Genyris

Why Write Yet Another Language?

There was no practical language with semantic web features.

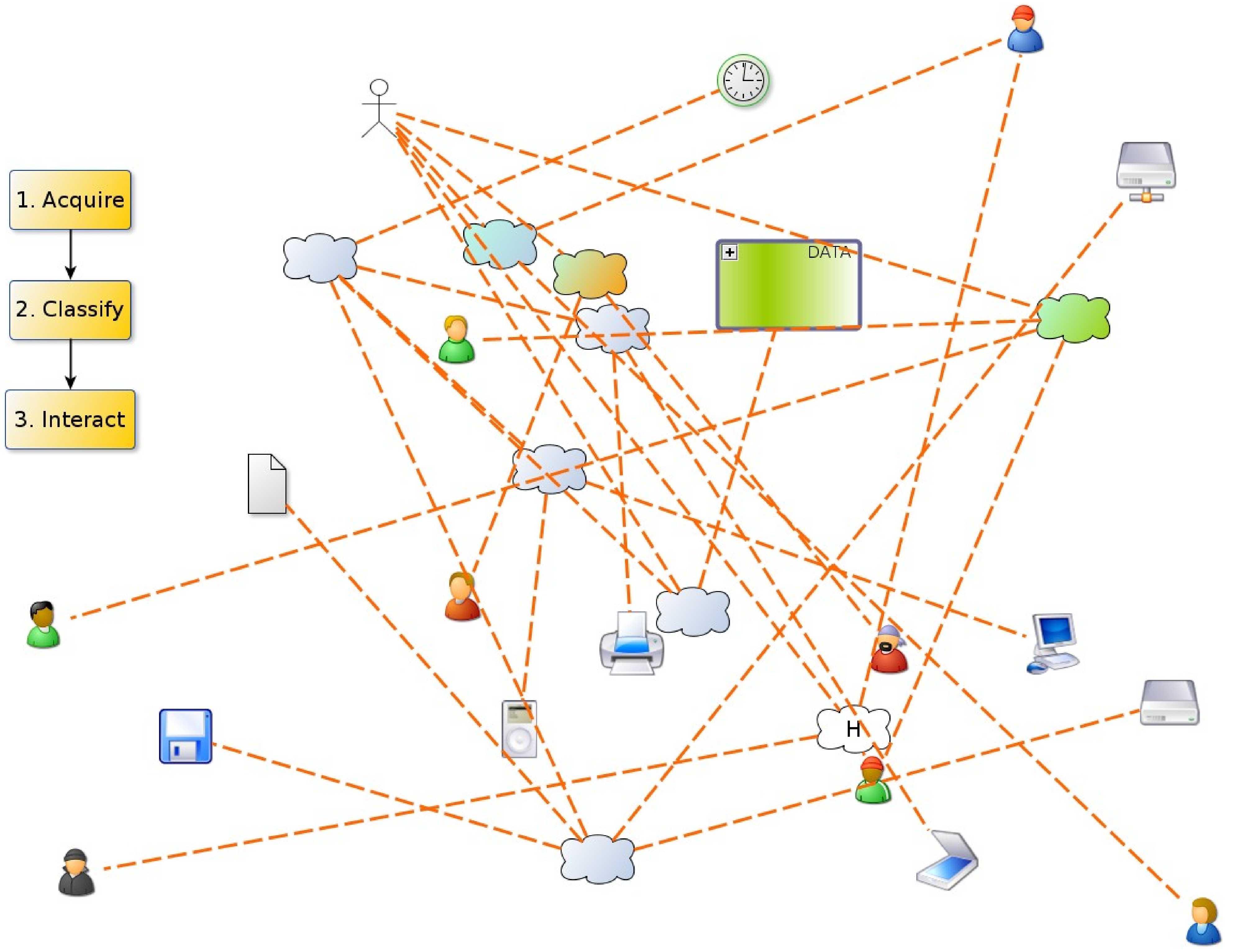
Semantic Web:

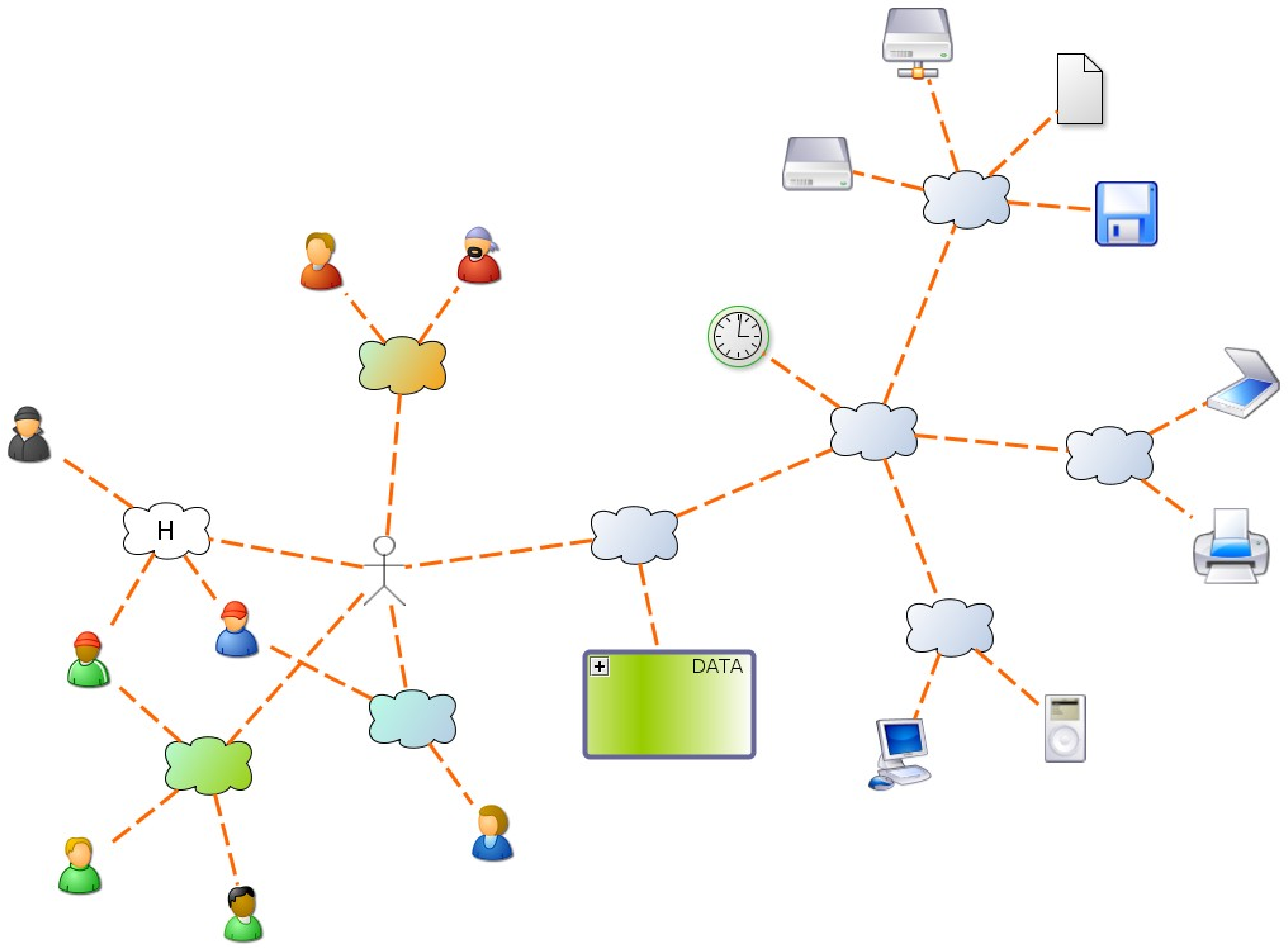
- Classification external to objects
- Multiple classifications per object
- Dynamic (re)classification
- World-Wide namespace

