

Artificial Intelligence

aswin.van.woudenberg@nhl.nl

NHL University of Applied Sciences

Table of Contents

What is artificial intelligence?

- Four approaches to artificial intelligence

- Artificial intelligence within the curriculum

- Philosophy of mind

- AI in fiction

- Approaches to AI

Chatbots and dialogue systems

Programming in Logic: An introduction to Prolog

- Getting started

- Facts, rules and queries

- Arithmetic

- Lists

- Definite Clause Grammars

Final course project

Definition of artificial intelligence

Definition

Artificial intelligence is the science and engineering of making intelligent machines¹.

¹John McCarthy, "What Is Artificial Intelligence?"

Definition of artificial intelligence

Definition of artificial intelligence

The art of creating machines that perform functions that require intelligence when performed by people.

– Ray Kurzweil

Definition of artificial intelligence

The art of creating machines that perform functions that require intelligence when performed by people.

– Ray Kurzweil

The branch of computer science that is concerned with the automation of intelligent behavior.

– George Luger and William Stubblefield

Definition of artificial intelligence

The art of creating machines that perform functions that require intelligence when performed by people.

– Ray Kurzweil

The branch of computer science that is concerned with the automation of intelligent behavior.

– George Luger and William Stubblefield

[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning...

– Richard Bellman

Definition of artificial intelligence

The art of creating machines that perform functions that require intelligence when performed by people.

– Ray Kurzweil

The branch of computer science that is concerned with the automation of intelligent behavior.

– George Luger and William Stubblefield

[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning...

– Richard Bellman

The study of the computations that make it possible to perceive, reason, and act.

– Patrick Winston

Definition of artificial intelligence

	Humanly	Rationally
Acting	<p>The art of creating machines that perform functions that require intelligence when performed by people.</p> <p>– Ray Kurzweil</p>	<p>The branch of computer science that is concerned with the automation of intelligent behavior.</p> <p>– George Luger and William Stubblefield</p>
Thinking	<p>[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning...</p> <p>– Richard Bellman</p>	<p>The study of the computations that make it possible to perceive, reason, and act.</p> <p>– Patrick Winston</p>

Four approaches to artificial intelligence

	Humanly	Rationally
Acting	Acting humanly: The Turing Test approach	Acting rationally: The rational agent approach
Thinking	Thinking humanly: The cognitive modeling approach	Thinking rationally: The "laws of thought" approach

Acting humanly: The Turing Test² approach

TURING TEST EXTRA CREDIT:
CONVINCE THE EXAMINER
THAT HE'S A COMPUTER.

YOU KNOW, YOU MAKE
SOME REALLY GOOD POINTS.

I'M ... NOT EVEN SURE
WHO I AM ANYMORE.



²Alan Turing, "Computing Machinery and Intelligence"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Acting humanly: The Turing Test approach

A computer passing the Turing Test would require the following capabilities:

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning

To pass the so-called Total Turing Test³ a computer would also need:

- Computer vision
- Robotics

³Stevan Harnad, "Other Bodies, Other Minds: A Machine Incarnation of an Old Philosophical Problem"

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking humanly: The cognitive modeling approach

- Design programs that think like humans
- Need to determine human thought processes
 - Introspection
 - Human experimentation
- A cognitive architecture summarizes the various results of cognitive psychology in a comprehensive computer model
 - ACT-R
 - Soar
 - EPIC
 - ICARUS
 - NARS
 - SNePS
 - OpenCog

Thinking rationally: The "laws of thought" approach

- An approach firmly grounded in logic
- Syllogism
 - Socrates is a man
 - All men are mortal
 - ∴ Socrates is mortal

Thinking rationally: The "laws of thought" approach

- An approach firmly grounded in logic
- Syllogism
 - Socrates is a man
 - All men are mortal
 - ∴ Socrates is mortal

Thinking rationally: The "laws of thought" approach

- An approach firmly grounded in logic
- Syllogism
 - Socrates is a man
 - All men are mortal
 - ∴ Socrates is mortal

Thinking rationally: The "laws of thought" approach

- An approach firmly grounded in logic
- Syllogism
 - Socrates is a man
 - All men are mortal
 - ∴ Socrates is mortal

Thinking rationally: The "laws of thought" approach

- An approach firmly grounded in logic
- Syllogism
 - Socrates is a man
 - All men are mortal
 - ∴ Socrates is mortal

Thinking rationally: The "laws of thought" approach

- An approach firmly grounded in logic
- Syllogism
 - Socrates is a man
 - All men are mortal
 - ∴ Socrates is mortal

Acting rationally: The rational agent approach

- Rational behavior: Doing the right thing
- The right thing: that which is expected to maximize goal achievement, given the available information
- Does not necessarily involve thinking - e.g., - blinking reflex - but should be in the service of rational action
- Agent: an entity that perceives and acts
- Abstractly, an agent is a function from percept histories to actions
 - $f : P^* \rightarrow A$

Acting rationally: The rational agent approach

- Rational behavior: Doing the right thing
- The right thing: that which is expected to maximize goal achievement, given the available information
- Does not necessarily involve thinking - e.g., - blinking reflex - but should be in the service of rational action
- Agent: an entity that perceives and acts
- Abstractly, an agent is a function from percept histories to actions
 - $f : P^* \rightarrow A$

Acting rationally: The rational agent approach

- Rational behavior: Doing the right thing
- The right thing: that which is expected to maximize goal achievement, given the available information
- Does not necessarily involve thinking - e.g., - blinking reflex - but should be in the service of rational action
- Agent: an entity that perceives and acts
- Abstractly, an agent is a function from percept histories to actions
 - $f : P^* \rightarrow A$

Acting rationally: The rational agent approach

- Rational behavior: Doing the right thing
- The right thing: that which is expected to maximize goal achievement, given the available information
- Does not necessarily involve thinking - e.g., - blinking reflex - but should be in the service of rational action
- Agent: an entity that perceives and acts
- Abstractly, an agent is a function from percept histories to actions
 - $f : P^* \rightarrow A$

Acting rationally: The rational agent approach

- Rational behavior: Doing the right thing
- The right thing: that which is expected to maximize goal achievement, given the available information
- Does not necessarily involve thinking - e.g., - blinking reflex - but should be in the service of rational action
- Agent: an entity that perceives and acts
- Abstractly, an agent is a function from percept histories to actions
 - $f : P^* \rightarrow A$

Acting rationally: The rational agent approach

- Rational behavior: Doing the right thing
- The right thing: that which is expected to maximize goal achievement, given the available information
- Does not necessarily involve thinking - e.g., - blinking reflex - but should be in the service of rational action
- Agent: an entity that perceives and acts
- Abstractly, an agent is a function from percept histories to actions
 - $f : P^* \rightarrow A$

Four approaches to artificial intelligence

	Humanly	Rationally
Acting	Acting humanly: The Turing Test approach	Acting rationally: The rational agent approach
Thinking	Thinking humanly: The cognitive modeling approach	Thinking rationally: The "laws of thought" approach

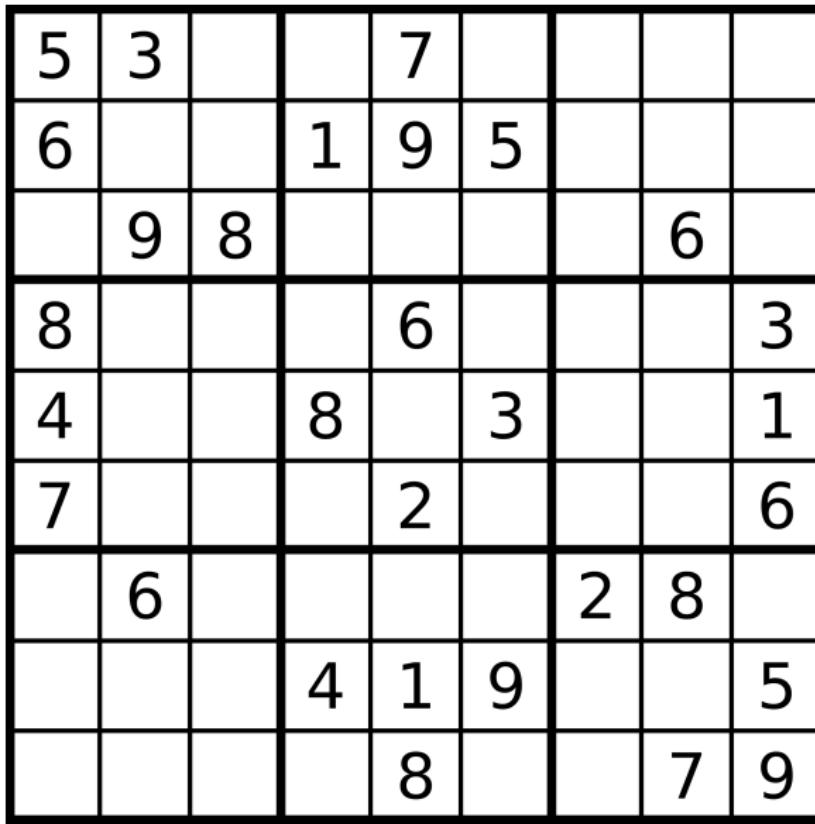
Sub-fields of artificial intelligence

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning
- Computer vision
- Robotics

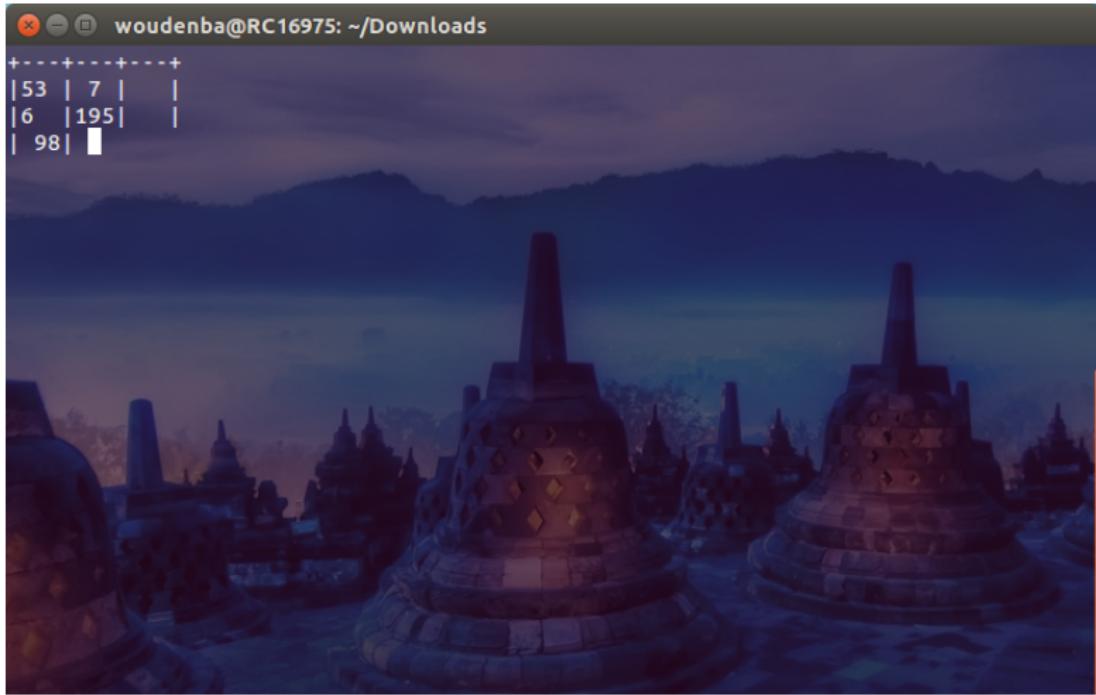
Artificial intelligence within the curriculum

- Programming I-III
- OOP
- Algorithmics
- Analytics
- Vision
- IDP
- Artificial Intelligence

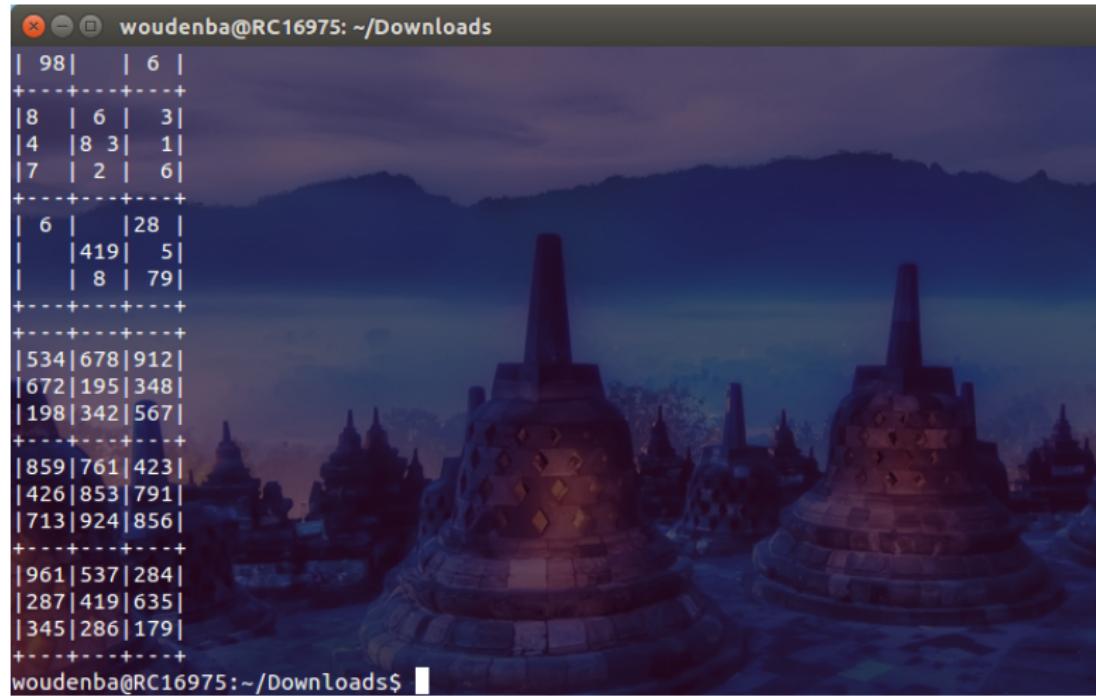
Sudoku solver



Sudoku solver



Sudoku solver



woudenba@RC16975: ~/Downloads

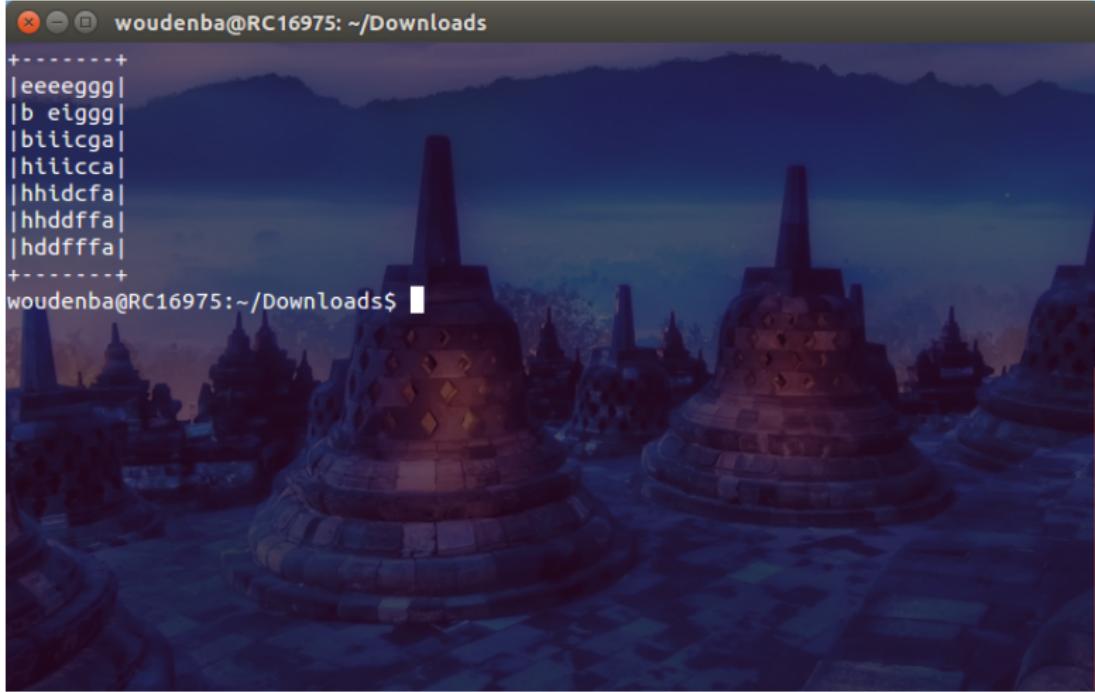
9	8		6					
8		6		3				
4		8	3	1				
7		2		6				
6			28					
		419		5				
		8		79				
534	678	912						
672	195	348						
198	342	567						
859	761	423						
426	853	791						
713	924	856						
961	537	284						
287	419	635						
345	286	179						

