

Programming Systems in Artificial Intelligence

Introduction

Siegfried Nijssen

2/02/16

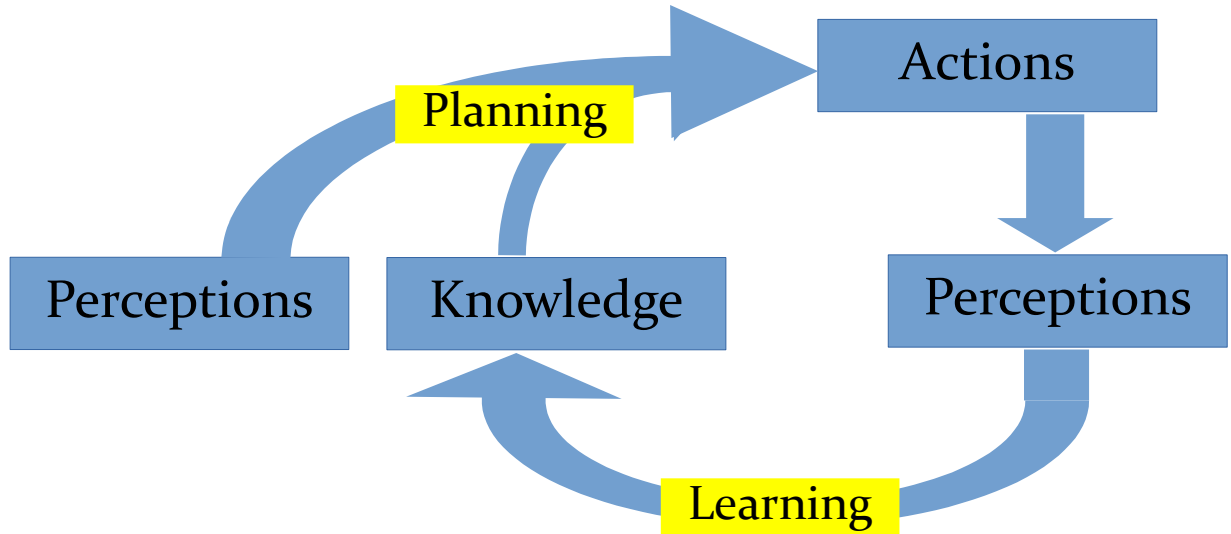


**Universiteit
Leiden**
The Netherlands

Discover the world at Leiden University

Artificial Intelligence

- Artificial intelligence studies the development of *intelligent agents*
- An *agent* is something that perceives and acts in an environment
- Requires solving several challenges:
 - Knowledge representation:
how does an agent represent its knowledge and perceptions?
 - Reasoning, planning:
how does an agent deduce an action based on its perceptions and its knowledge?
 - Learning:
how does an agent update its knowledge based on its perceptions?



Artificial Intelligence & Programming

- How do we implement knowledge, reasoning, and learning for a specific application?
- One approach: implement everything yourself in C++, Java or Python
- Example: learning a neural network
- Advantage:
 - You have full control over the computer and can make the code as efficient as you want
- Disadvantages:
 - Coding an application takes a lot of time
 - The code is not likely to be efficient

```
class Net
{
public:
    Net(const vector<unsigned> &topology);
    void feedForward(const vector<double> &inputVals);
    void backProp(const vector<double> &targetVals) {};
    void getResults(vector<double> &resultVals) const {};

private:
    vector<Layer> m_layers; // m_layers[layerNum][neuronNum]
};

void Net::feedForward(const vector<double> &inputVals)
{
    assert(inputVals.size() == m_layers[0].size() - 1);

    // Assign (latch) the input values into the input neurons
    for (unsigned i = 0; i < inputVals.size(); ++i) {
        m_layers[0][i].setOutputVal(inputVals[i]);
    }

    // Forward propagate
    for (unsigned layerNum = 1; layerNum < m_layers.size(); ++layerNum) {
        for (unsigned n = 0; n < m_layers[layerNum].size() - 1; ++n) {
            m_layers[layerNum][n].feedForward(b);
        }
    }
}

Net::Net(const vector<unsigned> &topology)
{
    unsigned numLayers = topology.size();
    for (unsigned layerNum = 0; layerNum < numLayers; ++layerNum) {
        m_layers.push_back(Layer());
        unsigned numOutputs = layerNum == topology.size() - 1 ? 0 : topology[layerNum + 1];
    }
}
```

