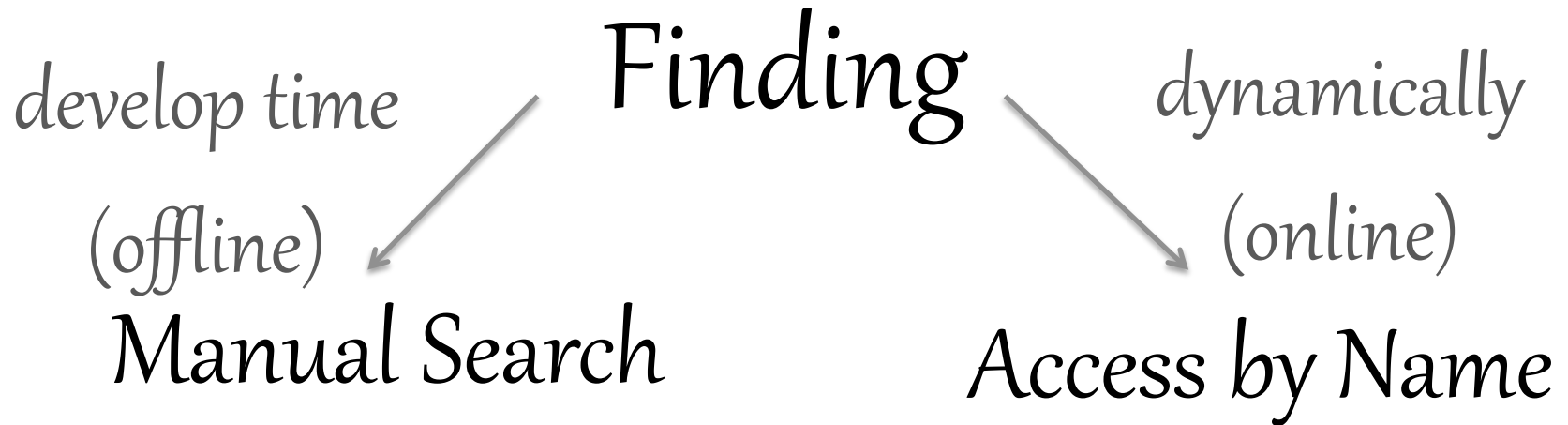
reverse engineering!

Maintaining

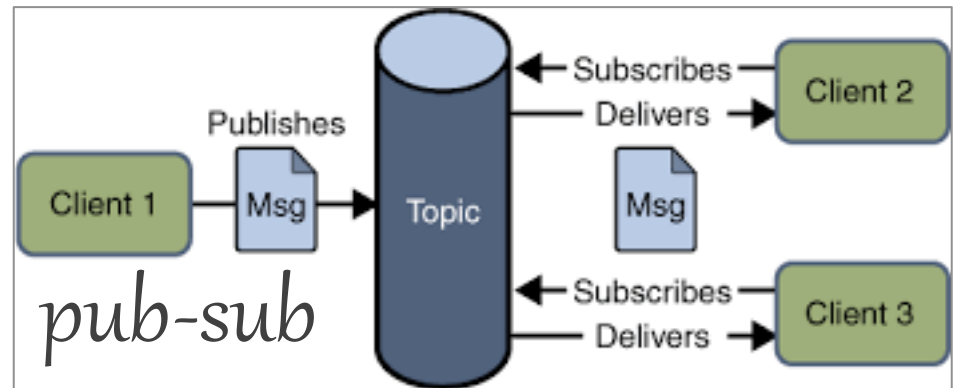
hard-coding!

fragility!

Status Quo in PL



```
import Date
Date.getTime()
```



Agreement on “name” doesn’t scale!

getDate(t)

parseDate(t)

getDate(t)

Date.getDate(t)

foo(t)

new version

entirely unrelated (eHarmony)

gets a new module

exactly what we want

Agreement on “name”
doesn't scale!

What's the alternative?