woudenba@RC16975:~/Downloads\$

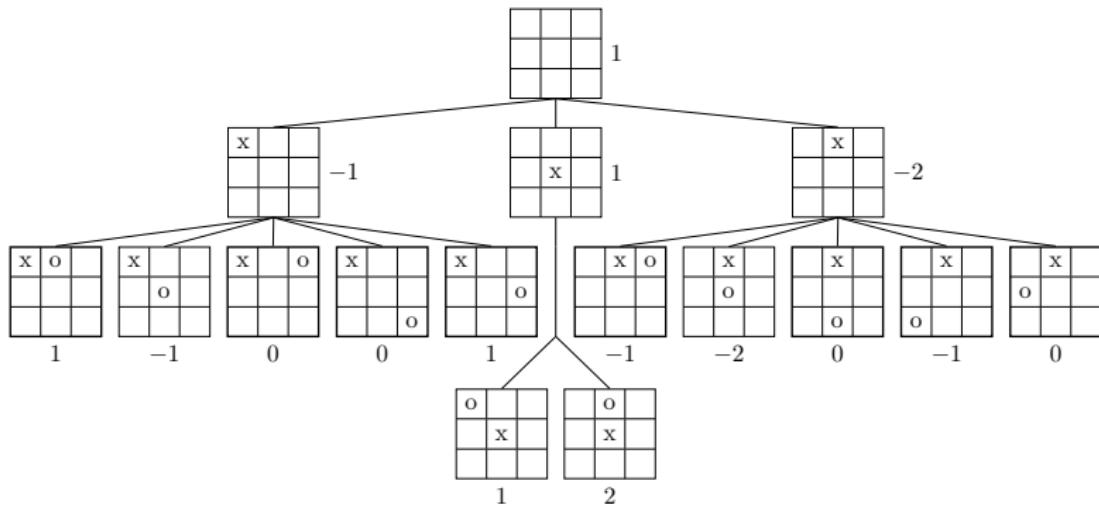
Skyline puzzle



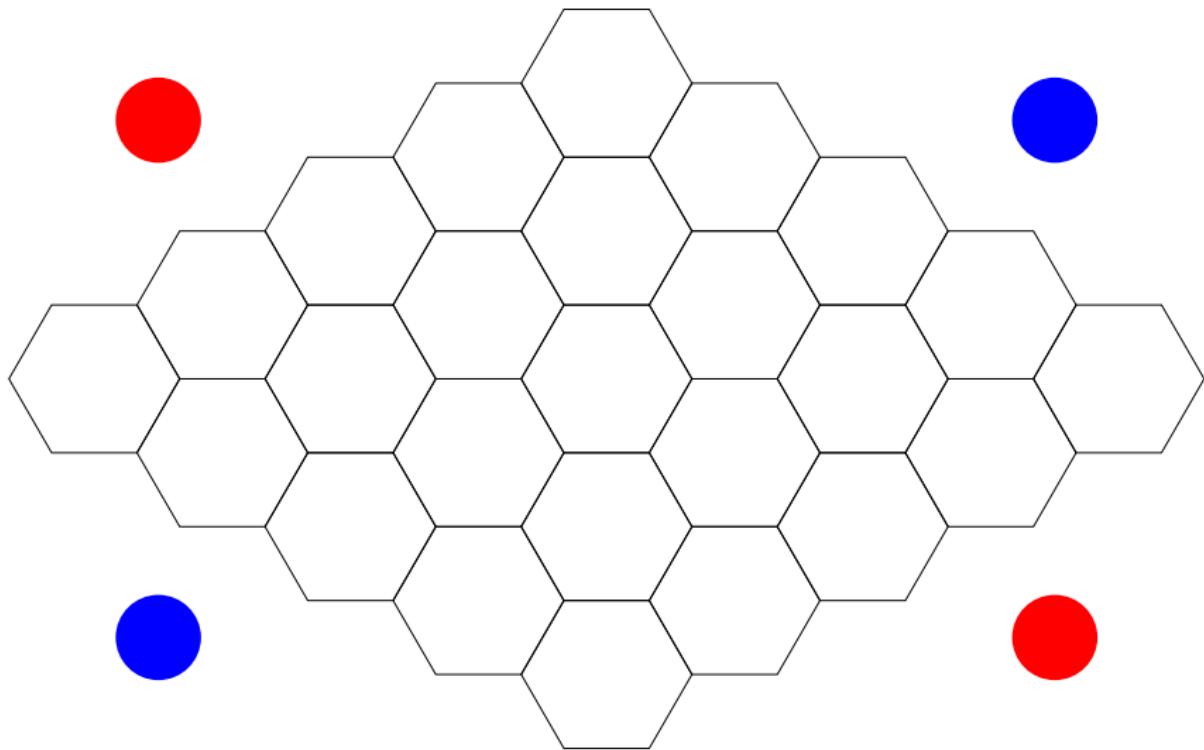
Skyline puzzle



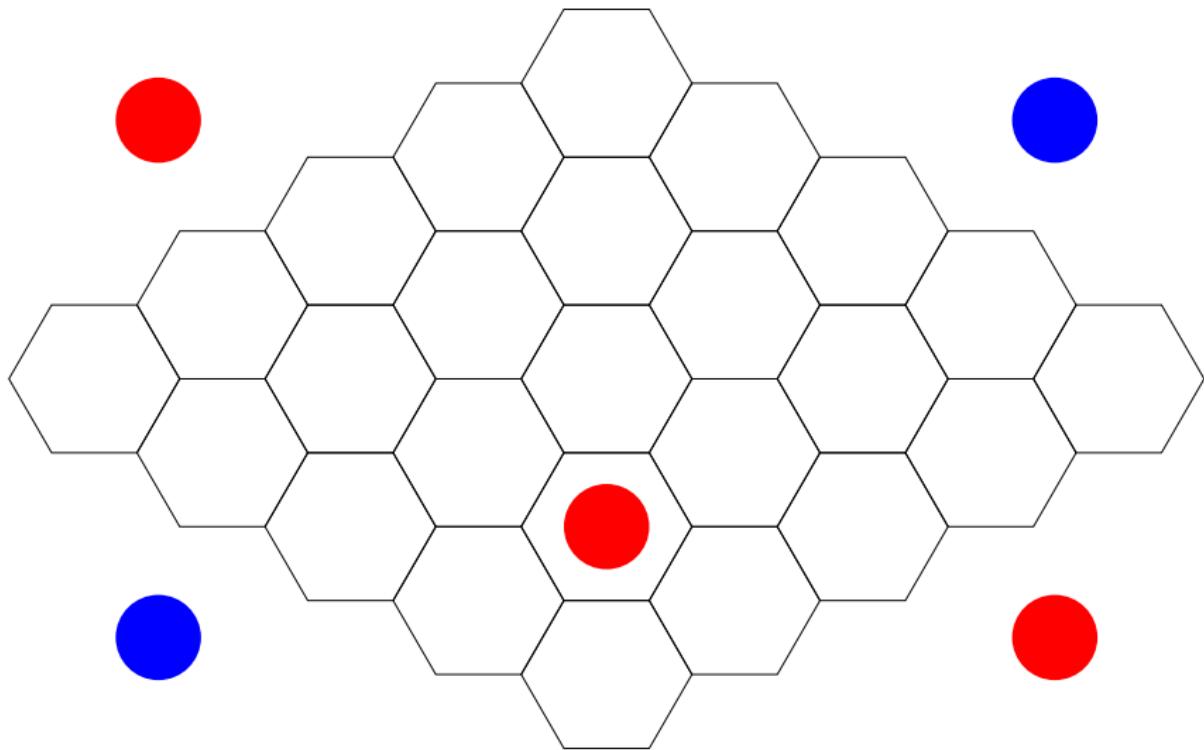
Game trees



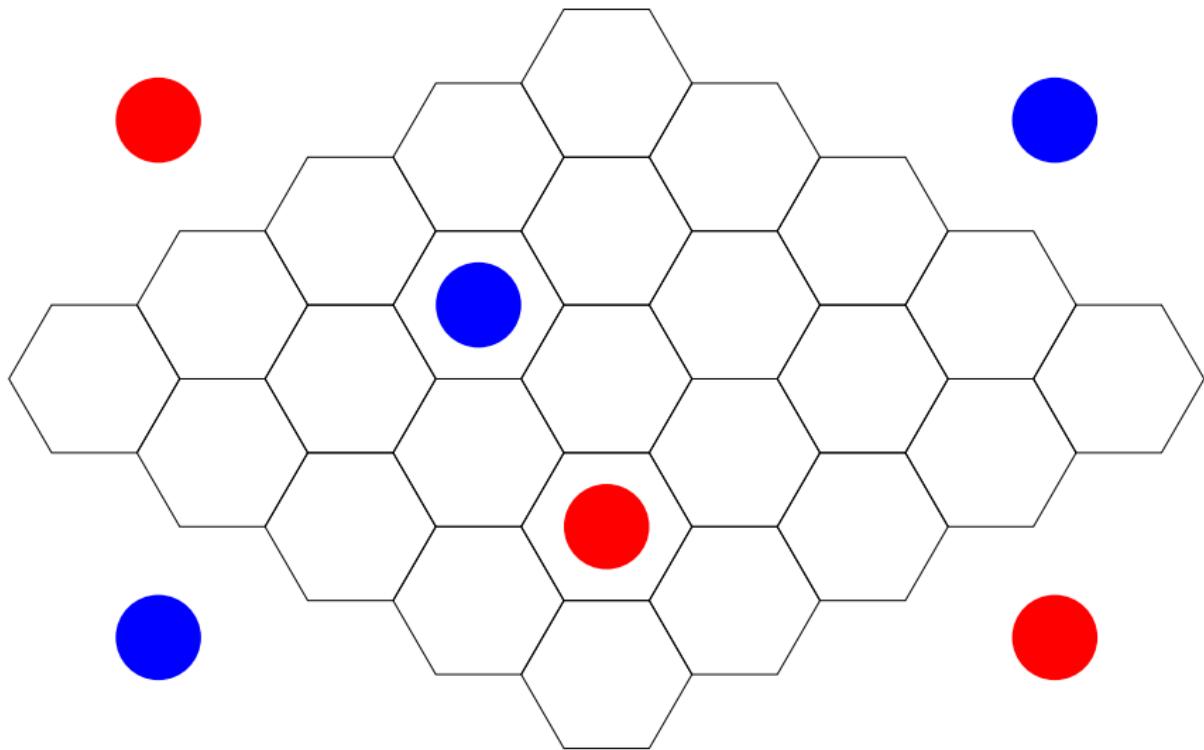
Hex



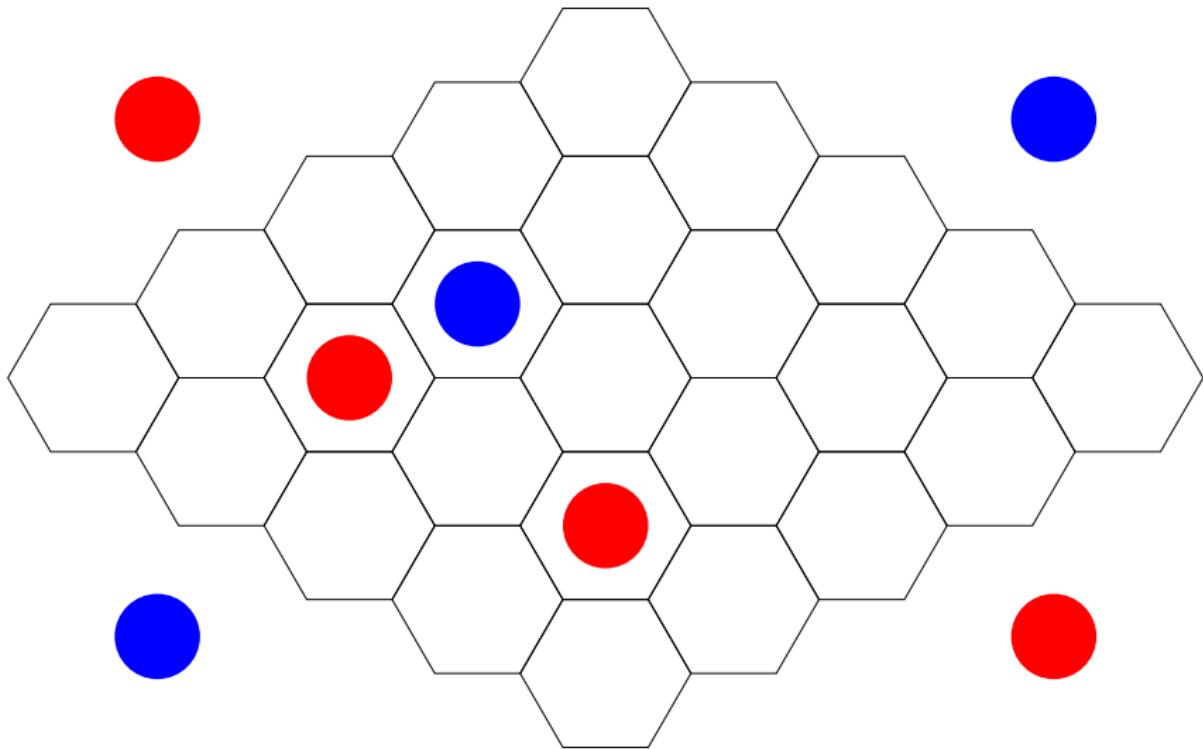
Hex



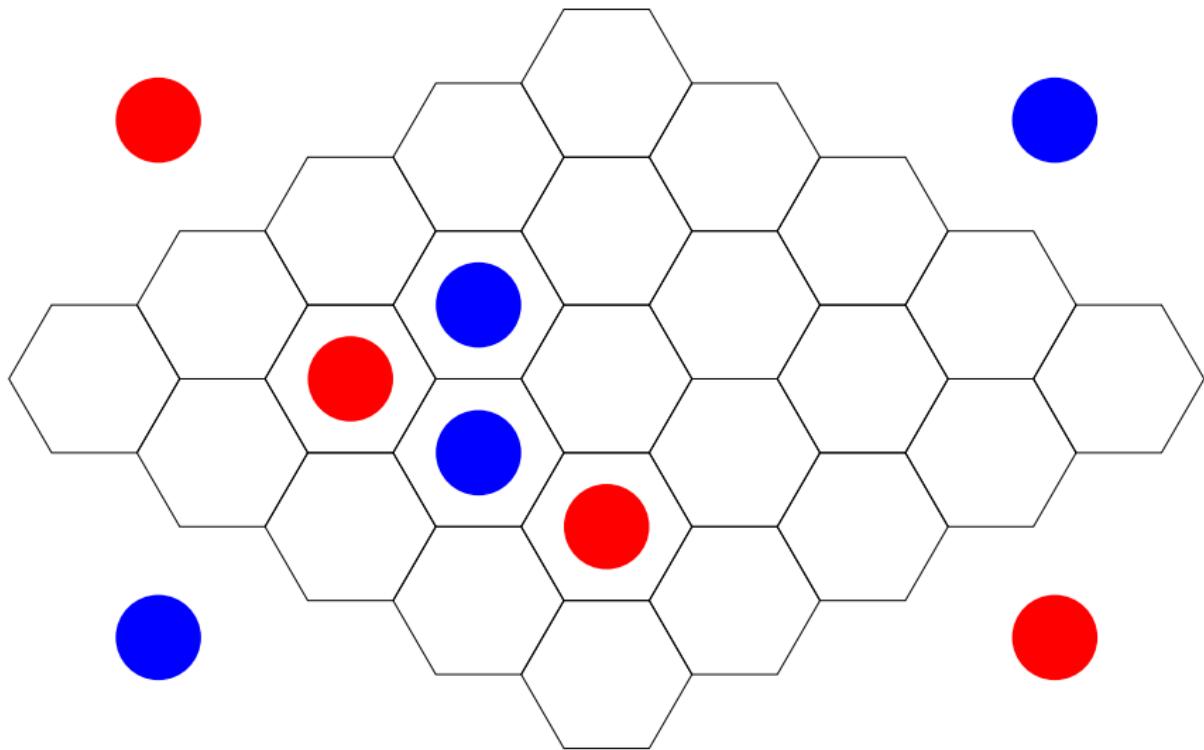
Hex



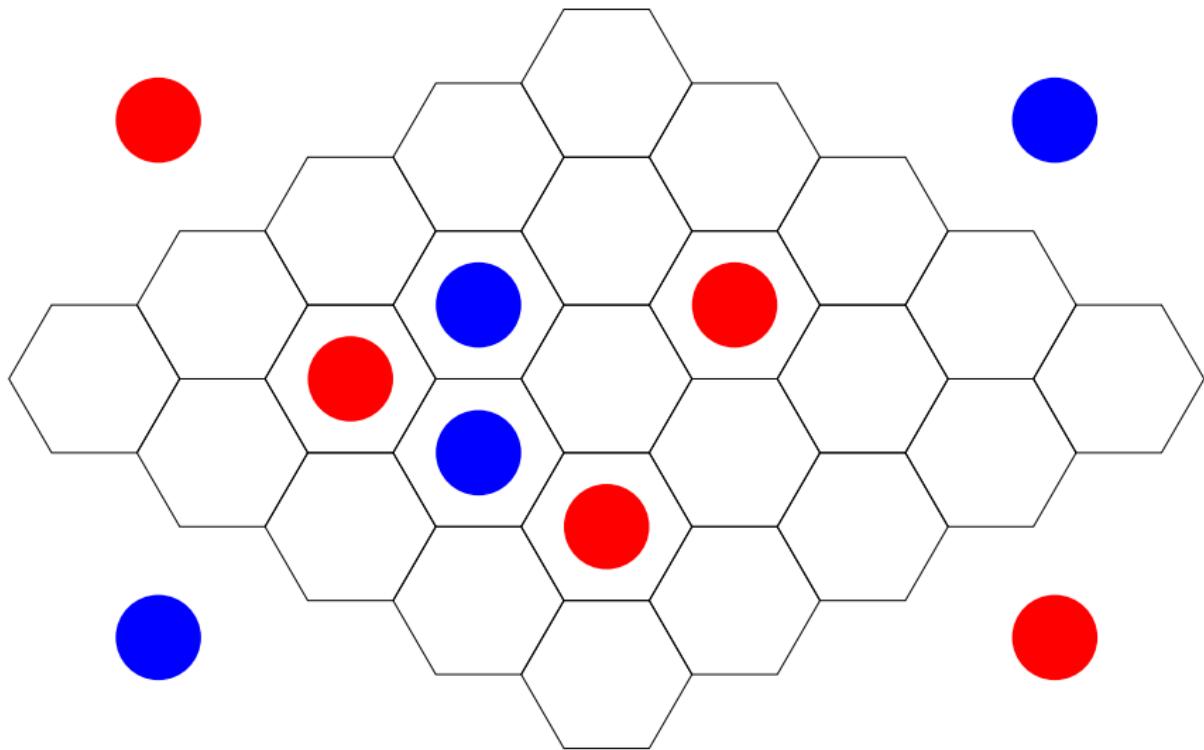
Hex



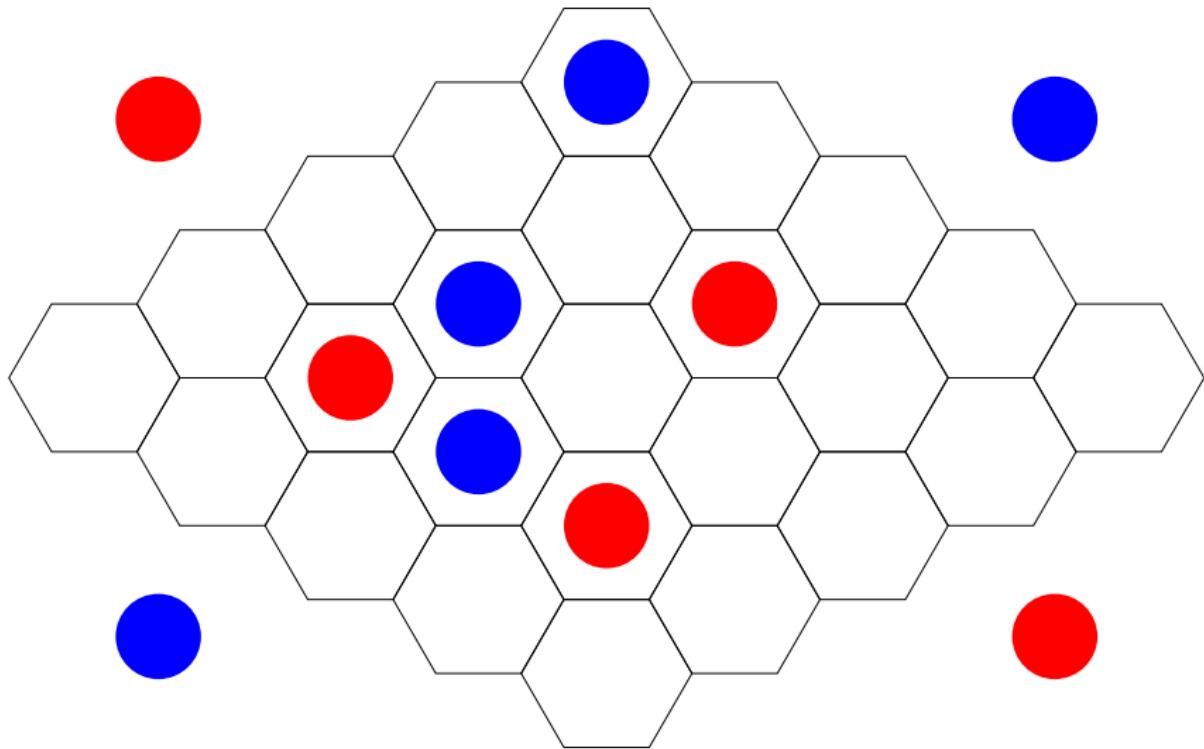
Hex



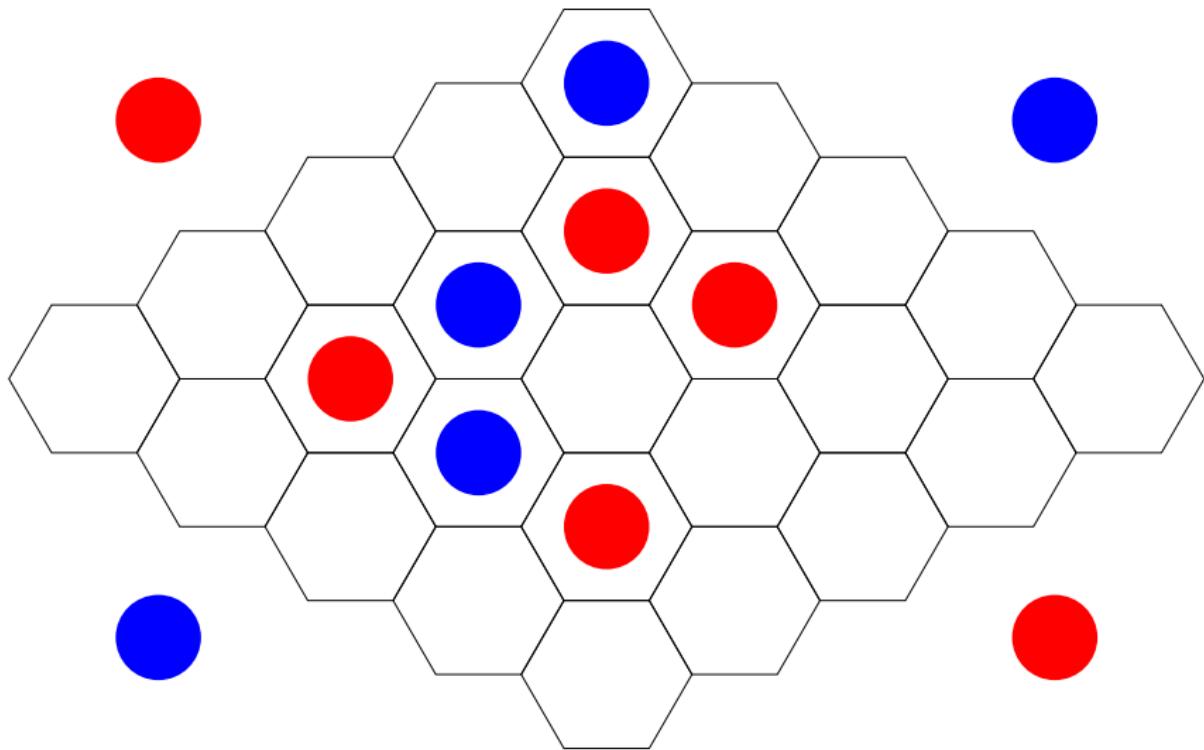
Hex



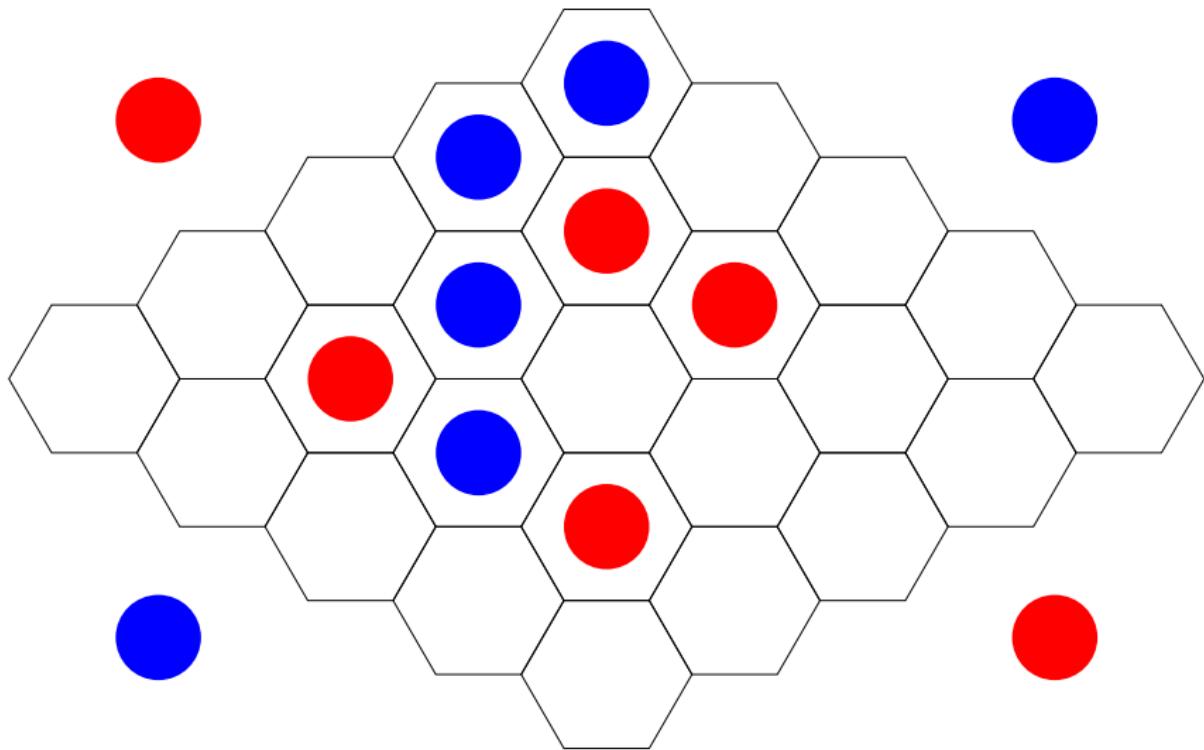
Hex



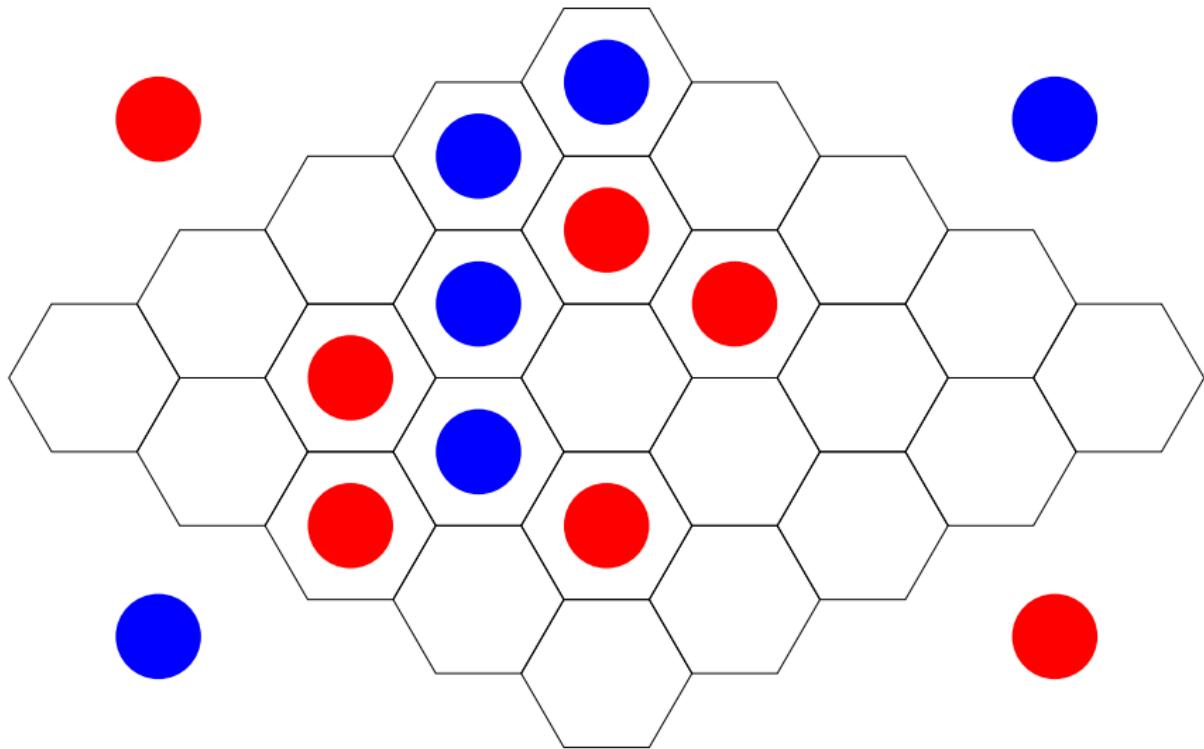
Hex



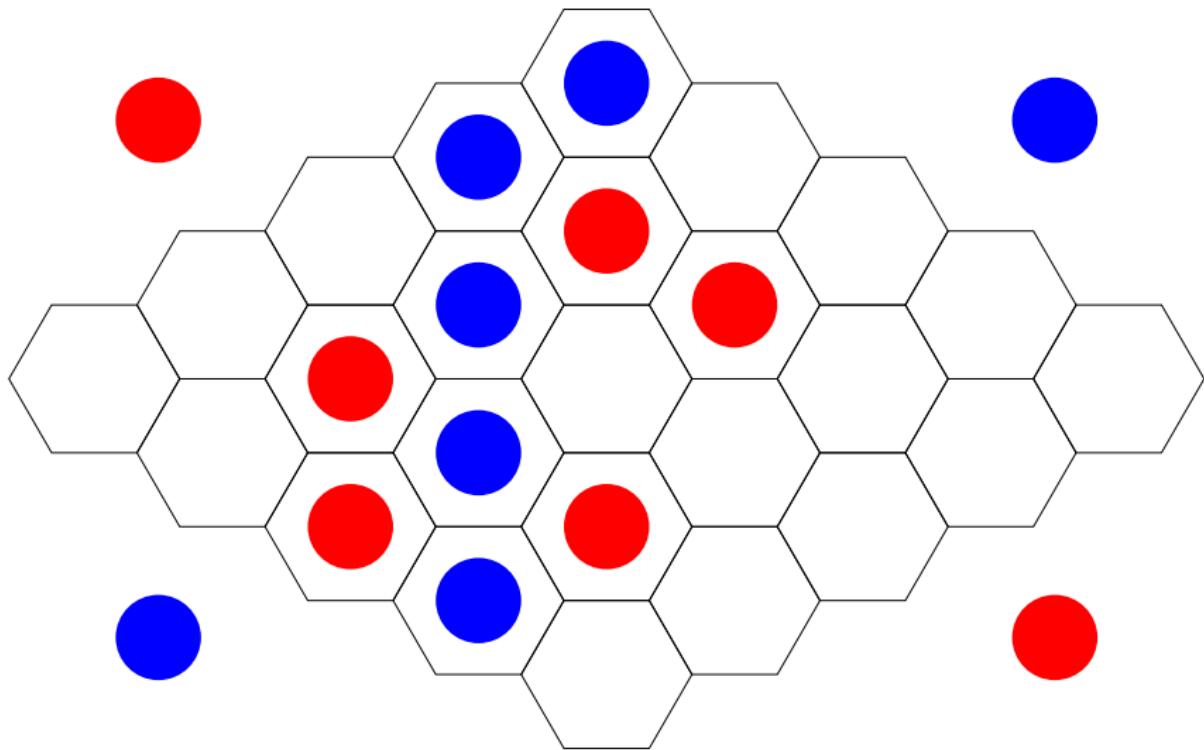
Hex



Hex



Hex

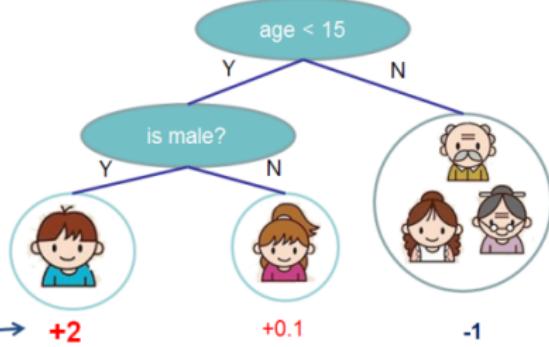


Machine learning

Input: age, gender, occupation, ...



Does the person like computer games



prediction score in each leaf → +2

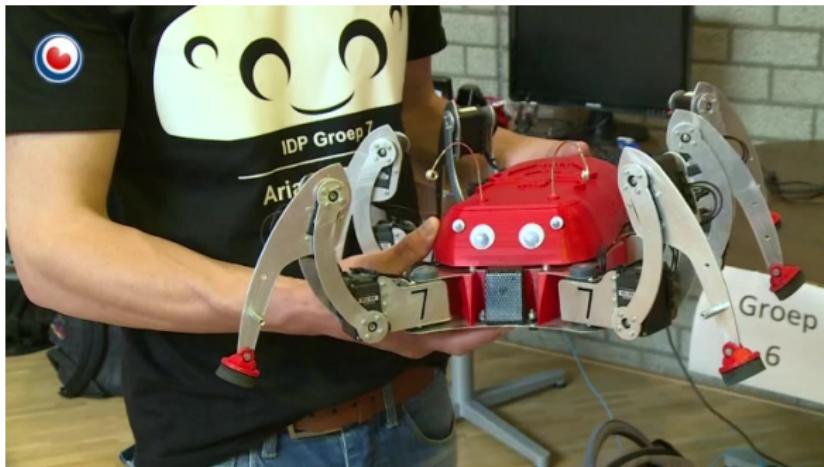
+0.1

-1

Computer vision



Robotics



Four approaches to artificial intelligence