Artificial Intelligence & Programming

- Second approach: implement in C++, Python or Java, using libraries
- Example: learning a classifier using scikitlearn
- Advantages:
 - Common functionality is easy to use: it is hidden in the “black box” of the library
 - Common tasks are executed efficiently: a good library is implemented efficiently
- Disadvantages:
 - Many libraries are not programmable and can hence only be used to perform common tasks

```
from sklearn import datasets, svm, metrics

digits = datasets.load_digits()

n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# We learn the digits on the first half of the digits
classifier.fit(data[:n_samples / 2],
              digits.target[:n_samples / 2])

# Now predict the value of the digit on the second
half:
expected = digits.target[n_samples / 2:]
predicted = classifier.predict(data[n_samples / 2:])

print("Classification report for classifier %s:\n%s"
      % (classifier,
         metrics.classification_report(expected,
                                       predicted)))
print("Confusion matrix:\n%s" %
      metrics.confusion_matrix(expected, predicted))
```

Artificial Intelligence & Programming

- Third approach: develop a specialized language for a range of tasks in artificial intelligence
- Example: solving sudoku in the MiniZinc language

	2	3					5	
			8	2		7	9	3
	6	4			9	8		
			2		7		4	
		9		8		1		
	4	2						
	8						3	
			6			2		1
4					1		8	

```

include "alldifferent.mzn";

int: S;
int: N = S * S;

set of int: PuzzleRange = 1..N;
set of int: SubSquareRange = 1..S;

array[1..N,1..N] of 0..N: start;
array[1..N,1..N] of var PuzzleRange: puzzle;

constraint forall(i,j in PuzzleRange)(
    if start[i,j] > 0 then puzzle[i,j] = start[i,j]
    else true endif );

constraint forall (i in PuzzleRange) (
    alldifferent( [ puzzle[i,j] | j in PuzzleRange ] ) );

constraint forall (j in PuzzleRange) (
    alldifferent( [ puzzle[i,j] | i in PuzzleRange ] ) );

constraint forall (a, o in SubSquareRange)(
    alldifferent( [ puzzle[(a-1) * S + a1,
                      (o-1)*S + o1] | a1, o1 in SubSquareRange ] ) );

solve satisfy;

```

Artificial Intelligence & Programming

- Third approach: develop a specialized language for a range of tasks in artificial intelligence
- Example: solving sudoku in the MiniZinc language
- Advantages:
 - The range of tasks in the scope of the language is implemented easily
 - The range of tasks in the scope of the language is implemented reasonably efficiently
- Disadvantages:
 - The programmer needs to learn a new language
 - The programs are not as efficient as highly optimized code written in assembler or C by a highly skilled programmer

```
include "alldifferent.mzn";

int: S;
int: N = S * S;

set of int: PuzzleRange = 1..N;
set of int: SubSquareRange = 1..S;

array[1..N,1..N] of 0..N: start;
array[1..N,1..N] of var PuzzleRange: puzzle;

constraint forall(i,j in PuzzleRange)(
    if start[i,j] > 0 then puzzle[i,j] = start[i,j]
    else true endif );

constraint forall (i in PuzzleRange) (
    alldifferent( [ puzzle[i,j] | j in PuzzleRange ] ) );

constraint forall (j in PuzzleRange) (
    alldifferent( [ puzzle[i,j] | i in PuzzleRange ] ) );

constraint forall (a, o in SubSquareRange)(
    alldifferent( [ puzzle[(a-1) *S + a1,
    (o-1)*S + o1] | a1, o1 in SubSquareRange ] ) );

solve satisfy;
```