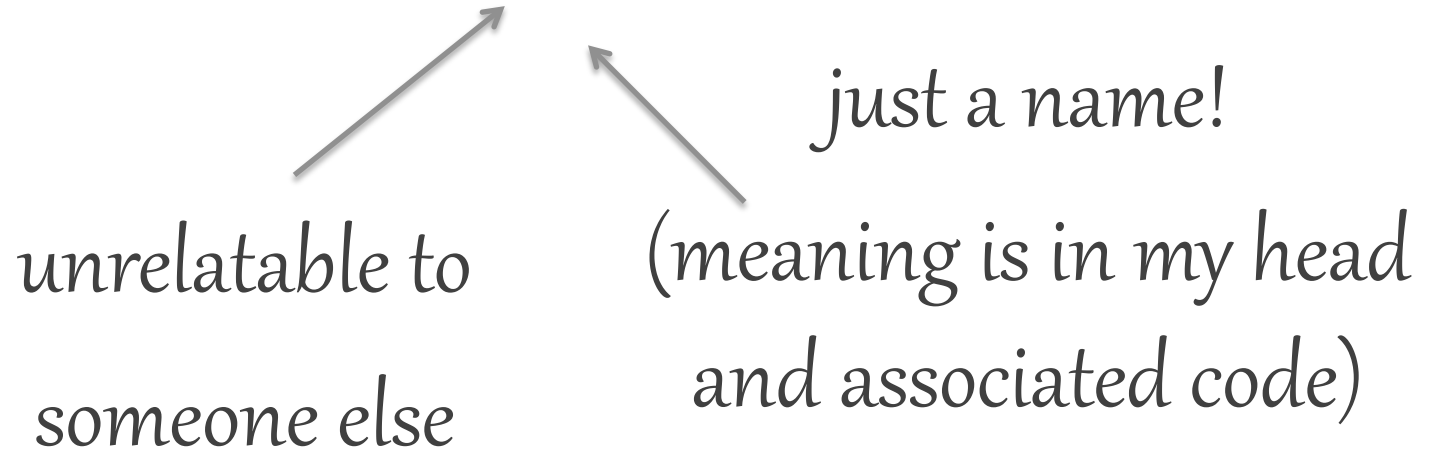
Agreement on “name” doesn't scale!

What's the alternative?

- type system?
- contracts?

Your symbols mean nothing to me!

`com.calendar.Date`



What's the problem?

`com.calendar.Date`

lack of :

- grounding to the “real world”
- shared automatable semantic “understanding”

How do we do it
in the real world?

(“It’s on the tip of my tongue...”)

Not a solution!

But a vision!

To set off a revolution!
of a PL where...

“Call by Meaning” Vision

programming environment provides:

- semantic ontology & specification lang.
- grounding to the “real world”
- embedded vast shared “real world” KB
- reasoning engine (theorem prover)
- automatable semantic “understanding”

Call by Meaning

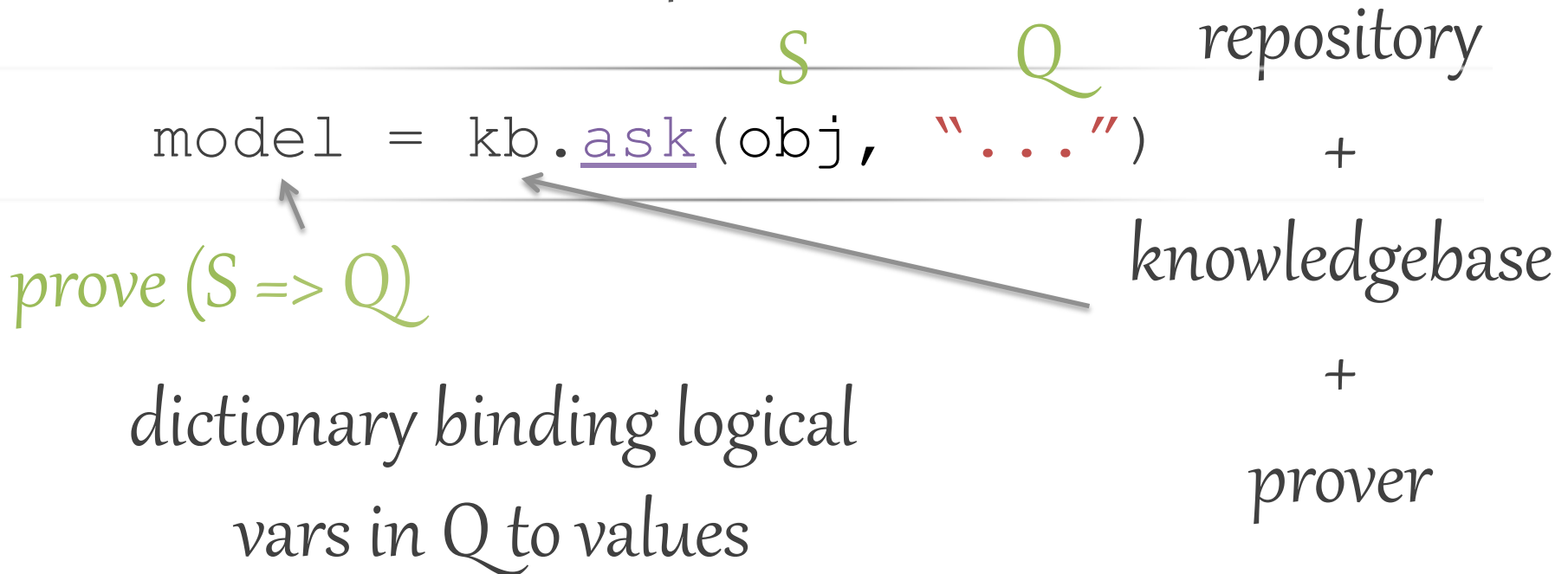
explain a component

```
spec: function () {  
  return "..."  
}
```