	Humanly	Rationally
Acting	Acting humanly: The Turing Test approach	Acting rationally: The rational agent approach
Thinking	Thinking humanly: The cognitive modeling approach	Thinking rationally: The "laws of thought" approach

Sub-fields of artificial intelligence

- Natural language processing
- Knowledge representation
- Automated reasoning
- Machine learning
- Computer vision
- Robotics

Weak vs strong AI

Definition

Weak AI (also known as *narrow AI*) defines non-sentient computer intelligence, typically focused on a narrow task.

Definition

Strong AI generally refers to a machine with consciousness, sentience and mind.

Definition

Some researchers equate strong AI to *artificial general intelligence* which refers to a machine with the ability to apply intelligence to any problem, rather than just one specific problem.

Turing test²

TURING TEST EXTRA CREDIT:
CONVINCE THE EXAMINER
THAT HE'S A COMPUTER.

YOU KNOW, YOU MAKE
SOME REALLY GOOD POINTS.

I'M ... NOT EVEN SURE
WHO I AM ANYMORE.



²Alan Turing, "Computing Machinery and Intelligence"

Chinese room³



³John Searle, "Minds, Brains and Programs"

Turing test alternatives

Several alternatives for the Turing test have been conceived, for instance:

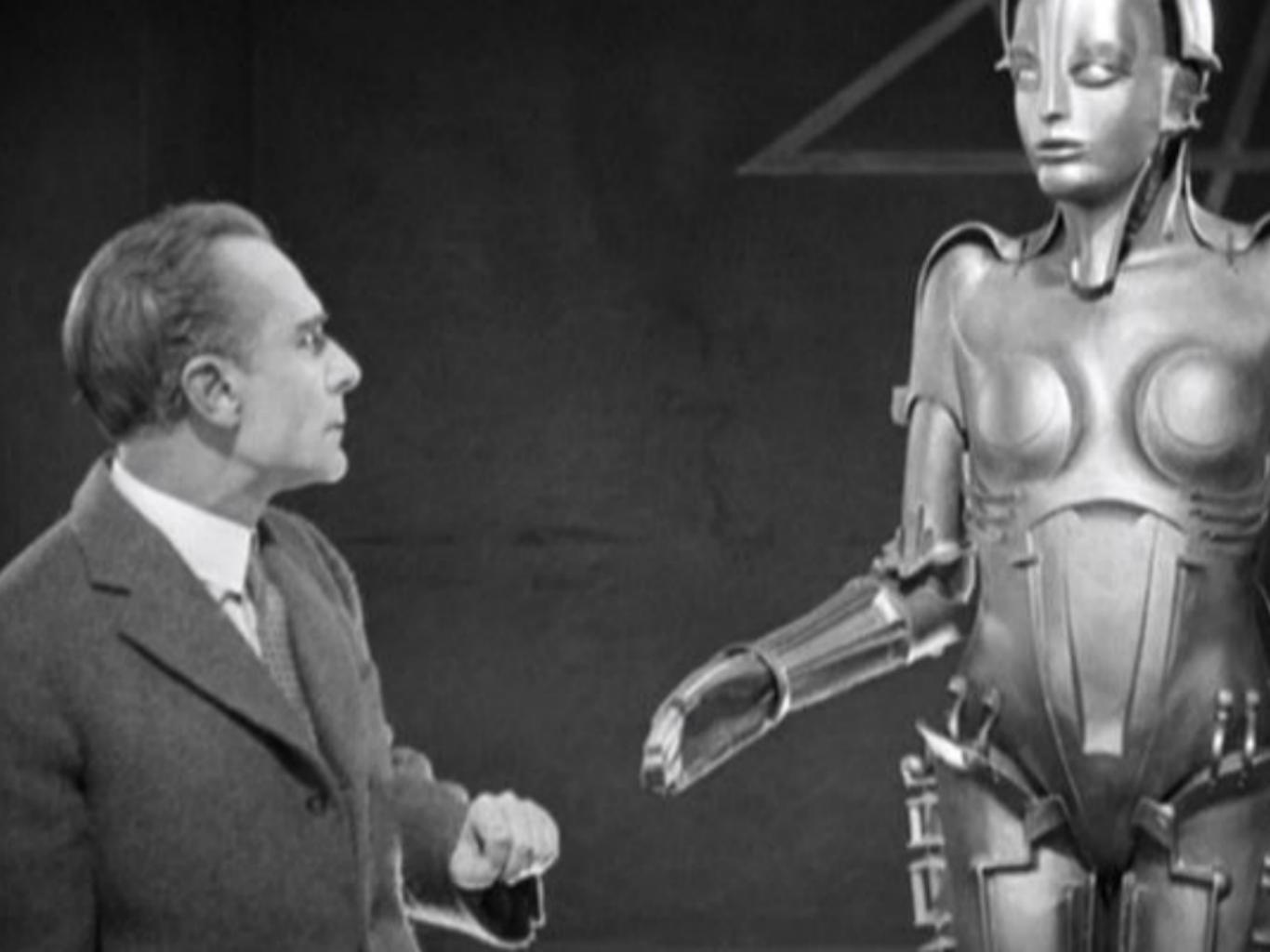
- The Wozniak Coffee Test⁴
Have a machine go into an average home and figure out how to make coffee.
- The Robot College Student Test⁴
A machine is given the task of enrolling in a university, taking and passing the same classes that humans would, and obtaining a degree.
- The Employment Test⁵
A machine is given the task of working an economically important job, and must perform as well or better than the level that humans perform at in the same job.

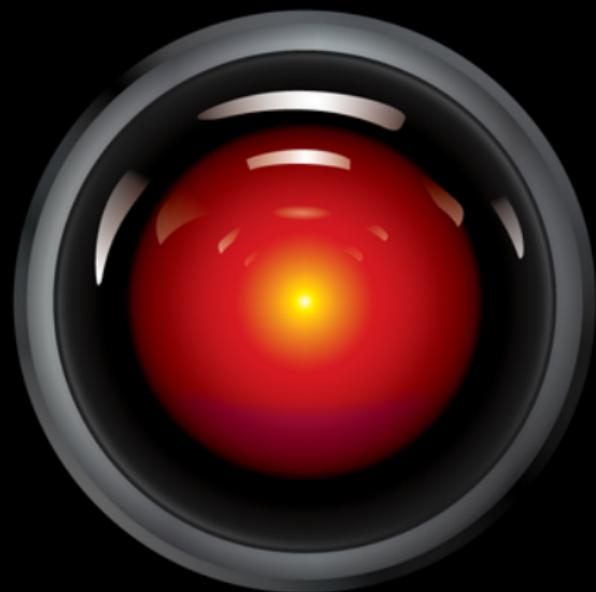
⁴Ben Goertzel, "The Architecture of Human-Like General Intelligence"