THOUGHT PROCESS

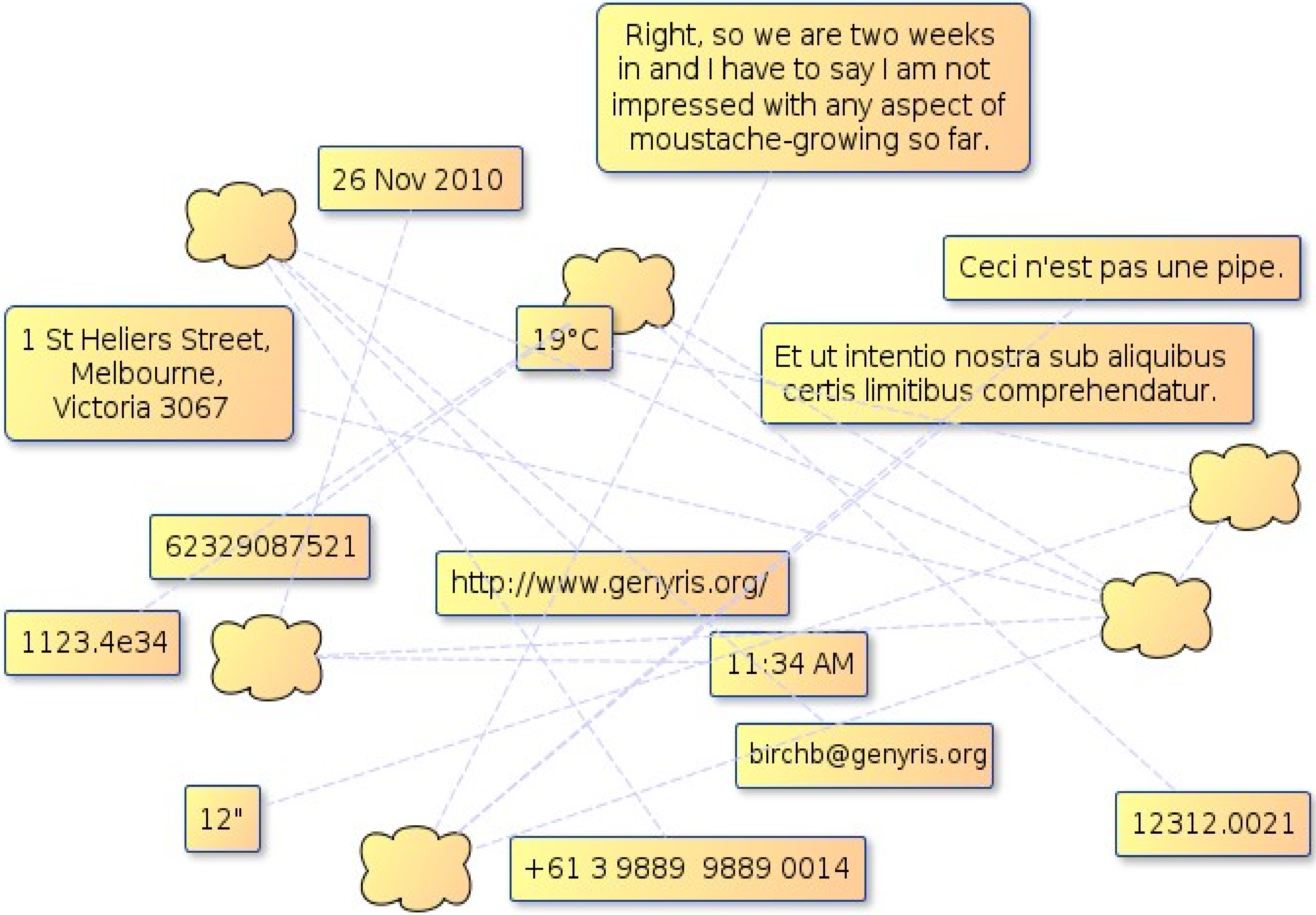
IMPLEMENTATION INFLUENCES

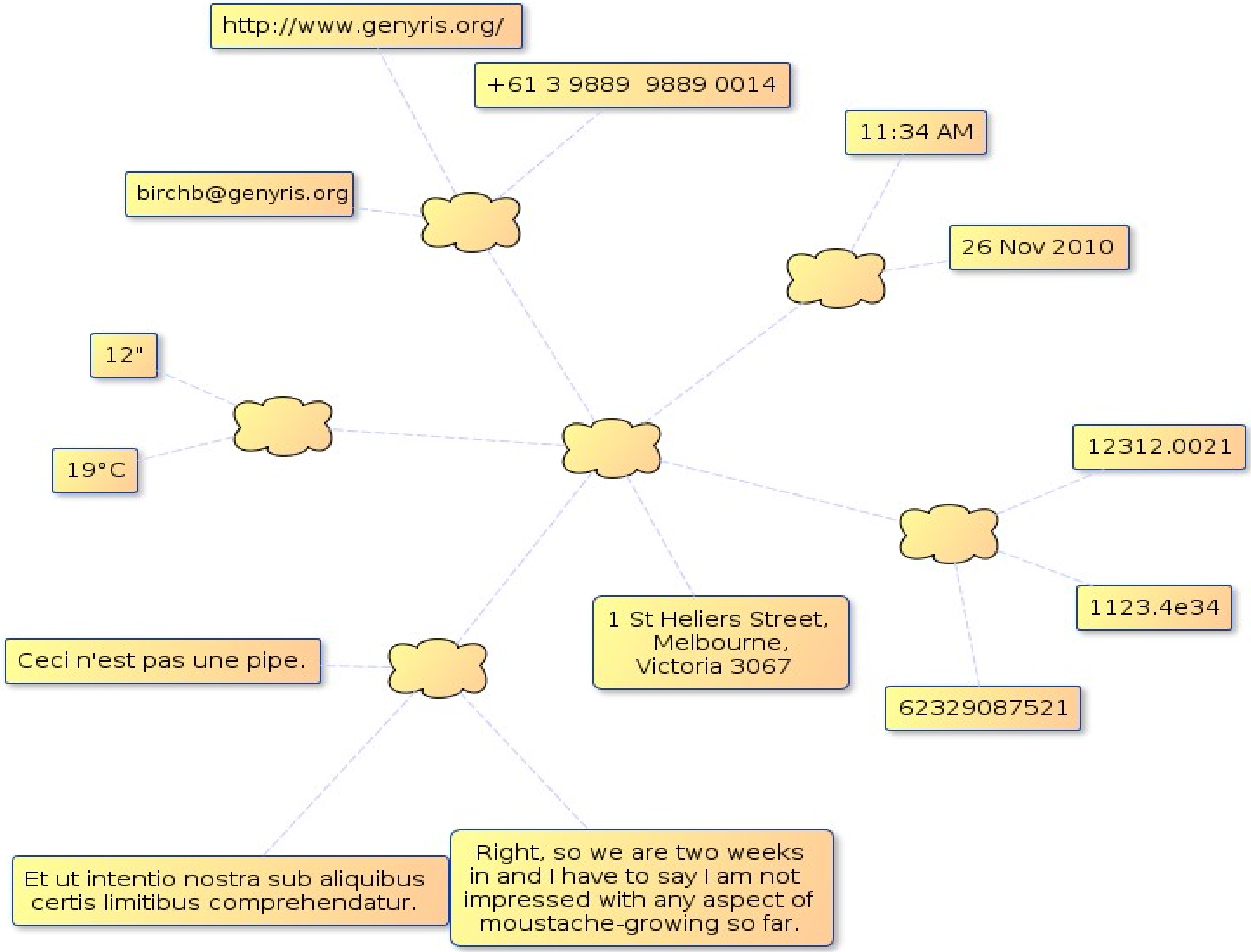
DEMONSTRATIONS

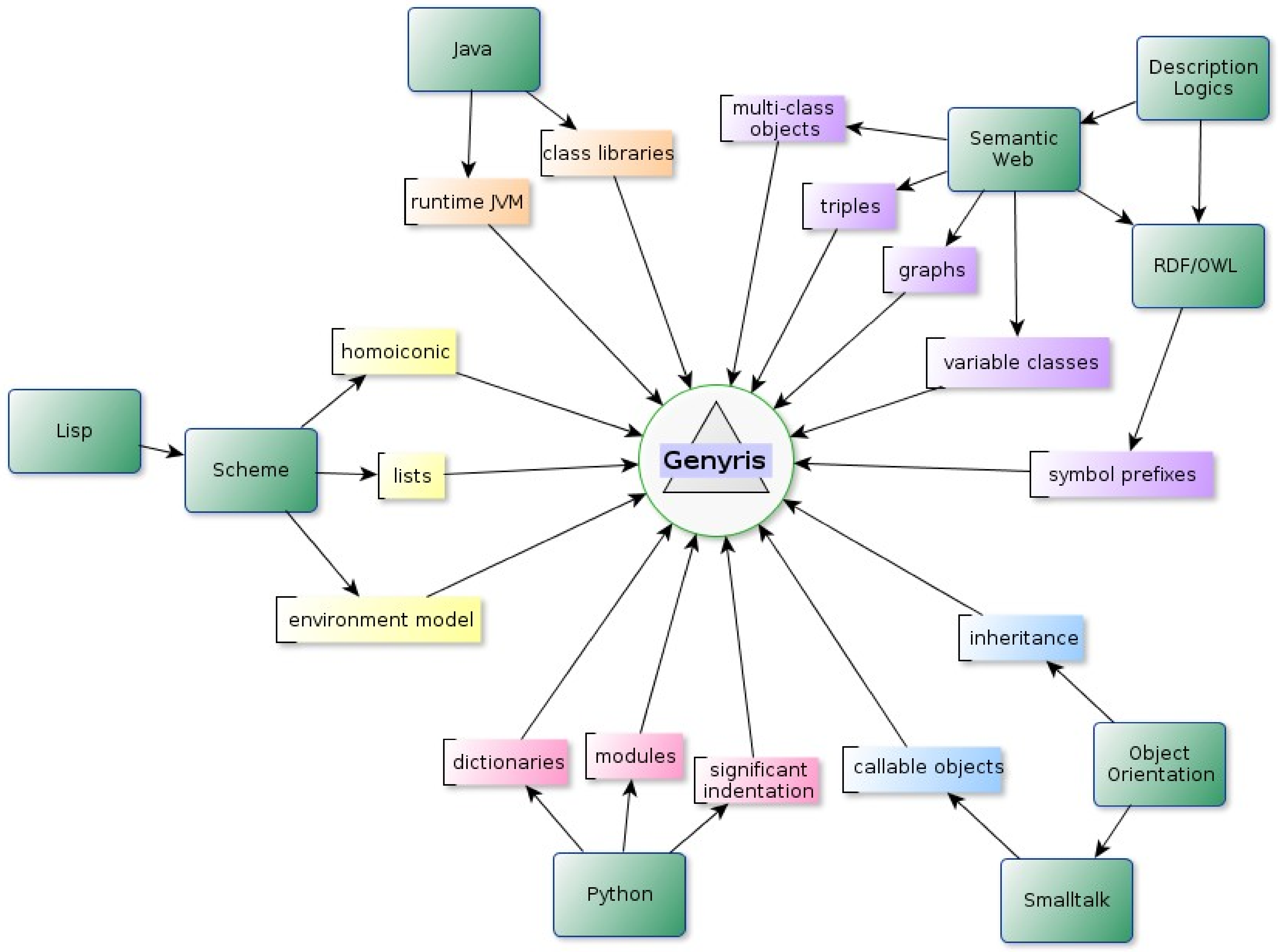

1. Acquire
2. Classify
3. Interact
DATA
H

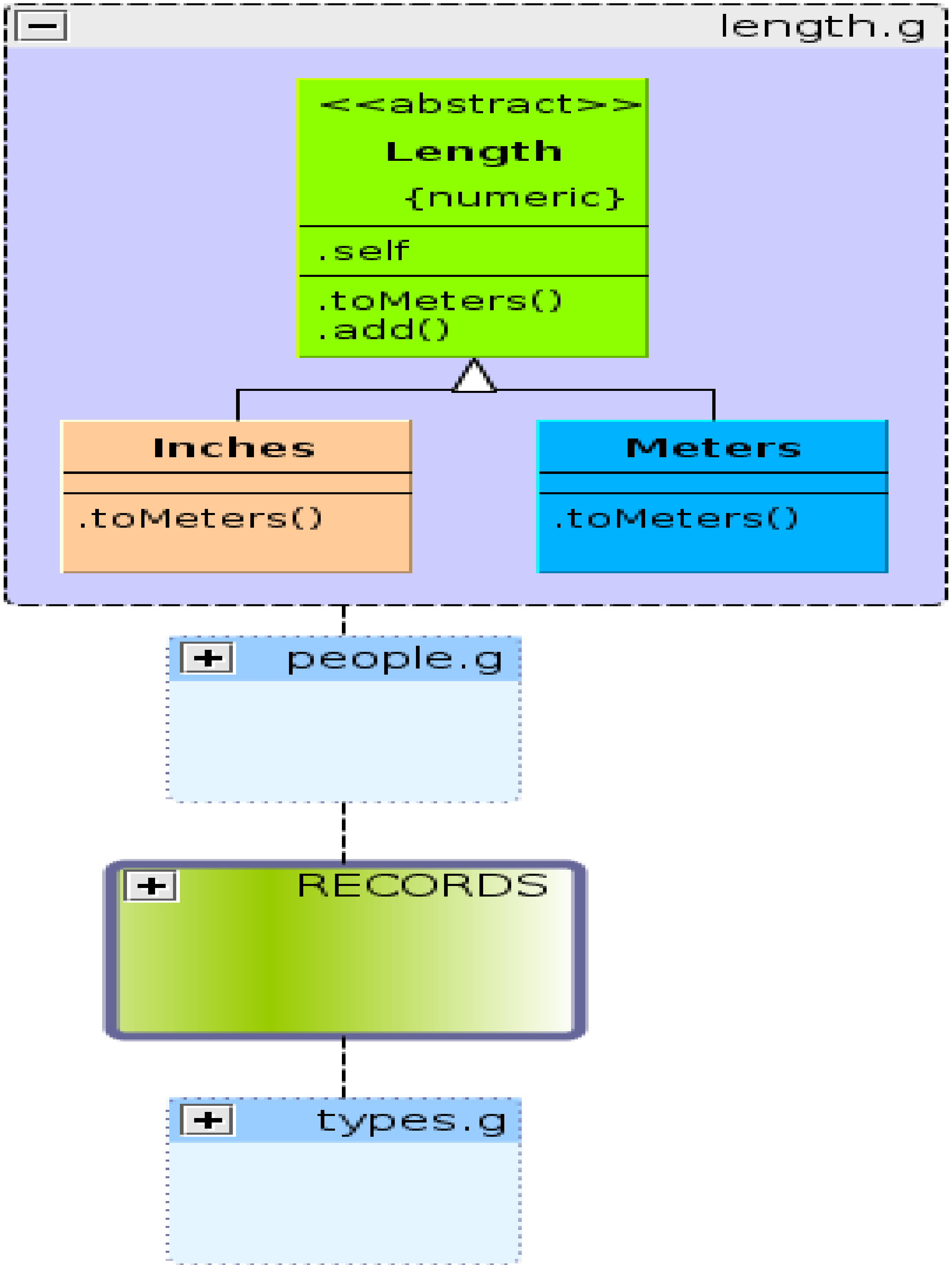


$$\begin{tikzcd}
\boxed{\text{1. Acquire}} \arrow[d] \\
\boxed{\text{2. Classify}} \arrow[d] \\
\boxed{\text{3. Interact}}
\end{tikzcd}$$