Artificial Intelligence & Programming

- Combined second and third approach:
embed a specialized language using libraries
in a general purpose language
- Example: solving sudoku using Numberjack
and Python
- Advantages:
 - The range of tasks in the scope of the language is still
implemented relatively easily
 - The range of tasks in the scope of the language is
implemented reasonably efficiently
- Disadvantages:
 - The code can be less elegant than that of a specialized
language
 - The programs are not as efficient as highly optimized
code written in assembler or C by a highly skilled
programmer

```

from Numberjack import *
import puzzles

start = puzzles.read ()
S = start.size ()
N = S * S
puzzle = Matrix(N, N, 1, N)

sudoku = Model ( \

    [ puzzle[i,j] == start[i,j]
      for i in range(0,N)
      for j in range(0,N)
      if start[i,j] > 0 ],

    [AllDiff(row) for row in puzzle.row],

    [AllDiff(col) for col in puzzle.col],

    [AllDiff(matrix[i:i+S, j:j+S])
      for i in range(0, N, S)
      for j in range(0, N, S)]

)

solver = sudoku.load ( "Mistral" )
solver.solve ()

```

Focus of This Lecture Series

- Specialized programming languages for artificial intelligence
- Embedded languages for artificial intelligence
- Algorithms used to execute these languages
- Examples of how these languages can be used

Constraint Programming

- Programming languages for solving *constraint satisfaction* and *constraint optimization* problems
- Constraint satisfaction problems are problems of the form:
 - **Given**
 - **Find one** assignment to *variables*
 - **Such that constraints on the variables are satisfied**

	2	3					5	
			8	2		7	9	3
	6	4			9	8		
			2		7		4	
		9		8		1		
	4	2						
	8						3	
			6			2		1
4					1		8	

Constraint Programming

- **Given:** a puzzle *start*
- **Find:** assignment to variables

$E_{11} \dots E_{99}$

- **Such that:**
 - Variables have values in the domains
 $E_{ij} \in \{1 \dots 9\}$ if $start_{ij} = 0$
 $E_{ij} = start_{ij}$ if $start_{ij} > 0$
 - These constraints are satisfied:
 $all_different([E_{11}, E_{12}, \dots, E_{19}])$
 $all_different([E_{21}, E_{22}, \dots, E_{29}])$
 \dots
 $all_different([E_{11}, E_{21}, \dots, E_{91}])$
 \dots
 $all_different([E_{11}, E_{22}, \dots, E_{33}])$
 \dots

	2	3					5	
			8	2		7	9	3
	6	4			9	8		
E_{41}	E_{42}		2		7		4	
E_{51}	E_{52}	9		8		1		
	4	2						
	8						3	
			6			2		1
4					1		8	

Constraint Programming

	2	3					5	
			8	2		7	9	3
	6	4			9	8		
E_{41}	E_{42}		2		7		4	
E_{51}	E_{52}	9		8		1		
	4	2						
	8						3	
			6			2		1
4					1		8	

```
include "alldifferent.mzn";
```

```
int: S;  
int: N = S * S;
```

Given

```
set of int: PuzzleRange = 1..N;  
set of int: SubSquareRange = 1..S;
```

```
array[1..N,1..N] of 0..N: start;
```

```
array[1..N,1..N] of var PuzzleRange: E;
```

Variables

```
constraint forall(i,j in PuzzleRange)(  
    if start[i,j] > 0 then E[i,j] = start[i,j]  
    else true endif );
```

Constraint: start puzzle

```
constraint forall (i in PuzzleRange) (  
    alldifferent( [ E[i,j] | j in PuzzleRange ] ) );
```

```
constraint forall (j in PuzzleRange) (  
    alldifferent( [ E[i,j] | i in PuzzleRange ] ) );
```

```
constraint forall (a, o in SubSquareRange)(  
    alldifferent( [ E[(a-1)*S + a1,  
    (o-1)*S + o1] | a1, o1 in SubSquareRange ] ) );
```

Constraints: different numbers in rows, columns, blocks

```
solve satisfy;
```

Constraint Programming

- Dutch railways introduced a new schedule in 2006 after a failed attempt to introduce “a circle around the church”
- Annual profits increased with €40 million after the introduction of the schedule
- The *periodic event scheduling* problem (scheduling of the trains; personel is not scheduled periodically) was solved using a constraint programming system



Constraint Programming

- Links shippers to a network of 200 shipping lines with connections to 600 ports in 123 countries
- Assign yard locations and loading plans under various operational and safety requirements
- Solution: Yard planning system, based on constraint programming