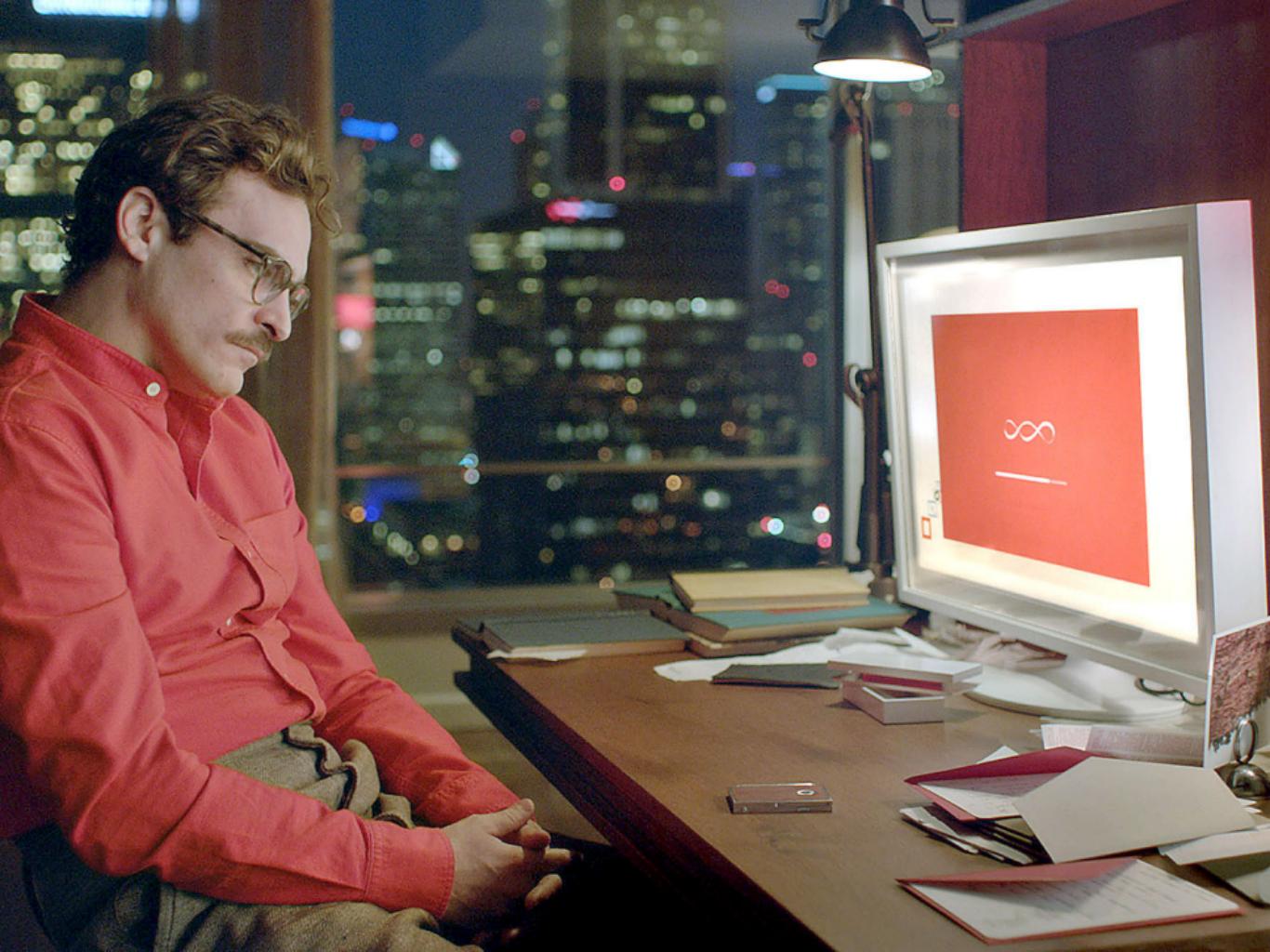
⁵Nils Nilsson, "Human-Level Artificial Intelligence? Be Serious!"

AI in fiction

- Metropolis*
- 2001: A Space Odyssey*
- A.I.: Artificial Intelligence
- I, Robot
- The Machine
- Her*
- Autómata
- Eva
- Big Hero 6
- Trancendence
- Blade Runner
- Chappie
- Avengers: Age of Ultron
- Äkta människor*
- Moon
- Short Circuit
- Battlestar Galactica
- Tron
- Terminator
- Ex Machina*
- Wall-E
- Colossus: The Forbin Project
- The Matrix
- D.A.R.Y.L.

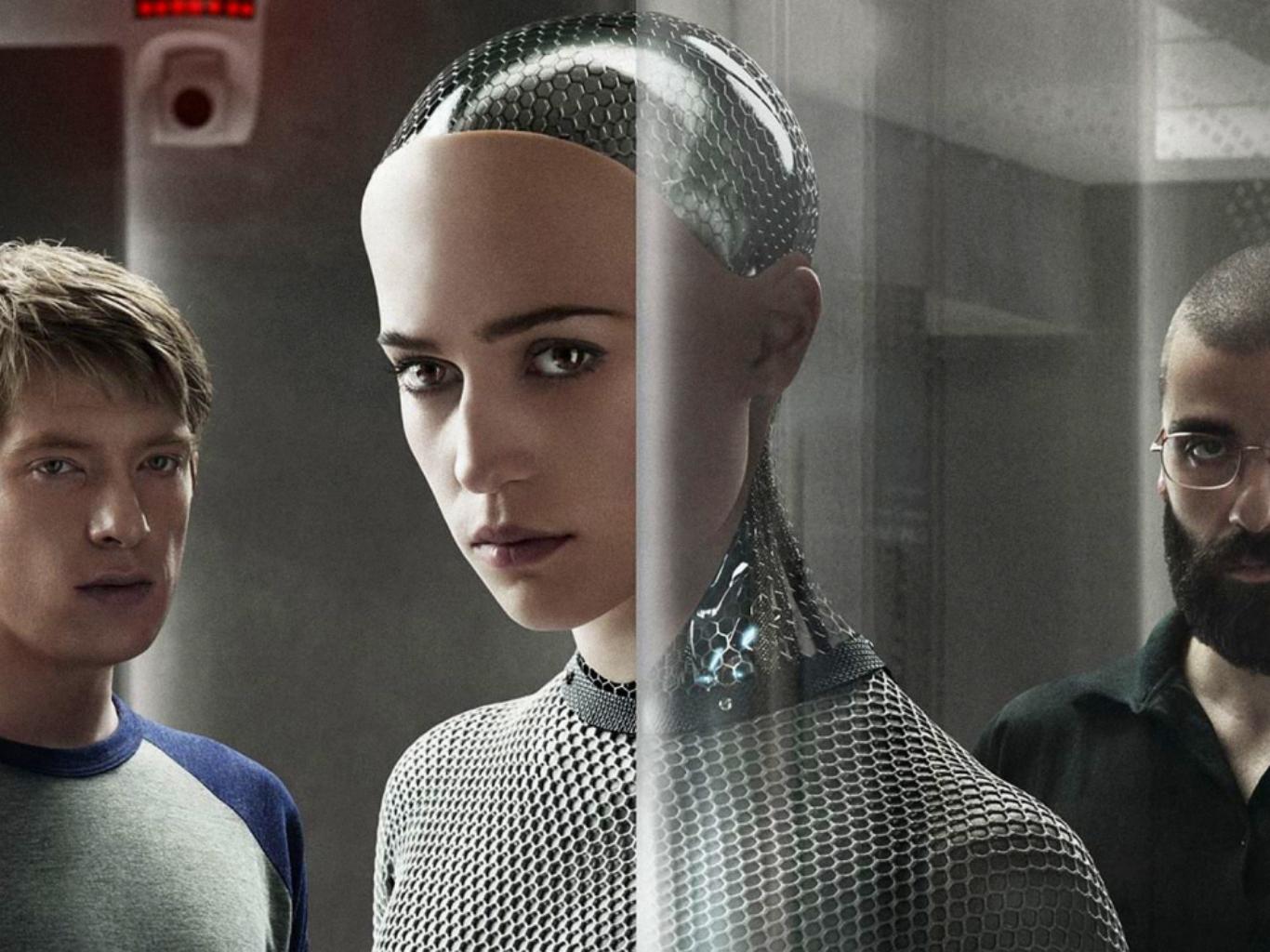








HUBMAX PH-D 12114XFE



Approaches and applications

- Expert systems
 - Medical diagnosis, legal advice, fault finding
- Neural networks
 - OCR, classification, clustering
- State space search
 - Game tree search, planning, path finding
- Naive Bayes classifier
 - Spam filtering, text classification, sentiment analysis
- Fuzzy logic
 - Sendai Subway system, HVAC systems, washing machines
- Genetic algorithms
 - Artificial creativity, computer-automated design, rostering

Backgrounds of AI researchers

Researchers in artificial intelligence come from a wide range of fields including but not limited to:

- Physics
- Mathematics
- Logic
- Statistics
- Psychology
- Philosophy
- Biology
- Neuroscience
- Linguistics
- Computer Science

Chatbots and dialogue systems

- Chatbots
- Dialogue Systems

IKEA NIEUWS IKEA MAAKT PLAATS VOOR NIEUW KEUKENSYSTEEM METOD

De kracht van licht is veiligere straten

Voor elke verkochte LEDARE led-lamp
schenkt de IKEA Foundation 1.- voor
licht in vluchtelingenkampen.



Deze actie geldt van 3 februari t/m 30 maart 2014



Ontdek
onze
laagste
prijzen

VASEN

Vaas

1.29



LAMPAN

Tafellamp

2.49



ANTILOP

Kinderstoel met
veiligheidsriempje

11.95



Vraag Anna



Anna zei: Welkom bij IKEA. Ik ben Anna,
de online assistente van IKEA. Je kan mij
vragen stellen over IKEA en onze producten
en diensten. Waarmee kan ik je helpen?

Typ hier om met Anna te praten



ELIZA

User Men are all alike.

ELIZA IN WHAT WAY?

User They're always bugging us about something or other.

ELIZA CAN YOU THINK OF A SPECIFIC EXAMPLE?

User Well, my boyfriend made me come here.

ELIZA YOUR BOYFRIEND MADE YOU COME HERE

User He says I'm depressed much of the time.

ELIZA I AM SORRY TO HEAR YOU ARE DEPRESSED

User It's true. I am unhappy

ELIZA DO YOU THINK COMING HERE WILL HELP
YOU NOT TO BE UNHAPPY

User I need some help, that much seems certain.

ELIZA WHAT WOULD IT MEAN TO YOU IF YOU GOT
SOME HELP

User Perhaps I could learn to get along with my
mother.

ALICE

```
<category>
  <pattern>WHO IS ALAN TURING</pattern>
  <template>Alan Mathison Turing, was a British
  philosopher, mathematician, logician,
  cryptanalyst and computer scientist. He was
  highly influential in the development of
  computer science, giving a formalisation of
  the concepts of "algorithm" and "computation"
  with the Turing machine, which can be
  considered a model of a general purpose
  computer. Turing is widely considered to be
  the father of theoretical computer science
  and artificial intelligence.
</template>
</category>
```

ALICE

```
<category>
  <pattern>CAN YOU *</pattern>
  <template>I don't know if I can <star/>, but
    I'm really good at swimming.</template>
</category>
```

ALICE

```
<category>
  <pattern>DO YOU KNOW WHO * IS</pattern>
  <template>
    <srai>WHO IS <star/></srai>
  </template>
</category>
```

ALICE

```
<category>
  <pattern>YES</pattern>
  <that>DO YOU LIKE MOVIES</that>
  <template>
    What is your favorite movie?
  </template>
</category>
```

ALICE

```
<category>
  <pattern>* ALWAYS *</pattern>
  <template>
    Can you think of a specific example?
  </template>
</category>
```

ALICE

```
<category>
  <pattern>I AM *</pattern>
  <template>
    How long have you been <star/>?
  </template>
</category>
```

Two approaches

- Chatbots
- Dialogue Systems



DICO



+



-

0

DICO

User Call Lisa please.

DICO OK, Lisa. Do you want to use the home number or the mobile?

User Her home number please.

Question accommodation

User Call Lisa's home number.

Question accommodation

User Lisa.

DICO I don't quite understand. Do you want to add an entry to the phonebook, call a person, change an entry in the phonebook, delete an entry from the phonebook or search for a name?

Task switching

User Change Lisa's home number.

DICO Okay. Let's see. What phone number do you want instead?

User Check my messages.

DICO You have got this message. Hi! I have got a new home number, it is (031) 234567. Best regards, Lisa.

Returning to change an entry. What phone number do you want instead?

User oh three one two three four five six seven.

DICO Okay. Changing Lisa's home number to oh three one two three four five six seven

Information revision

User I would like some price information.

GoDiS How would you like to travel?

User I'd like to go by train.

GoDiS Where do you want to go to?

User No, not by train, I want to go by plane.

GoDiS Okay. Where do you want to go to?

Grounding

GoDiS What city do you want to go to?

User ksd sdv we v sdv London

GoDiS Do you want to go to London?

User yes

GoDiS Okay. What city do you want to go from?

User I want to go from Paris.

GoDiS Okay. In what month do you want to leave?

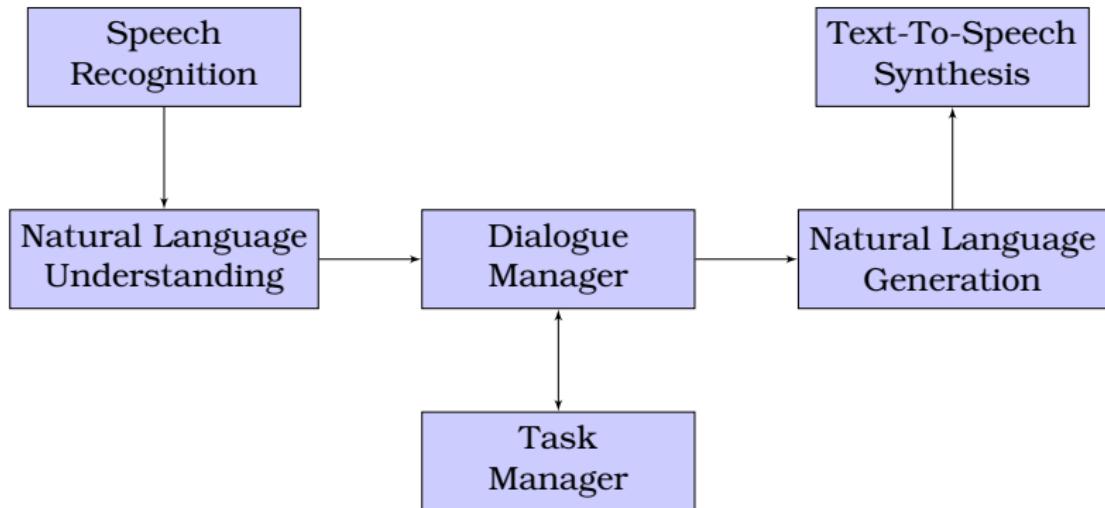
User sdv I want to leave in October.

GoDiS Okay. You want to leave in October.