Call by Meaning

ask questions about a component

- discover API
- adapt its data



Call by Meaning

discover a component

S Q
`obj = kb.find("...")`

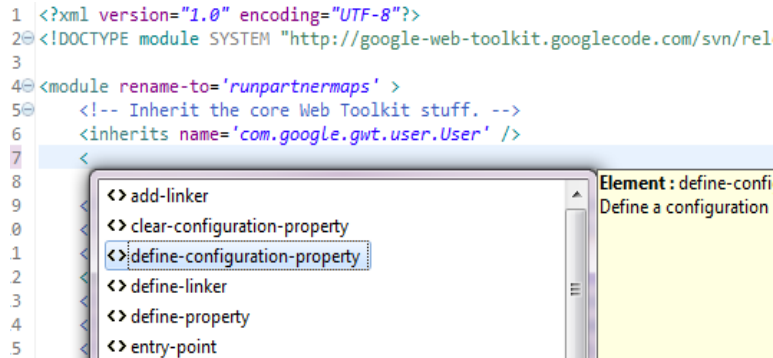


$\text{prove } (S \Rightarrow Q)$

Call by Meaning

offline

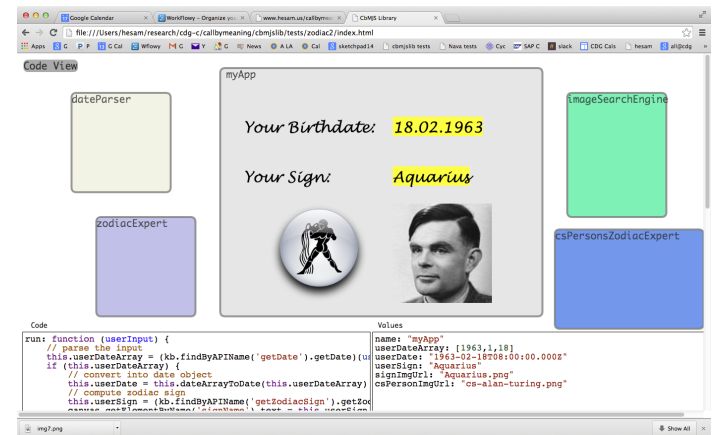
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE module SYSTEM "http://google-web-toolkit.googlecode.com/svn/rel
3
4 <module rename-to='runpartnermaps' >
5   <!-- Inherit the core Web Toolkit stuff. -->
6   <inherits name='com.google.gwt.user.User' />
7   <
8
9   <> add-linker
10  <> clear-configuration-property
11  <> define-configuration-property
12  <> define-linker
13  <> define-property
14  <> entry-point
15
```



The screenshot shows an IDE window with XML code. A list of auto-suggestions is displayed for the 'define-configuration-property' element. The suggestions include: add-linker, clear-configuration-property, define-configuration-property (highlighted), define-linker, define-property, and entry-point. A tooltip for the selected element shows 'Element: define-confi' and 'Define a configuration'.

IDE Auto Suggestions

Part of Algorithm!



online

Computers don't speak
human or share human
knowledge!

20+ Years of Semantic Web



ConceptNet 5
Computers just got smarter.



Discovering

dateParser

```
getDate_spec: function () {  
  return  
    '(and (primaryFunction ?GETDATE SemanticParsing programUsed)'  
    + '(programVariableIsaFunctionWithArg1 ?GETDATE ?Y CharacterString)'  
    + '(programVariableIsaFunctionWithOutput ?GETDATE ?Z Date))'  
}
```

myApp

```
kb.findByMeaning(  
  '(and (programVariableHasProperty ?THAT ?THAT-P)'  
    + '(programParsesInputInto ?THAT-P CharacterString Date))',  
  this, this.dateParserFound)
```


Agreement on “name” doesn't scale!

getDate(t)

parseDate(t)

getDate(t)

Date.getDate(t)

foo(t)

new version

entirely unrelated (eHarmony)

gets a new module

exactly what we want

Discovering API

dateParser