Probabilistic Programming

- 1) The probability of breast cancer is 1% for a woman at 40 who participates in a routine screening.
- 2) If a woman has breast cancer, the probability is 80% that she will have a positive mammography.
- 3) If a woman does not have breast cancer, the probability is 9.6% that she will also have a positive mammography.
- 4) A woman in this age group had a positive mammography in a routine screening.
- 5) What is the probability that she actually has breast cancer?

$$\frac{0.8 \times 0.01}{0.8 \times 0.01 + 0.096 \times 0.99} = 0.078$$



Probabilistic Programming

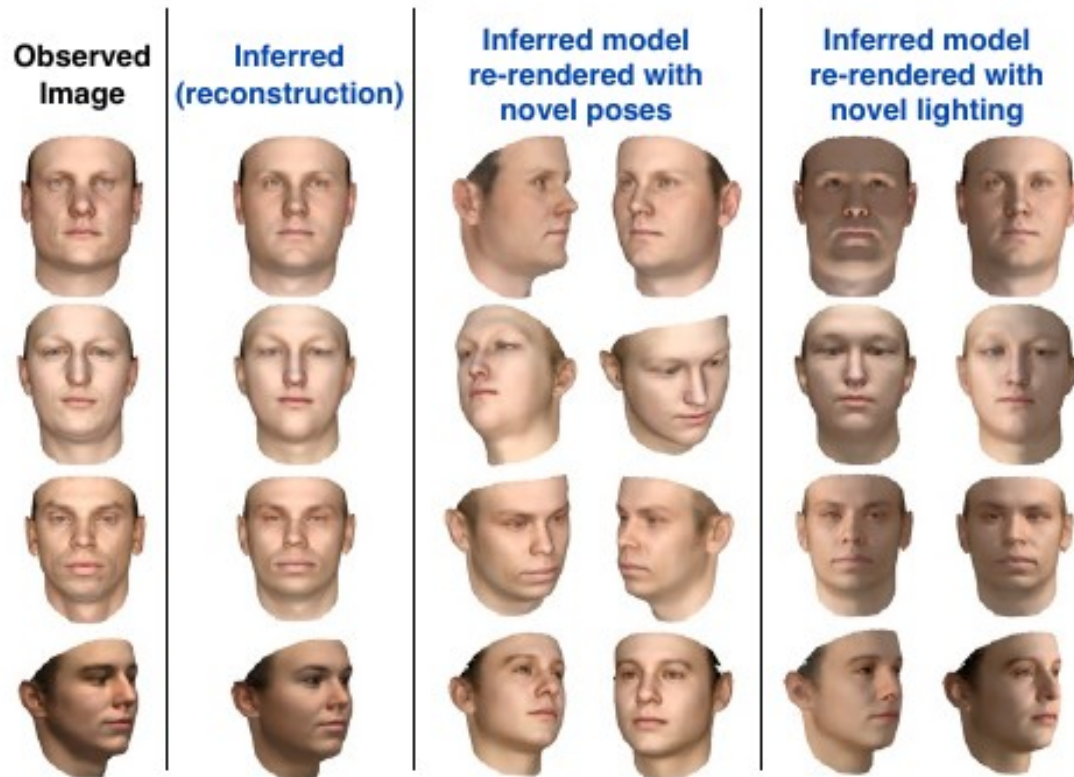
- 1) The probability of breast cancer is 1% for a woman at 40 who participates in a routine screening.
- 2) If a woman has breast cancer, the probability is 80% that she will have a positive mammography.
- 3) If a woman does not have breast cancer, the probability is 9.6% that she will also have a positive mammography.
- 4) A woman in this age group had a positive mammography in a routine screening.
- 5) What is the probability that she actually has breast cancer?

```
(define samples
  (mh-query 100 100

    (define breast-cancer (flip 0.01))      1)
    (define positive-mammogram
      (if breast-cancer
          (flip 0.8)                          2)
          (flip 0.096)))                    3)
    breast-cancer                             5)
    positive-mammogram                         4)
  )
  )
(hist samples "breast cancer")
```