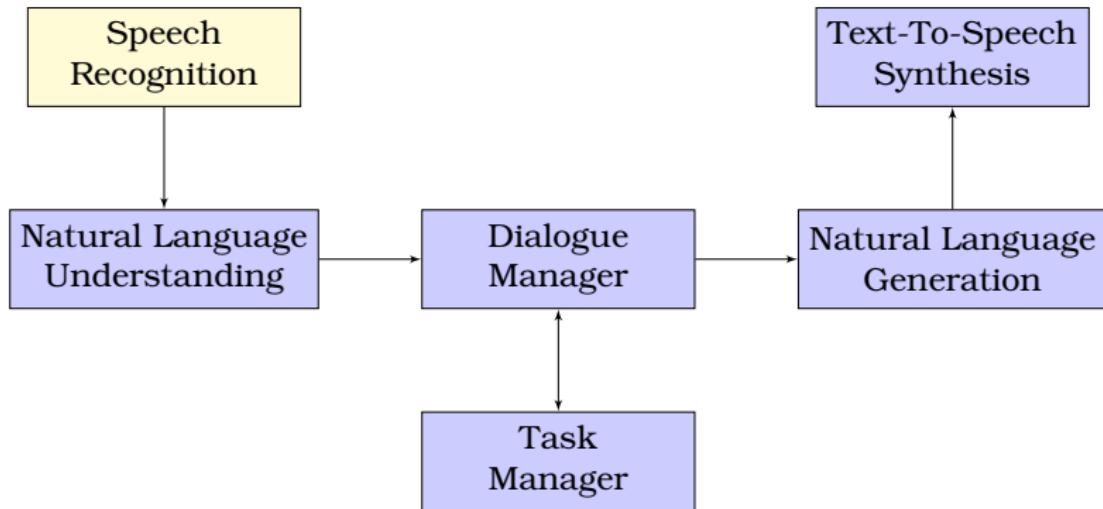
Example dialogue system applications

- Automated reservation systems
 - CU Communicator System
 - TOOT
 - Mercury Flight reservation system
- Natural language e-mail interfaces
 - ELVIS (E-maiL Voice Interactive System)
 - MailSec
- Planning & Problem solving systems
 - TRIPS & TRAINS
 - Circuit-Fix-It shop system
- Movie information system (MovieLine)
- Automated banking systems (Nuance)
- Weather information system (Jupiter)

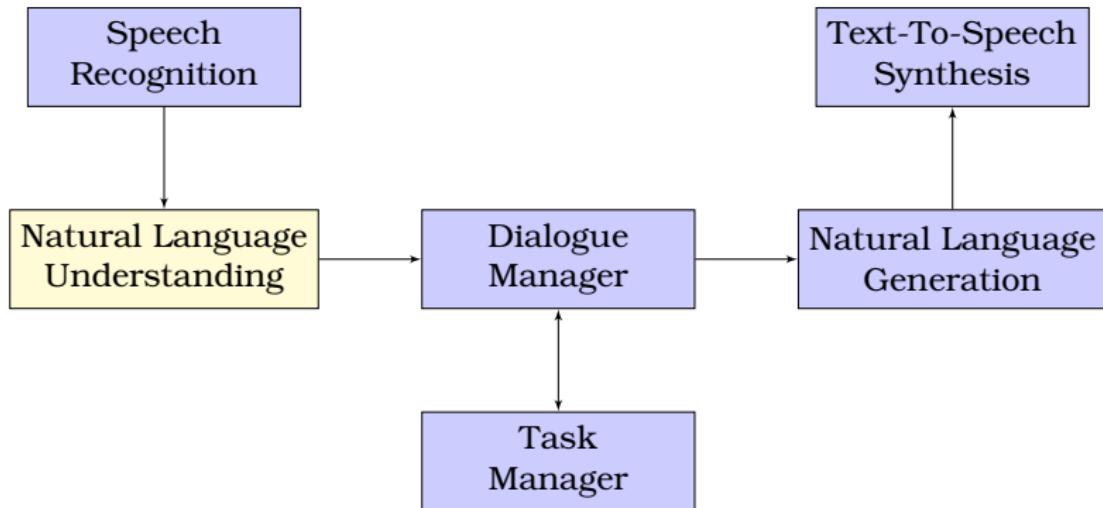
A spoken dialogue system architecture



A spoken dialogue system architecture

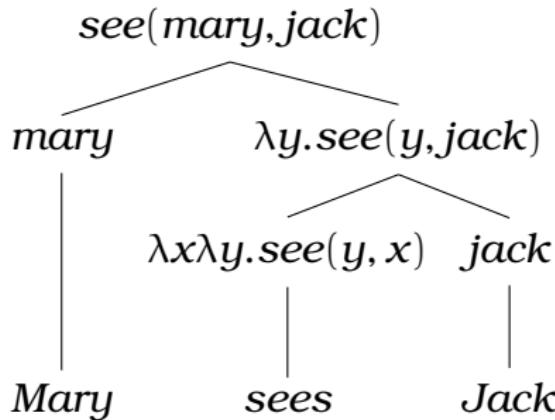


A spoken dialogue system architecture



Natural Language Understanding

"Mary sees Jack" \rightarrow $see(mary, jack)$



Natural Language Understanding

Show me morning flights from Boston to San Francisco on Tuesday.

SHOW:

FLIGHTS:

ORIGIN:

CITY: Boston

DATE:

DATE-OF-WEEK: Tuesday

TIME:

PART-OF-DAY: morning

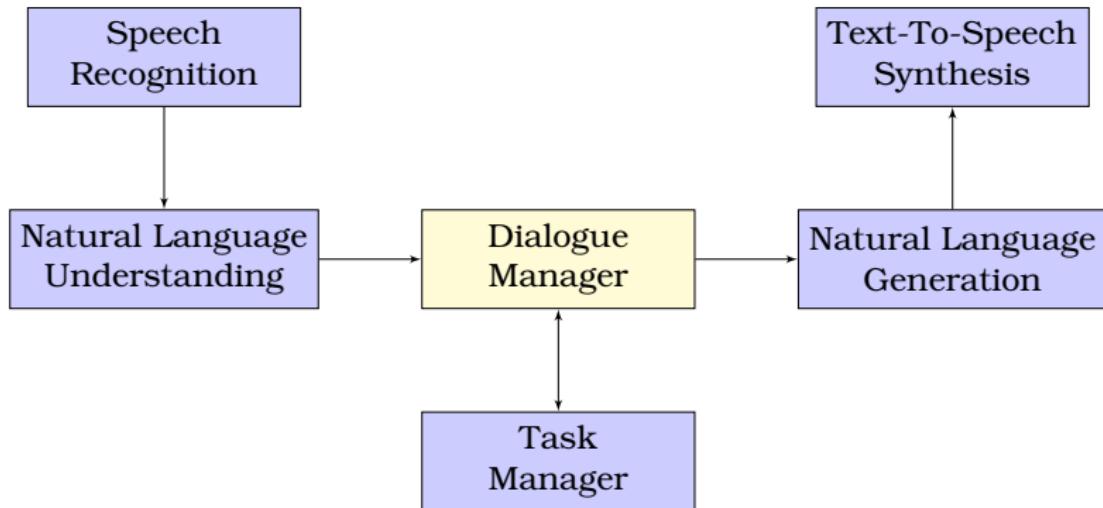
DEST:

CITY: San Francisco

Natural Language Understanding

```
input_form( [visa], ask(need_visa) ) .  
input_form( [price], ask(X^price(X)) ) .  
  
input_form( [by|S], answer(how(C)) ) :-  
    lexsem( S, C), sem_sort(C, means_of_transport)  
input_form( [in|S], answer(month(C)) ) :-  
    lexsem( S, C), sem_sort( C, month ).
```

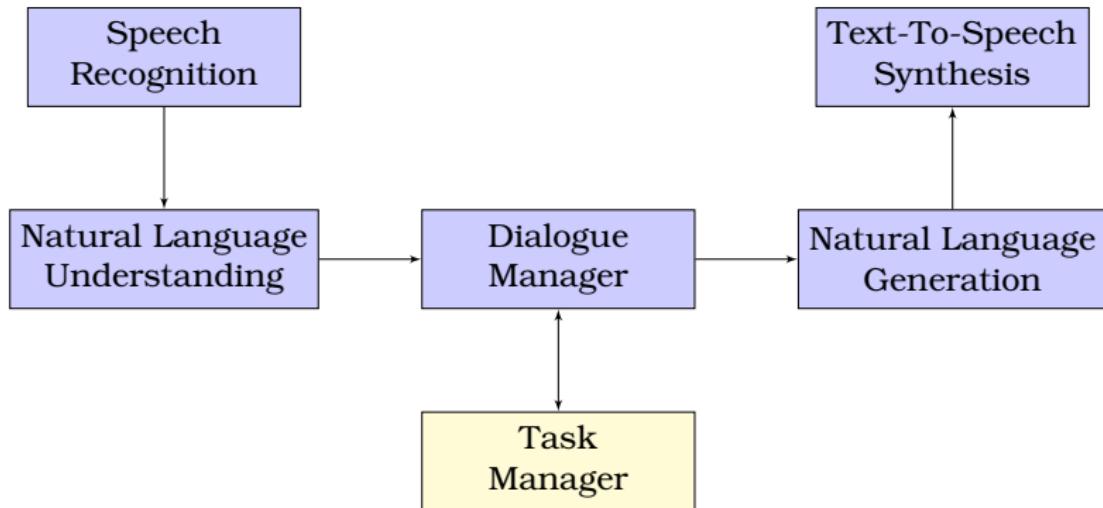
A spoken dialogue system architecture



Dialogue Manager

PRIVATE :	AGENDA	: <i>OpenQueue(Action)</i>
	PLAN	: <i>OpenStack(PlanConstruct)</i>
	BEL	: <i>Set(Prop)</i>
	TMP	: $\begin{bmatrix} \text{USR} & : & \text{Tmp} \\ \text{SYS} & : & \text{Tmp} \end{bmatrix}$
	NIM	: <i>OpenQueue(Pair(DP, Move))</i>
	COM	: <i>Set(Prop)</i>
SHARED :	ISSUES	: <i>OpenStack(Question)</i>
	ACTIONS	: <i>OpenStack(Action)</i>
	QUD	: <i>OpenStack(Question)</i>
	PM	: <i>OpenQueue(Move)</i>
	LU	: $\begin{bmatrix} \text{SPEAKER} & : & \text{Participant} \\ \text{MOVES} & : & \text{Set}(Move) \end{bmatrix}$

A spoken dialogue system architecture

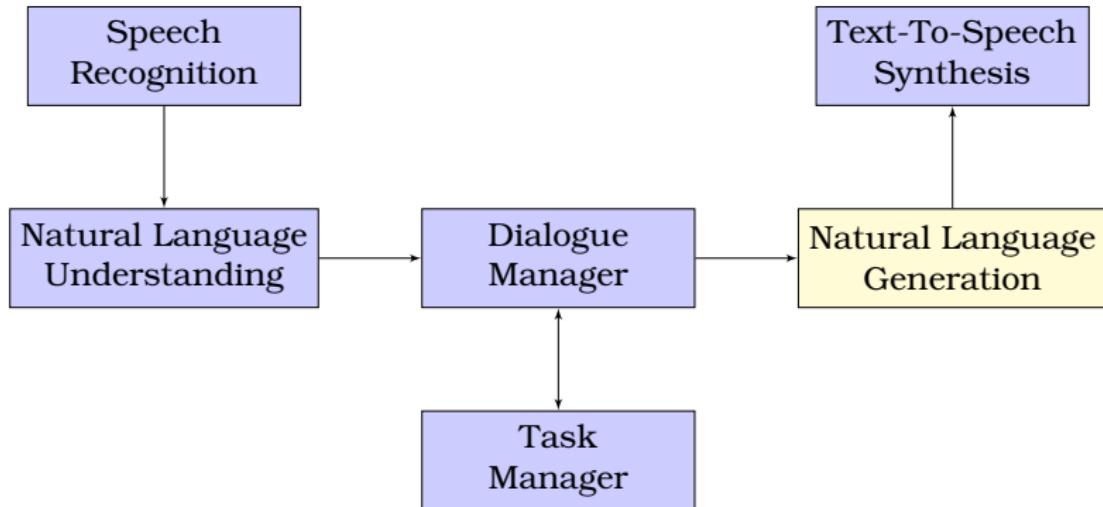


Task Manager

```
plan( X^price(X),  
      [ findout(X1^(how(X1))),  
        findout(X2^(dest_city(X2))),  
        findout(X3^(dept_city(X3))),  
        findout(X4^(month(X4))),  
        findout(X9^(dept_day(X9))),  
        findout(X5^(class(X5))),  
        consultDB(X7^(price(X7)))  
      ] ).
```

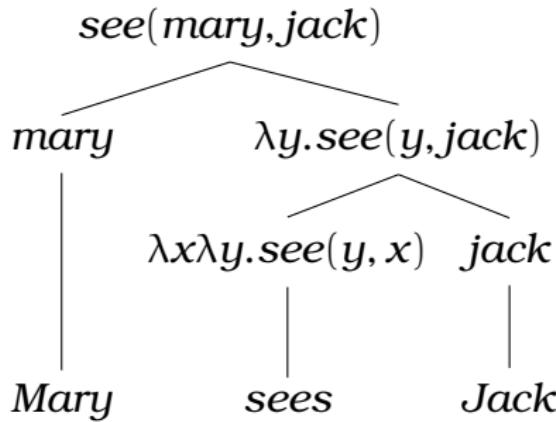
```
plan( need_visa,  
      [ findout(X^dest_city(X)),  
        findout(X1^citizenship(X1)),  
        consultDB( need_visa )  
      ] ).
```

A spoken dialogue system architecture



Natural Language Generation

$see(mary, jack) \rightarrow "Mary \ sees \ Jack"$



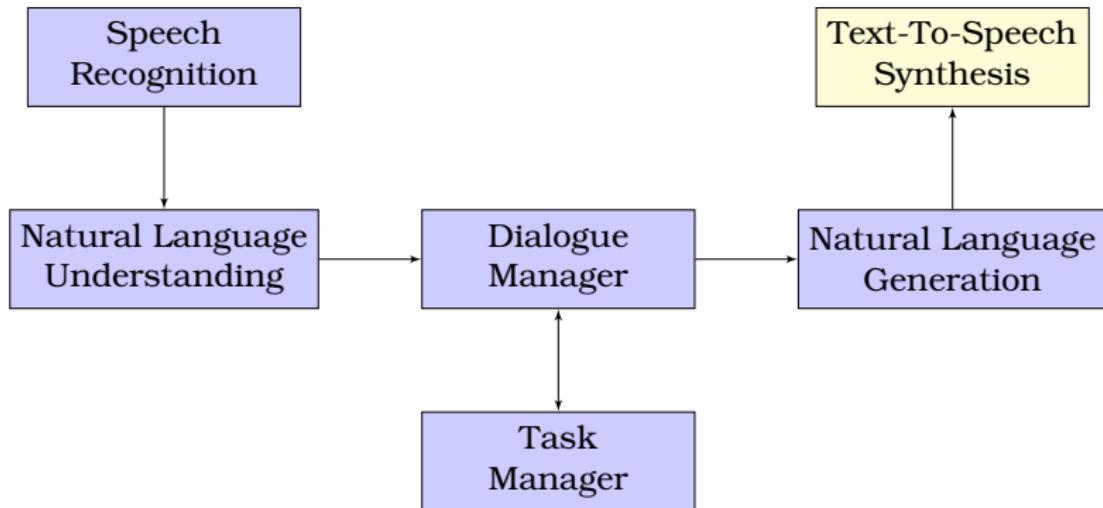
Natural Language Generation

Flights from $< src >$ to $< dest >$ on $< date >$. One moment please.