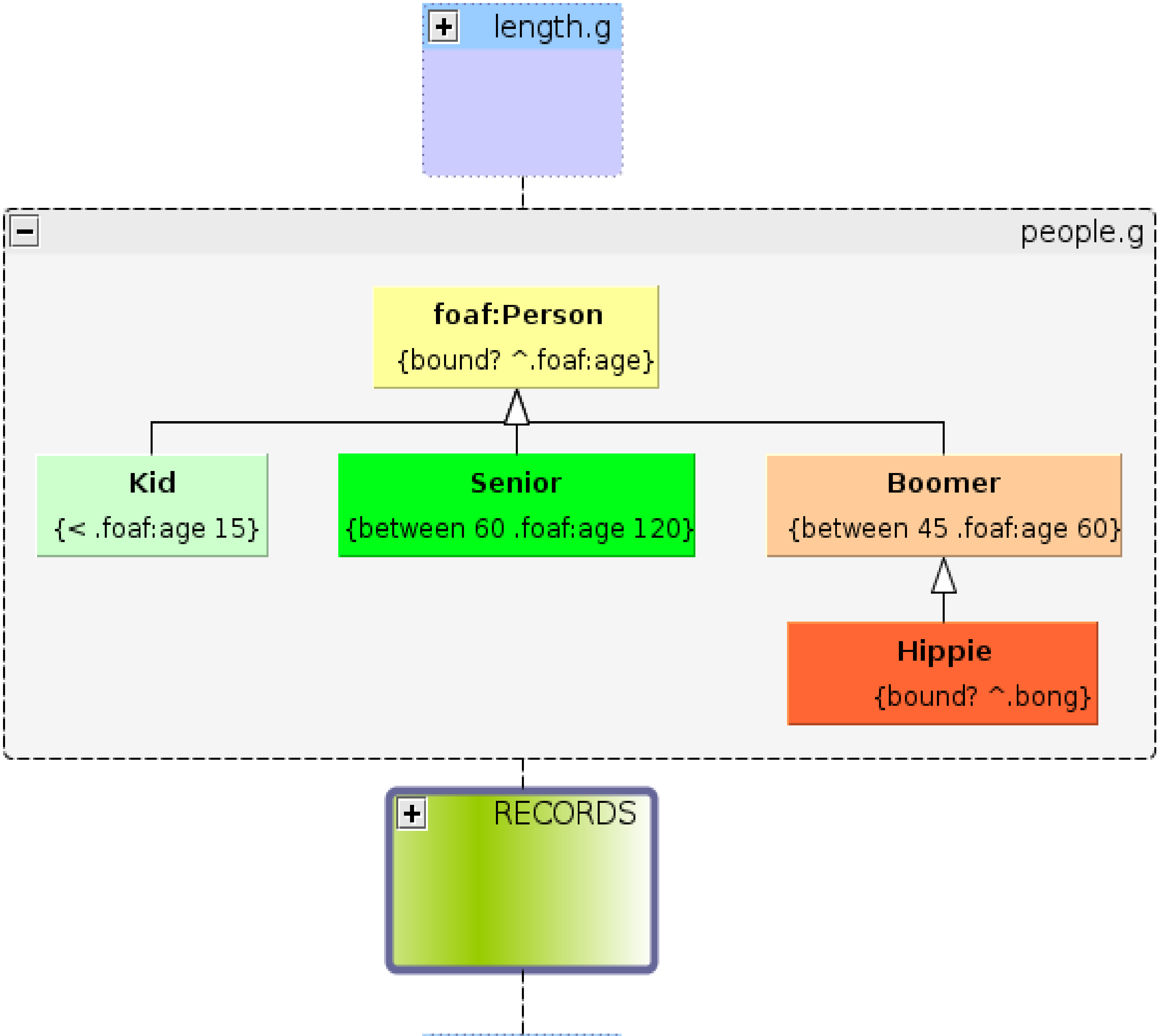

Right, so we are two weeks in and I have to say I am not impressed with any aspect of moustache-growing so far.
26 Nov 2010
Ceci n'est pas une pipe.
19°C
Et ut intentio nostra sub aliquibus certis limitibus comprehendatur.
1 St Heliers Street, Melbourne, Victoria 3067
62329087521
http://www.genyris.org/
1123.4e34
11:34 AM
birchb@genyris.org
12"
12312.0021
+61 3 9889 9889 0014


http://www.genyris.org/
+61 3 9889 9889 0014
11:34 AM
birchb@genyris.org
26 Nov 2010
12"
19°C
12312.0021
1123.4e34
1 St Heliers Street, Melbourne, Victoria 3067
Ceci n'est pas une pipe.
62329087521
Et ut intentio nostra sub aliquibus certis limitibus comprehendatur.
Right, so we are two weeks in and I have to say I am not impressed with any aspect of moustache-growing so far.