```
getDate_spec: function () {  
  return  
    '(and (primaryFunction ?GETDATE SemanticParsing programUsed)'  
      + '(programVariableIsaFunctionWithArg1 ?GETDATE ?Y CharacterString)'  
      + '(programVariableIsaFunctionWithOutput ?GETDATE ?Z Date))'  
    + '(programVariableHasName ?GETDATE "getDate")'
```

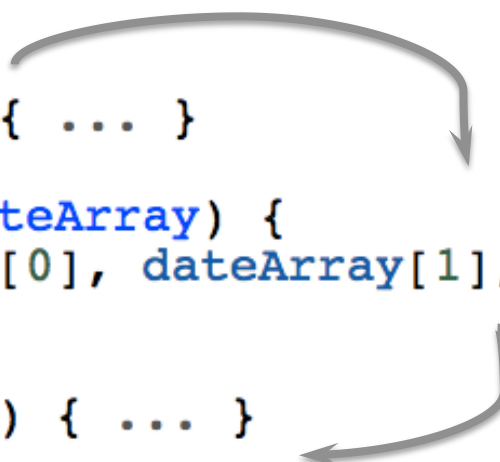
myApp

{NAME: "getDate"}

```
kb.askByMeaning(  
  '(and (programVariableHasProperty ?THAT ?THAT-P)'  
    + '(behaviorCapable ?THAT-P SemanticParsing programUsed)'  
    + '(programVariableIsaFunctionWithArg1 ?THAT-P ?THAT-Y CharacterString)'  
    + '(programVariableIsaFunctionWithOutput ?THAT-P ?THAT-Z Date)'  
    + '(programVariableHasName ?THAT-P ?NAME))',  
  this, dateParser, false, this.onParserAPIDiscovered)
```

Hmmm, some glue code...

```
getDate: function (dateText) { ... }  
  
dateArrayToDate: function (dateArray) {  
    return new Date(dateArray[0], dateArray[1], dateArray[2])  
}  
  
getZodiacSign: function (date) { ... }
```

A diagram consisting of three curved arrows. The first arrow starts from the 'dateText' parameter in the 'getDate' function and points to the 'dateArray' parameter in the 'dateArrayToDate' function. The second arrow starts from the 'dateArray' parameter in the 'dateArrayToDate' function and points to the 'date' parameter in the 'getZodiacSign' function. The third arrow starts from the 'date' parameter in the 'getZodiacSign' function and points back to the 'dateText' parameter in the 'getDate' function, forming a cycle.

dateParser

Adapting

```
getDate_spec: function () {  
  return this.output ?  
    '(programVariableIsaFunctionWithOutput ?GETDATE'  
    + ' (DayFn ' + this.output[2]  
    + ' (MonthFn ' + monthNames[this.output[1]]  
    + ' (YearFn ' + this.output[0] + '))) Date)'  
    : undefined  
}
```

4

myApp

```
kb.askByMeaning(  
  '(and (programVariableHasProperty ?THAT ?GETDATE)'  
    + '(programVariableIsaFunctionWithOutput ?GETDATE ?DATE ?_)'  
    + '(dayNumberOfDate ?DAY ?DATE)'  
    + '(monthOfDate ?MONTH ?DATE)'  
    + '(yearOfDate ?YEAR ?DATE))',  
  this, this.dateParser, false, this.dateParserInfoReceived)  
}
```

{ DAY: 4, ... }

dateParser

Adapting+

```
getDate_spec: function () {  
    return this.output ?  
        '(programVariableIsaFunctionWithOutput ?GETDATE'  
        + ' (DayFn ' + this.output[2]  
        + ' (MonthFn ' + monthNames[this.output[1]]  
        + ' (YearFn ' + this.output[0] + '))) Date)'  
        : undefined  
}
```

myApp

{SIGN: "Cancer", ...}

```
kb.askByMeaning(  
    '(and (programVariableHasProperty ?THAT ?THAT-GETDATE)'  
    + '(programVariableIsaFunctionWithOutput ?THAT-GETDATE ?DATE ?_)'  
    + '(zodiacSignDateRange ?SIGN '  
    + ' (DayOfYearFn ?BEGIN-MONTH ?BEGIN-DAY) '  
    + ' (DayOfYearFn ?END-MONTH ?END-DAY)) '  
    + '(laterThanOrCooriginatingWith '  
    + ' (DayFn ?END-DAY (MonthFn ?END-MONTH (YearFn 1776))) ?DATE)'  
    + '(laterThanOrCooriginatingWith ?DATE '  
    + ' (DayFn ?BEGIN-DAY (MonthFn ?BEGIN-MONTH (YearFn 1776))))',  
    this, this.dateParser, false, this.dateParserInfoReceived)
```

Many challenges to making this a reality!

writing specs!

scalability of proving!

adopting a lingua-franca spec lang!

steep learning curve!

other forms of specs?

unpredictability/complexity!

safety & security!

Related Work

- Semantic Web ← an “afterthought”
- PL approaches ← no grounding in “real world”
 - Zones project
 - Wolfram Language, Type Providers
 - API discovery (offline/online)

Thank you!

<http://www.hesam.us/callbymeaning>