Probabilistic Programming

Generate 3D faces based on 2D pictures of a face



```
function PROGRAM(MU, PC, EV, VERTEX_ORDER)
    face=Dict(); shape = []; texture = [];
    for S in ["shape", "texture"]
        for p in ["nose", "eyes", "outline", "lips"]
            coeff = MvNormal(0,1,1,99)
            face[S][p] = MU[S][p]+PC[S][p].*(coeff.*EV[S][p])
        end
    end
    shape = face["shape"][:]; tex = face["texture"][:];
    camera = Uniform(-1,1,1,2); light = Uniform(-1,1,1,2)
    rendered_img = MeshRenderer(shape,tex,light,camera)
    ren_ftrs = getFeatures("CNN_Conv6", rendered_img)
    observe(MvNormal(0,0.01), rendered_img-obs_img)
    observe(MvNormal(0,10), ren_ftrs-obs_cnn)
end

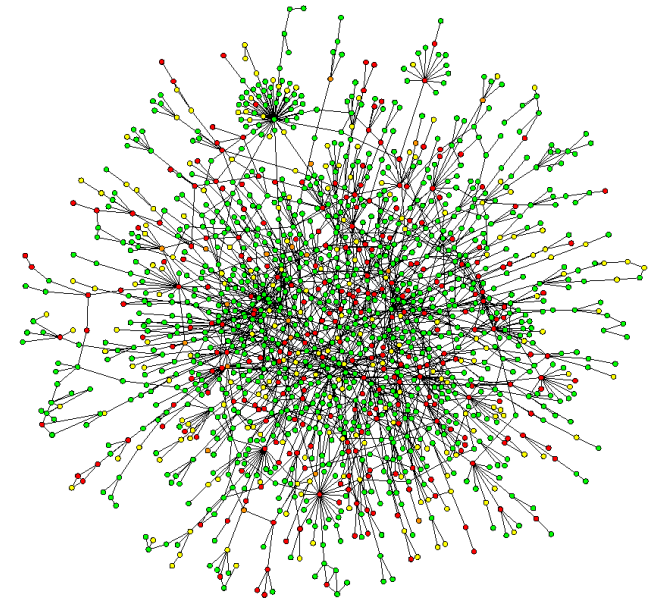
global obs_img = imread("test.png")
global obs_cnn = getFeatures("CNN_Conv6", img)
TR = trace(PROGRAM,args=[MU,PC,EV,VERTEX_ORDER])
learn_datadriven_proposals(TR,100000,"CNN_Conv6")
load_proposals(TR)
infer(TR,CB,20,["DATA-DRIVEN"])
infer(TR,CB,200,["ELLIPTICAL"])
```

Specification
of distributions

Observations

Probabilistic Programming

- Computer Vision
- Spam filters
- Speech recognition
- Bioinformatics
- Network analysis



“Algebraic Programming”

- Neural networks, sum product networks can be seen as large algebraic expressions
- Can we write such expressions down in a simple language and automatically optimize them?
 - Running independent expressions in parallel
 - Performing optimizations
 $x/y*y = x$
 - Common subexpression elimination
 - Symbolic differentiation
 $f(x) = x^2$
 $df/dx = 2x$

```
import theano
from theano import tensor

# declare two symbolic floating-point
scalars
a = tensor.dscalar()
b = tensor.dscalar()

# create a simple expression
c = a + b

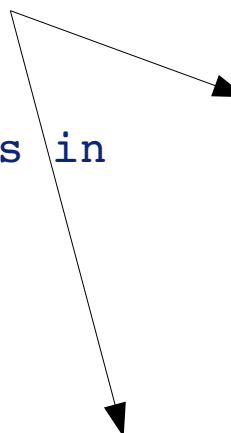
# convert the expression into a callable
# object that takes (a,b)
# values as input and computes a value for c
f = theano.function([a,b], c)

# bind 1.5 to 'a', 2.5 to 'b', evaluate 'c'
assert 4.0 == f(1.5, 2.5)
```

“Algebraic Programming”

```
import tensorflow as tf
import numpy as np
```

Declare the learning
problem



```
# Create 100 phony x, y data points in
# NumPy, y = x * 0.1 + 0.3
x_data = np.random.rand(100).
           astype("float32")
y_data = x_data * 0.1 + 0.3
```

```
# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y -
y_data))
optimizer = \
    tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the
# variables. We will 'run' this first.
init = tf.initialize_all_variables()
```

```
# Try to find values for W and b that
# compute y_data = W * x_data + b
# (We know that W should be 0.1 and b 0.3,
# Tensorflow will figure that out for us.)
W = tf.Variable(tf.random_uniform([1],
    -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b
```