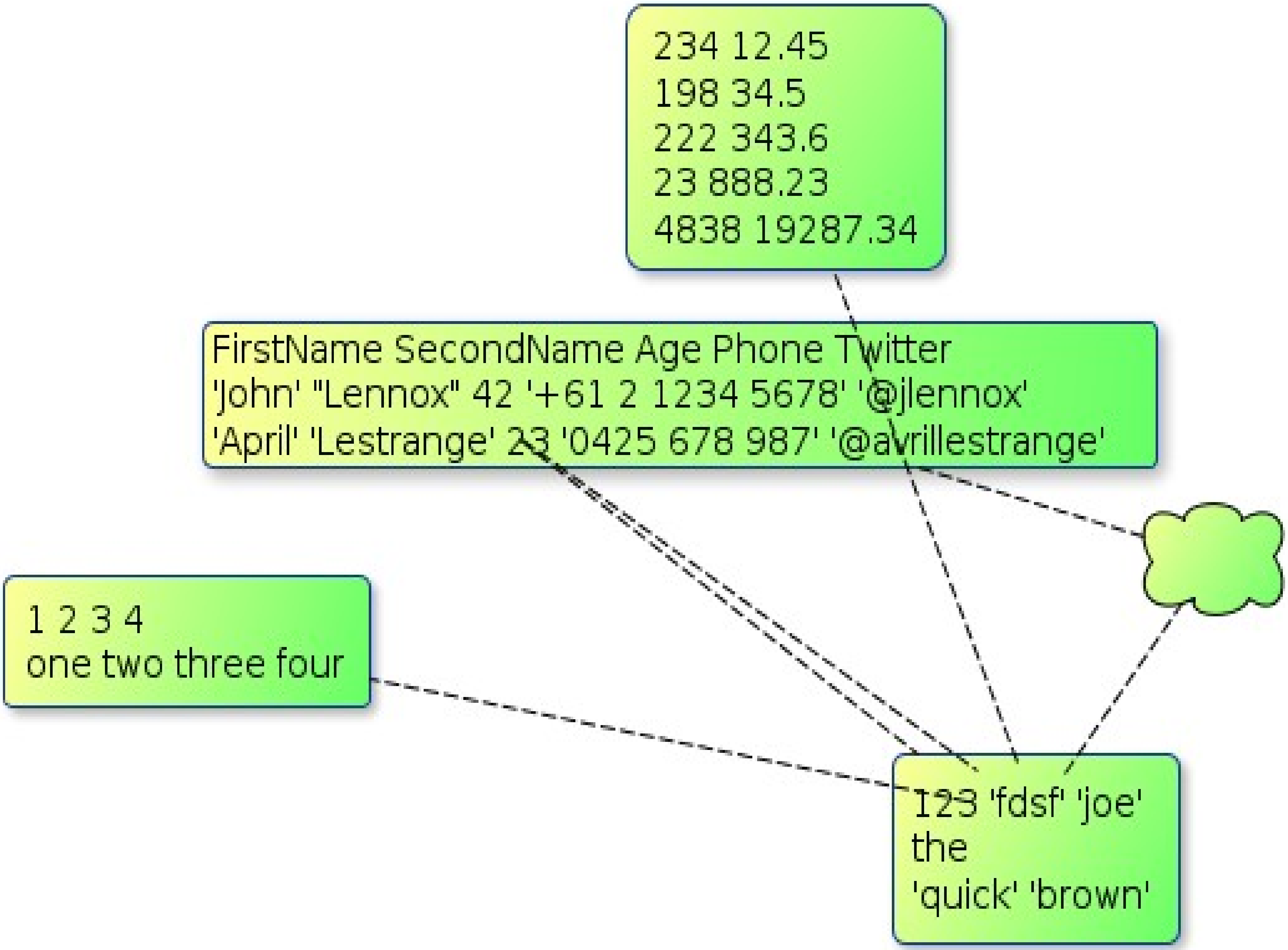

Java
Description Logics
multi-class objects
class libraries
Semantic Web
runtime JVM
triples
graphs
RDF/OWL
variable classes
homoiconic
Lisp
Scheme
lists
Genyris
symbol prefixes
environment model
inheritance
dictionaries
modules
significant indentation
callable objects
Object Orientation
Python
Smalltalk