Natural Language Generation

```
output_form( answer(price(Price)) ,  
            ['The price is ', Price, ' crowns.']).  
  
output_form( answer(need_visa) ,  
            ['Yes, you need a Visa.']).  
output_form( answer(not(need_visa)) ,  
            ['No, you dont need a Visa.']).
```

A spoken dialogue system architecture



Two approaches

- Chatbots

- Behaviouristic
- Open domain
- Easy to build/maintain
- Turn-taking
- Non goal-oriented
- Doesn't handle different dialogue phenomena

- Dialogue Systems

- Cognitivistic
- Limited domain
- Requires specialist
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question
 - accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question
 - accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question
 - accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question
 - accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question
 - accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question
 - accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Two approaches

- Chatbots
 - Behaviouristic
 - Open domain
 - Easy to build/maintain
 - Turn-taking
 - Non goal-oriented
 - Doesn't handle different dialogue phenomena
- Dialogue Systems
 - Cognitivistic
 - Limited domain
 - Requires specialist
 - Mixed-initiative dialogue
 - Goal-oriented
 - Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

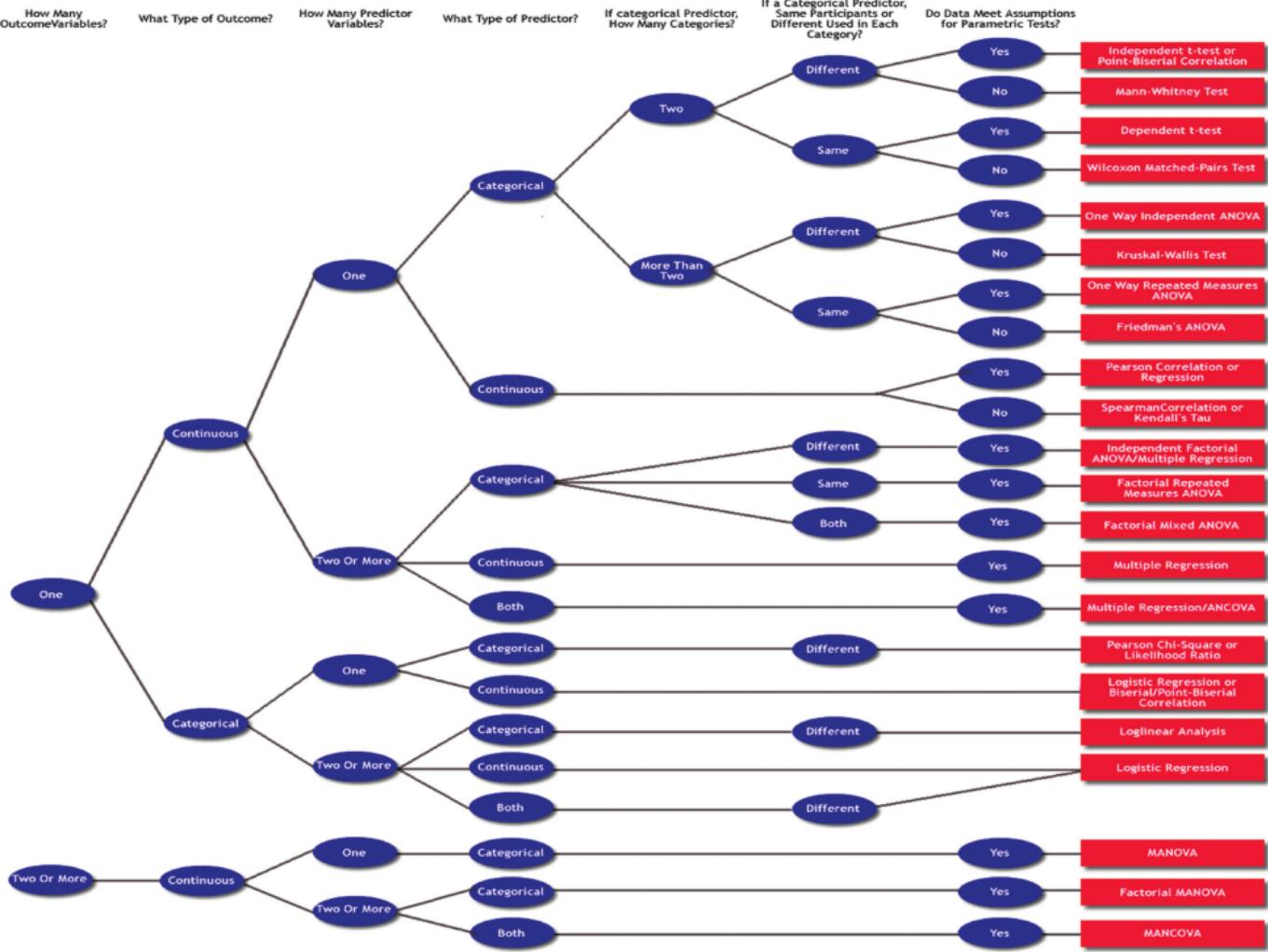
Proposed Solution

Hybrid Approach

- Open domain
- Easy to build/maintain
- Mixed-initiative dialogue
- Goal-oriented
- Handles dialogue phenomena, such as:
 - Grounding
 - Question accommodation
 - Task switching
 - Information revision

Methodology

- Implement a statistics tutor
- Have students interact with the system, and have them:
 - get help on a statistics question
 - ask two off-topic questions
- Improve system based on user interactions



Experiments

The research question is:

What is the relationship between gender and dropout rates from an MSc course?

What you have:

- the variable **sex**, with possible values male/female
- the variable **dropout** with possible values Yes/No

Demo

Demo

<https://www.youtube.com/watch?v=a7MMgnqak30>

Getting started

The prolog implementation that we will use is called SWI-Prolog.

It is free and there are versions for Linux, Windows, and MacOS X, which you can download from the SWI-Prolog web site.

www.swi-prolog.org

It can also be found in the Debian/Ubuntu repositories.

A knowledge base of facts

```
1  % kb1.pl
2
3  wizard(harry).
4  wizard(ron).
5  wizard(hermione).
6
7  muggle(uncle_vernon).
8  muggle(aunt_petunia).
9
10 chases(crookshanks,scabbers).
```

Queries we can ask

```
?- wizard(harry).  
true.
```

```
?- chases(crookshanks, scabbers).  
true.
```

```
?- muggle(harry).  
false.
```

```
?- muggle(dumbledore).  
false.
```

```
?- wizard(dumbledore).  
false.
```

```
?- witch(hermione).  
ERROR: toplevel: Undefined procedure: witch/1
```

```
% kb1.pl
```

```
wizard(harry).  
wizard(ron).  
wizard(hermione).
```

```
muggle(uncle_vernon).  
muggle(aunt_petunia).
```

```
chases(crookshanks, scabbers).
```

More queries we can ask

```
?- muggle(X).  
X = uncle_vernon ;  
X = aunt_petunia.  
  
?- chases(X,Y).  
X = crookshanks ;  
Y = scabbers.  
  
?- chases(X,X).  
false.
```

```
% kb1.pl  
  
wizard(harry).  
wizard(ron).  
wizard(hermione).  
  
muggle(uncle_vernon).  
muggle(aunt_petunia).  
  
chases(crookshanks,scabbers).
```

A bit of syntax: atoms and variables

- Atoms
 - All terms that consist of letters, numbers, and the underscore and start with a non-capital letter are atoms: `harry`, `uncle_vernon`, `ritaSkeeter`, `nimbus2000`, ...
 - All terms that are enclosed in single quotes are atoms: `'Professor Dumbledore'`, ...
 - Certain special symbols are also atoms: `+`, `,`, ...
- Variables
 - All terms that consist of letters, numbers, and the underscore and start with a capital letter or an underscore are variables: `X`, `Hermione`, `_ron`, ...
 - `_` is an anonymous variable: two occurrences of `_` are different variables.

A bit of syntax: compound terms

- Compound terms
 - Compound terms are of the form:
functor(argument, ..., argument).
 - Functors have to be atoms.
 - Arguments can be any kind of Prolog term, e.g., compound terms.

```
likes(ron,hermione), likes(harry,X),  
f(a,b,g(h(a)),c), ...
```

A bit of syntax: facts and queries

- Facts are compound terms which are followed by a full stop.

```
wizard(hermione) .
```

```
muggle(uncle_vernon) .
```

```
chases(crookshanks,scabbers) .
```

- Queries are also compound terms which are followed by a full stop.

```
?- wizard(hermione) .
```

A bit of syntax: facts and queries

- Facts are compound terms which are followed by a full stop.

```
wizard(hermione) .
```

```
muggle(uncle_vernon) .
```

```
chases(crookshanks,scabbers) .
```

- Queries are also compound terms which are followed by a full stop.

```
?- wizard(hermione) .
```



Prompt provided by the Prolog interpreter.

A bit of syntax: facts and queries

- Facts are compound terms which are followed by a full stop.

```
wizard(hermione) .
```

```
muggle(uncle_vernon) .
```

```
chases(crookshanks,scabbers) .
```

- Queries are also compound terms which are followed by a full stop.

```
?- wizard(hermione) .
```

Query.

Prompt provided by the Prolog interpreter.

A knowledge base of facts and rules

% kb2.pl

eating(dudley).

unhappy(hedwig).

kicking(dudley, hedwig).

happy(aunt_petunia) :- happy(dudley).

happy(uncle_vernon) :- happy(dudley), unhappy(harry).

happy(dudley) :- kicking(dudley, harry).

happy(dudley) :- eating(dudley).

A knowledge base of facts and rules

% kb2.pl

eating(dudley).

unhappy(hedwig).

kicking(dudley, hedwig).

happy(aunt_petunia) :- happy(dudley).

happy(uncle_vernon) :- happy(dudley), unhappy(harry).

happy(dudley) :- kicking(dudley, harry).

happy(dudley) :- eating(dudley).

if ... then: If **happy(dudley)** is true, then
happy(aunt_petunia) is true.

A knowledge base of facts and rules

```
% kb2.pl
```

```
eating(dudley).
```

```
unhappy(hedwig).
```

```
kicking(dudley, hedwig).
```

```
happy(aunt_petunia) :- happy(dudley).
```

```
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
```

```
happy(dudley) :- kicking(dudley, harry).
```

```
happy(dudley) :- eating(dudley).
```

A knowledge base of facts and rules

```
% kb2.pl
```

```
eating(dudley).
```

```
unhappy(hedwig).
```

```
kicking(dudley, hedwig).
```

```
happy(aunt_petunia) :- happy(dudley).
```

```
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
```

~~```
happy(dudley) :- kicking(dudley, harry).
```~~~~```
happy(dudley) :- eating(dudley).
```~~

and: If `happy(dudley)` is true and `unhappy(harry)` is true,
then `happy(uncle_vernon)` is true.

A knowledge base of facts and rules

```
% kb2.pl
```

```
eating(dudley).
```

```
unhappy(hedwig).
```

```
kicking(dudley,hedwig).
```

```
happy(aunt_petunia) :- happy(dudley).
```

```
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
```

```
happy(dudley) :- kicking(dudley,harry).
```

```
happy(dudley) :- eating(dudley).
```

A knowledge base of facts and rules

```
% kb2.pl
```

```
eating(dudley).
```

```
unhappy(hedwig).
```

```
kicking(dudley, hedwig).
```

```
happy(aunt_petunia) :- happy(dudley).
```

```
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
```

```
happy(dudley) :- kicking(dudley, harry).
```

```
happy(dudley) :- eating(dudley).
```

or: If `kicking(dudley, harry)` is true or if `eating(dudley)` is true, then `happy(dudley)` is true.

Queries we can ask

```
?- happy(dudley).  
true.  
  
?- happy(aunt_petunia).  
true.  
  
?- happy(uncle_vernon).  
false.  
  
?- happy(X).  
X = aunt_petunia ;  
X = dudley.
```

```
% kb2.pl  
  
eating(dudley).  
  
unhappy(hedwig).  
  
kicking(dudley, hedwig).  
  
happy(aunt_petunia) :- happy(dudley).  
happy(uncle_vernon) :- happy(dudley),  
                      unhappy(harry).  
happy(dudley) :- kicking(dudley, harry).  
happy(dudley) :- eating(dudley).
```

A bit of syntax: rules

- Rules are of the form $Head :- Body.$
- Like facts and queries, they have to be followed by a full stop.
- Head is a compound term.
- Body is compound term or a sequence of compound terms separated by commas.

```
happy(aunt_petunia) :- happy(dudley).
```

```
happy(uncle_vernon) :- happy(dudley), unhappy(harry).
```

Facts and rules containing variables

% kb3.pl

```
father(albert,james).  
father(james,harry).
```

```
mother(ruth,james).  
mother(lili,harry).
```

```
wizard(lili).  
wizard(ruth).  
wizard(albert).
```

```
wizard(X) :- father(Y,X),  
           wizard(Y),  
           mother(Z,X),  
           wizard(Z).
```

For all X, Y, Z, if
father(Y, X) is true and
wizard(Y) is true and
mother(Z, X) is true and
wizard(Z) is true, then
wizard(X) is true. I.e.,
for all X, if X's father and
mother are wizards, then
X is a wizard.

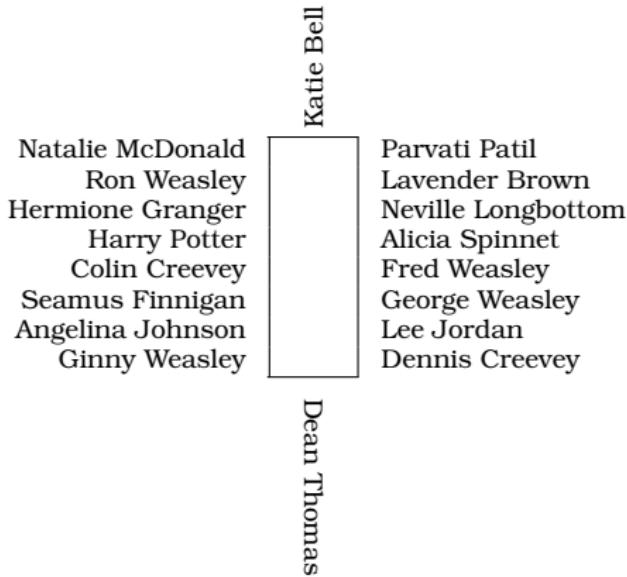
Queries we can ask

```
?- wizard(james).  
true.  
  
?- wizard(harry).  
true.  
  
?- wizard(X).  
X = lili ;  
X = ruth ;  
X = albert ;  
X = james ;  
X = harry.  
  
?- wizard(X), mother(Y,X), wizard(Y).  
X = james  
Y = ruth ;  
X = harry  
Y = lili.
```

```
% kb3.pl  
  
father(albert,james).  
father(james,harry).  
  
mother(ruth,james).  
mother(lili,harry).  
  
wizard(lili).  
wizard(ruth).  
wizard(albert).  
  
wizard(X) :- father(Y,X),  
            wizard(Y),  
            mother(Z,X),  
            wizard(Z).
```

Exercise: The Gryffindor table

The following picture shows who is sitting at the Gryffindor table.



Define the predicate `sits_right_of/2` to represent who is sitting right of whom. `sits_right_of(X, Y)` should be true if X is to the right of Y.