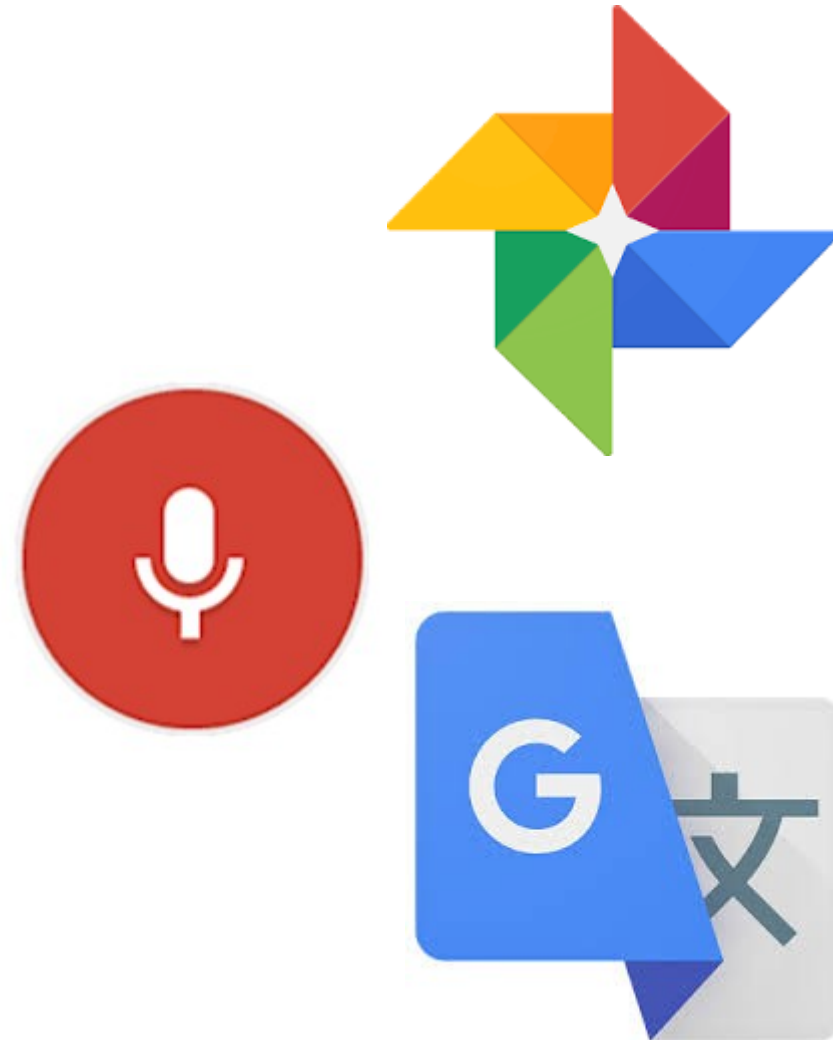
```
# Launch the graph.
sess = tf.Session()
sess.run(init)
```

Solves the learning
problem

```
# Fit the line.
for step in xrange(201):
    sess.run(train)
    if step % 20 == 0:
        print step, sess.run(W),
sess.run(b)
```

“Algebraic Programming”

- Google Photos
- Google Voice Search
- Google Translate
- Handwriting recognition



Artificial Intelligence ↔ Declarative Programming

- Declarative programming allows to program **WHAT** the computer has to do:
 - Find a solution to a constraint satisfaction problem
 - Calculate the probability of a certain event given observations
 - Find the optimum to a numerical optimization problem with a well-defined loss function
- Does not specify **HOW** the computer has to do this

Programming Paradigms in AI

Two main traditional programming paradigms in AI:

- Logic programming
- Functional programming

Logic Programming

Developed in 1972-1973 to allow the implementation of intelligent systems that one could ask questions to

Every psychiatrist is a person.