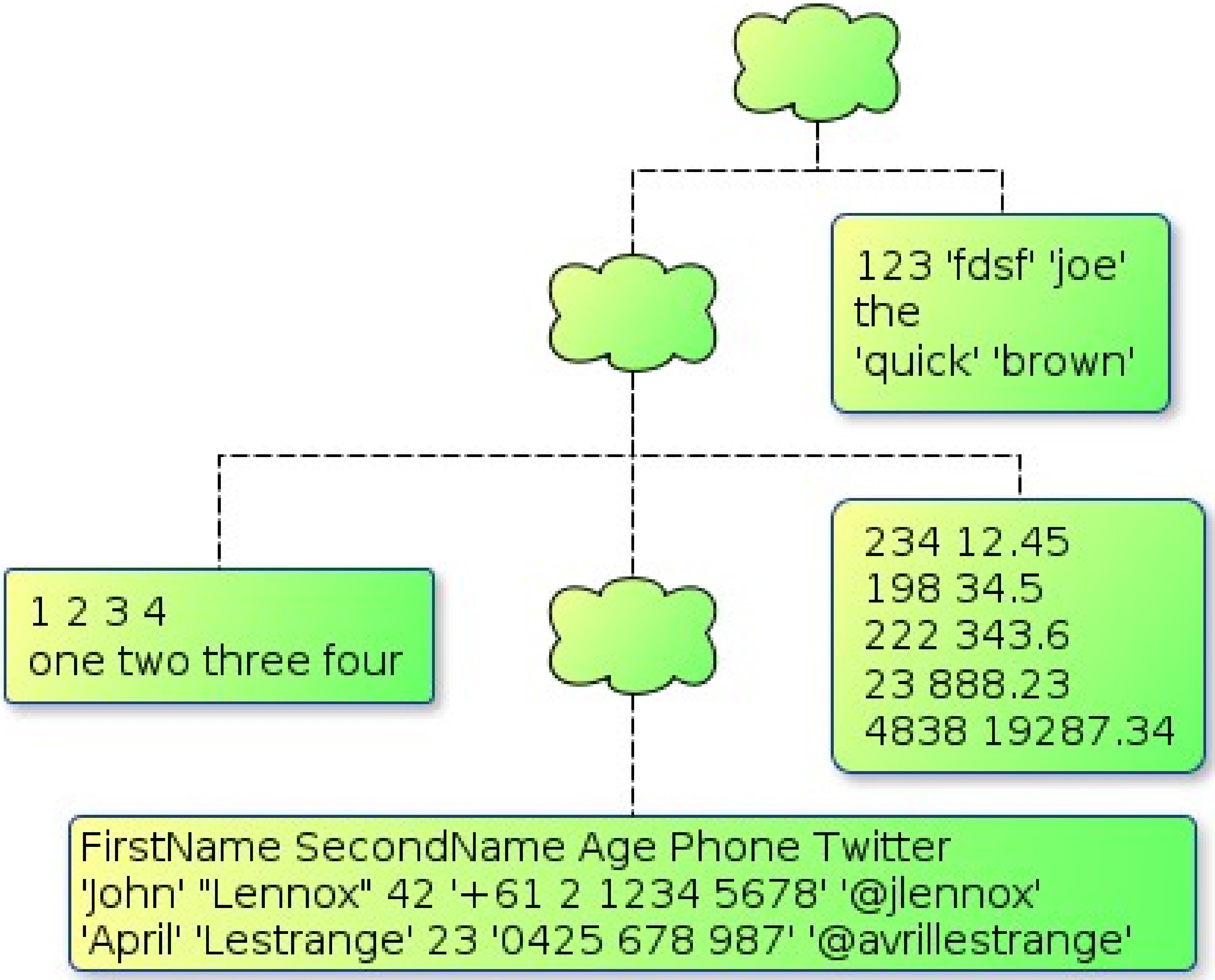

length.g
<<abstract>>
Length
{numeric}
.self
.toMeters()
.add()
Inches
.toMeters()
Meters
.toMeters()
people.g
RECORDS
types.g

```
class foaf:Person ()
   def .valid? (obj)
      # simple type inference from foaf:age to foaf:Person
      obj
         bound? ^.foaf:age

class Kid (foaf:Person)
   def .valid? (obj)
      obj
         < .foaf:age 15

class Senior (foaf:Person)
   def .valid? (obj)
      obj
         between 60 .foaf:age 120

class Boomer (foaf:Person)
   def .valid? (obj)
      obj
         between 45 .foaf:age 60
```


length.g
people.g
foaf:Person
{bound? ^.foaf:age}
Kid
{< .foaf:age 15}
Senior
{between 60 .foaf:age 120}
Boomer
{between 45 .foaf:age 60}
Hippie
{bound? ^.bong}
RECORDS

```
class Length()
  def .toMeters()
    raise "Oops - you invoked an abstract class."

class Inches(Length)
  def .toMeters()
    tag Meters (* .self 0.0254)

class Meters(Length)
  def .toMeters() .self

Length
  def .add(other)
    tag Meters
      + (.toMeters)
        other (.toMeters)


define a-meter
  tag Meters 1

define a-foot (tag Inches 12)

assert
  equal?
    a-foot (.add a-meter)
    1.3048
```


234 12.45
198 34.5
222 343.6
23 888.23
4838 19287.34
FirstName SecondName Age Phone Twitter
'John' "Lennox" 42 '+61 2 1234 5678' '@jlennox'
'April' 'Lestrange' 23 '0425 678 987' '@avrillestrange'
1 2 3 4
one two three four
123 'fdsf' 'joe'
the
'quick' 'brown'

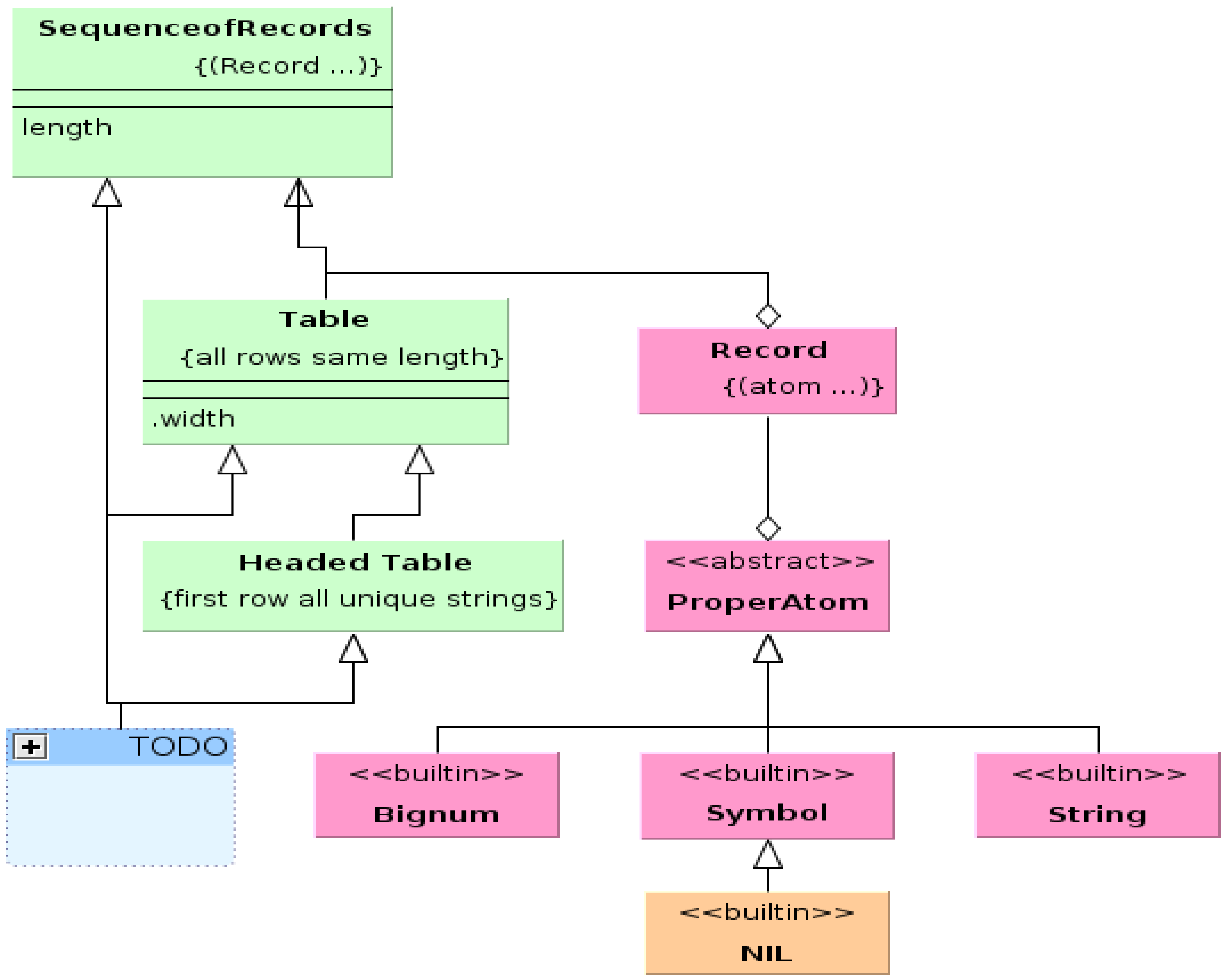


```
123 'fdsf' 'joe'
the
'quick' 'brown'
```

```
1 2 3 4
one two three four
```

```
234 12.45
198 34.5
222 343.6
23 888.23
4838 19287.34
```

```
FirstName SecondName Age Phone Twitter
'John' "Lennox" 42 '+61 2 1234 5678' '@jlennox'
'April' 'Lestrange' 23 '0425 678 987' '@avrillestrange'
```


SequenceofRecords
{(Record ...)}
length
Table
{all rows same length}
.width
Record
{(atom ...)}
Headed Table
{first row all unique strings}
<<abstract>>
ProperAtom
TODO
<<builtin>>
Bignum
<<builtin>>
Symbol
<<builtin>>
String
<<builtin>>
NIL

```
@prefix : "http://www.genyris.org/lang/types#"

class :ProperAtom()
  # atomic atoms !
  def .valid?(obj)
     or
       is-instance? obj String
       is-instance? obj Bignum
       is-instance? obj Symbol


class :Record()
  # a list of proper atoms
  def .valid?(obj)
    cond
      (is-instance? obj Pair)
        cond
          obj!right  # end of the list?
            and
               :ProperAtom!valid? obj!left
               :Record!valid? obj!right
          else
            :ProperAtom!valid? obj!left
  def .same-type?(type record)
    and
      is-instance? record!left type
      cond
        record!right  # more in the list?
          :Record!same-type? type record!right
        else
          true
  def .unique?(record)
    and
      not (member? record!left record!right)
      cond
        record!right  # more in the list?
          :Record!unique? record!right
        else
          true
```

```
class :SequenceOfRecords()
  # A list of records
  def .valid?(obj)
    cond
      (is-instance? obj Pair)
        cond
          obj!right  # end of the list?
            and
               :Record!valid? obj!left
               :SequenceOfRecords!valid? obj!right
          else
            :Record!valid? obj!left


class :Table(:SequenceOfRecords)
  # A list of equal-length records
  def .valid?(obj)
    define mylength (length obj!left)
    cond
      obj!right
        cond
           (equal? mylength (:Table!valid? obj!right))
              mylength
      else   # end of the list
        mylength
  # method returns the width of the table
  def .width()
    length (.self .left)

class :HeadedTable(:Table)
  # A list of equal-length records with a first-row
  # header of unique strings or symbols
  def .valid?(obj)
    and
      :Record!unique? obj!left
      or
        :Record!same-type? String obj!left
        :Record!same-type? Symbol obj!left
```