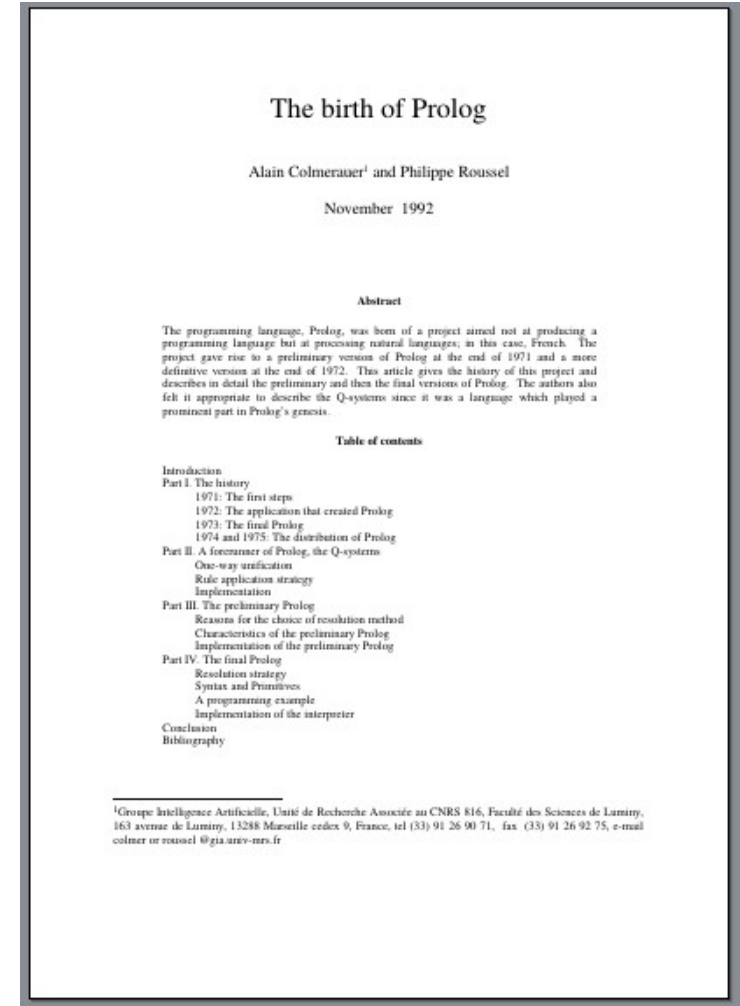
Every person he analyzes is sick.

Jacques is a psychiatrist in Marseille.

Is Jacques a person?

Where is Jacques?

Is Jacques sick?



Logic Programming

Most popular language is Prolog

Prolog is the basis for:

- Constraint logic programming systems
- Probabilistic logic programming systems
- Algebraic logic programming systems

Functional Programming

- Development started in 1958: LISP (LISt Processing)
- Program and data consists of lists that are manipulated using functions
- Expressions perform *symbolic manipulation*
- Most of early AI was symbolic
- LISP was used for instance in:
 - The Dynamic Analysis and Replanning Tool used during the first Gulf War to plan military movements
 - SPIKE, the planning and scheduling application for the Hubble Space Telescope
 - American Express Authorizer's Assistant, checking credit card transactions in the early 1990s
- Functional programming is the basis for
 - Probabilistic functional programming
 - Functional reactive programming

Course Overview

1) Lectures

- Concepts of logic programming, functional programming
- Introduction to constraint programming
- Introduction to probabilistic programming
- Introduction to algebraic programming

2) Paper presentations

3) Practical exercise

Grade is determined by

- Paper presentation: 20%
- Active participation in paper presentations: 10%
- Exercise and report: 70%

Practical Information

- Tuesdays at 11:15 in room 403
- Website: www.liacs.leidenuniv.nl/home/snijssen/PSAI
- Room number: 110
- E-mail address: s.nijssen@liacs.leidenuniv.nl
- No book: we use papers
- Marked as “SCDM” in the schedule...

Context in Leiden

- Builds upon two bachelor courses in particular:
 - Concepts of programming languages
 - Artificial intelligence
- Is related to many other master courses:
 - Bayesian networks
 - Neural networks
 - Databases and data mining
 - Advances in data mining
 - Seminar distributed data mining
 - Multicriteria Optimization and Decision Analysis
 - Seminar combinatorial algorithms
 - Parallel algorithms