Exercise: The Gryffindor table

Based on this knowledge base, formulate the rules defining the following predicates:

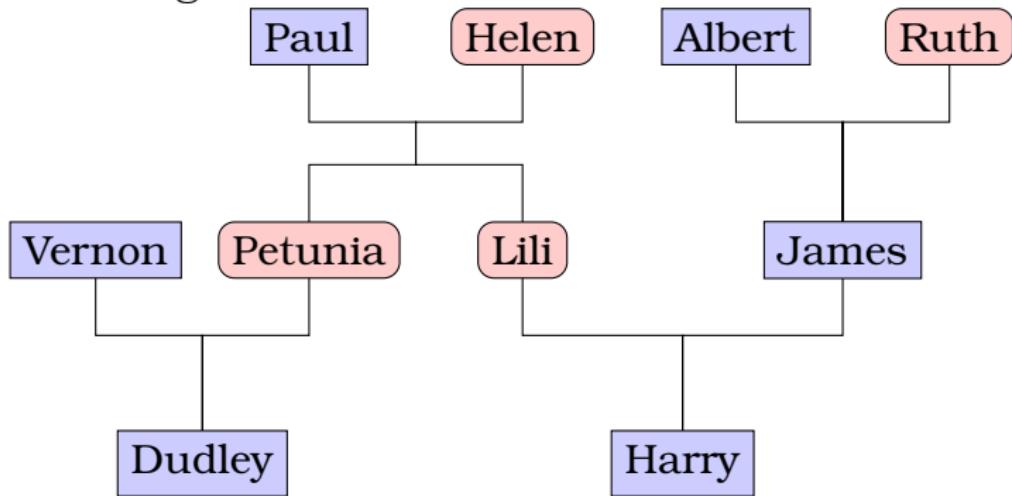
- `sits_left_of/2`: `sits_left_of(X, Y)` should be true if X is to the left of Y.
- `are_neighbours_of/3`: `are_neighbours_of(X, Y, Z)` should be true if X is to the left of Z and Y is to the right of Z.
- `next_to_each_other/2`: `next_to_each_other(X, Y)` should be true if X is next to Y.

Test your implementation by asking queries. For example:

- Is Lavender to the right of Parvati?
- Is Lavender to the right of Neville?
- Who is to the right of Hermione?
- Who is sitting at the table?
- Who is sitting two seats to the right of Hermione?
- Who is sitting between Neville and Fred?

Exercise: Family relationships

Use the predicates `male/1`, `female/1`, and `parent_of/2` to represent the following family tree as a Prolog knowledge base.



Exercise: Family relationships

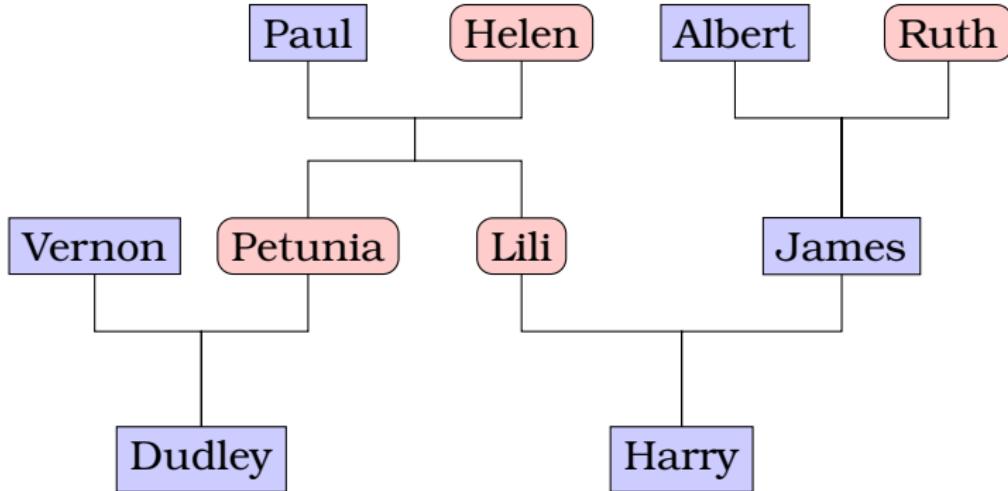
Now, formulate rules to capture the following relationships:

- `father_of(Father,Child)` **and** `mother_of(Mother,Child)`
- `grandfather_of(Grandfather,Child)` **and**
`grandmother_of(Grandmother,Child)`
- `sister_of(Sister,Person)` **and** `brother_of(Brother,Person)`
- `aunt_of(Aunt,Person)` **and** `uncle_of(Uncle,Person)`

To test your knowledge base ask all kinds of queries.
For example,

- Does Harry have an aunt? Who?
- Who are the grandparents of Harry?
- Who are the grandchildren of Paul and Helen?
- Does James have a sister?

Ancestors



Task

Define a predicate `ancestor_of(X, Y)` which is true if X is an ancestor of Y.

Ancestors

```
grandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,Y).  
  
greatgrandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,A),  
                           parent_of(A,Y).  
  
greatgreatgrandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,A),  
                                parent_of(A,B), parent_of(B,Y).
```

Ancestors

```
grandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,Y).  
  
greatgrandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,A),  
                           parent_of(A,Y).  
  
greatgreatgrandparent_of(X,Y) :- parent_of(X,Z), parent_of(Z,A),  
                                parent_of(A,B), parent_of(B,Y).  
  
ancestor_of(X,Y) :- parent_of(X,Y).  
ancestor_of(X,Y) :- parent_of(X,Z), ancestor_of(Z,Y).
```

Exercise: Finding paths through a maze

Imagine that the following knowledge base describes a maze.

The facts determine which points are connected, i.e., from which point you can get to which other point in one step.

Furthermore, imagine that all paths are one-way streets, so that you can only walk them in one direction. So, you can get from point 1 to point 2, but not the other way round.

% *maze.pl*

```
connected(1, 2).  
connected(3, 4).  
connected(5, 6).  
connected(7, 8).  
connected(9, 10).  
connected(12, 13).  
connected(13, 14).  
connected(15, 16).  
connected(17, 18).  
connected(19, 20).  
connected(4, 1).  
connected(6, 3).  
connected(4, 7).  
connected(6, 11).  
connected(14, 9).  
connected(11, 15).  
connected(16, 12).  
connected(14, 17).  
connected(16, 19).
```

Exercise: Finding paths through a maze

Write a predicate `path/2` that tells you from which point in the maze you can get to which other point when chaining together connections.

Now ask some queries.

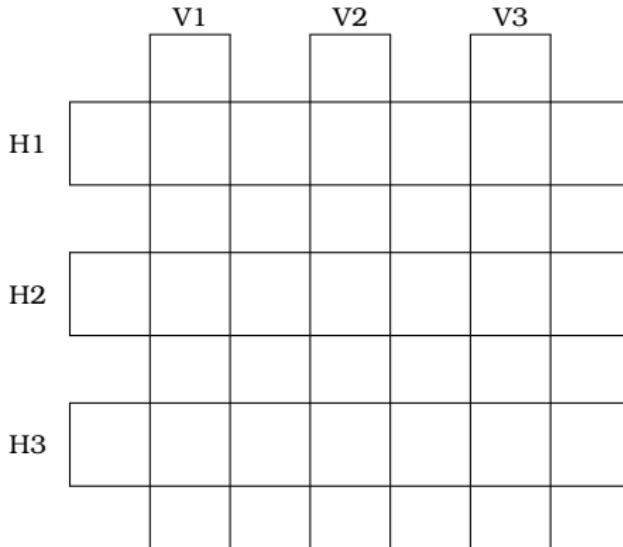
- Can you get from point 5 to point 10?
- Which other point can you get to when starting in point 1?
- And which points can be reached from point 13?

Exercise: Crossword puzzle

Here are six English words:

*abalone, abandon, anagram,
connect, elegant, enhance.*

They are to be arranged in a crossword puzzle like fashion in the grid given below.



Exercise: Crossword puzzle

The following knowledge base represents a lexicon containing these words.

```
word(abalone, a, b, a, l, o, n, e).  
word(abandon, a, b, a, n, d, o, n).  
word(enhance, e, n, h, a, n, c, e).  
word(anagram, a, n, a, g, r, a, m).  
word(connect, c, o, n, n, e, c, t).  
word(elegant, e, l, e, g, a, n, t).
```

Write a predicate `crosswd/6` that computes all the different ways of how to fill the grid. The first three arguments should be the vertical words from left to right and the following three arguments the horizontal words from top to bottom.

Exercise: Crossword puzzle

Hint

The second letter of word V1 has to be the same as the second letter of word H1, the fourth letter of word V1 has to be the same as the second letter of word H2, and the sixth letter of word V1 has to be the same as the second letter of word H3.

Similar constraints can be specified for the other words.

Arithmetic in Prolog

```
?- 3+5 = +(3, 5).  
true.
```

```
?- 3+5 = +(5, 3).  
false.
```

```
?- 3+5 = 8.  
false.
```

The built-in predicate `is`

```
?- X is 3+5.
```

```
X = 8.
```

```
?- X is 30-4.
```

```
X = 26.
```

```
?- X is 3*5.
```

```
X = 15.
```

```
?- X is 9/4.
```

```
X = 2.25.
```

Built-in predicates for comparing values of arithmetic expressions

```
?- 8 > 3.
```

```
true.
```

```
?- 8+2 > 9-2.
```

```
true.
```

```
?- 8 < 3.
```

```
false.
```

```
?- 8 =:= 3.
```

```
false.
```

```
?- 8 =\= 3.
```

```
true.
```

```
?- 8 >= 3.
```

```
true.
```

```
?- 8 =< 3.
```

```
false.
```

Lists

Lists start and end with square brackets, and each of the items they contain is separated by a comma.

Here are some examples:

```
[a,b,c,d,e,f,g]
```

```
[apple,pear,bananas,breadfruit]
```

```
[] % An empty list
```

```
[1,2,3,4]
```

```
[compound(terms),are(also,possible),in(lists)]
```

```
[a,freddie,A_Variable,apple]
```

```
[a,b,[c,d,e,[]],f,g]
```

Lists

Prolog also has a special facility to split the first part of the list (called the head) away from the rest of the list (known as the tail).

We can place a special symbol `|` (pronounced 'bar') in the list to distinguish between the first item in the list and the remainder of the list.

```
?- [first,second,third] = [A|B].  
A = first,  
B = [second, third].
```

Working with lists

`trans_a_b/2`: a predicate for "translating" a list of *as* into a list of *bs*.

`trans_a_b(X, Y)` should be true if *X*, the 'input', is a list of *as* and *Y*, the 'output', is a list of *bs* which has just as many *bs* as the input has *as*.

```
trans_a_b([], []).  
trans_a_b([a|InputTail], [b|OutputTail]) :-  
    trans_a_b(InputTail, OutputTail).
```

If the input is empty, then the output is empty as well. Otherwise the first *a* in the input list has to correspond to a *b* in the output list. The tail of the output list has to be the "translation" of the input list.

Working with lists

`element_of/2`: a predicate for testing whether a list contains a given Prolog term.

`element_of(X, Y)`: should be true if `X` is an element of `Y`.

```
element_of(X, [X|_]).  
element_of(X, [_|Tail]) :- element_of(X, Tail).
```

If the first element of the list is the one that we are looking for, we are done. Otherwise, check whether the term we are looking for is in the tail of the list.

In SWI-Prolog `element_of/2` is predefined under the name `member/2`.

Working with lists

`concatenate/3`: a predicate for concatenating two lists.

`concatenate(X, Y, Z)` should be true if `Z` is the concatenation of `X` and `Y`; for example, concatenating `[a]` with `[b, c]` yields `[a, b, c]`.

```
concatenate([], L, L).  
concatenate([Head|Tail], L, [Head|NewTail]) :-  
    concatenate(Tail, L, NewTail).
```

Concatenating the empty list with any other list `L` yields `L`. Otherwise, the first element of the output list has to be the same as the first element of the first input list. And the tail of the output list is the concatenation of the tail of the first input list with the second input list.

In SWI-Prolog `concatenate/3` is predefined under the name `append/3`.

Prolog predicates can be used in many ways

```
?- trans_a_b([a,a,a],[b,b,b]).  
true.
```

```
?- trans_a_b([a,a,a],L).  
L = [b,b,b].
```

```
?- trans_a_b([a,a,a],[b]).  
false.
```

```
?- trans_a_b(L,[b,b]).  
L=[a,a].
```

```
?- member(a,[a,b,c]).  
true.
```

```
?- member(X,[a,b,c]).  
X = a ;  
X = b ;  
X = c ;  
false.
```

```
?- member(a,L).  
L = [a| G280] ;  
L = [ G279, a| G283] ;  
L = [ G279, G282, a| G286] ;  
L = [ G279, G282, G285, a| G289]  
true.
```

Exercise: List processing

Define a predicate `all_as(List)` which succeeds if List is a list containing only a's as elements (if it contains any elements at all).

Hint

Ask yourself: what are the properties that an `all_as` list should have? One possibility is that it is empty. If it is not empty, then its first element should be an a and the tail should be an `all_as` list.

Exercise: List processing

Define a predicate `replace_a_b_c(InList,OutList)` where `OutList` is obtained from `InList` by replacing all a's in `InList` with b's, all b's with c's, and all c's with a's.

Exercise: List processing

Define a predicate `list_length(List, Length)` where `Length` is the number of elements in list `List`.

Hint

The length of empty lists is obviously 0. We don't need any calculations to find this out. The length of non-empty lists is one more than the length of their tails. That is, if `L` is the length of the tail, the length of the list is `L + 1`.

Exercise: Finding routes through a maze

Consider again the following knowledge base that describes a maze.

Define a predicate

route(From, To, Route) where From is the starting point, To the end point and Route is a list of points you pass while traversing to the end point.

Hint

Try to start with the path/2 predicate you made earlier and extend it.

% maze.pl

```
connected(1, 2).  
connected(3, 4).  
connected(5, 6).  
connected(7, 8).  
connected(9, 10).  
connected(12, 13).  
connected(13, 14).  
connected(15, 16).  
connected(17, 18).  
connected(19, 20).  
connected(4, 1).  
connected(6, 3).  
connected(4, 7).  
connected(6, 11).  
connected(14, 9).  
connected(11, 15).  
connected(16, 12).  
connected(14, 17).  
connected(16, 19).
```

Definite Clause Grammars

```
% english.pl
```

```
s --> np, vp.
```

```
np --> art, noun.
```

```
np --> pn.
```

```
vp --> verb, np.
```

```
verb --> [likes].
```

```
verb --> [kicks].
```

```
verb --> [shoots].
```

```
art --> [the].
```

```
art --> [some].
```

```
pn --> [mary].
```

```
pn --> [john].
```

```
noun --> [cat].
```

```
noun --> [dog].
```

Definite Clause Grammars

```
?- s([mary,shoots,john],[]).  
true.
```

```
?- s([some,cat,likes,mary],[]).  
true.
```

```
?- s(S,[]).  
S = [the, cat, likes, the, cat] ;  
S = [the, cat, likes, the, dog] ;  
S = [the, cat, likes, some, cat] ;  
S = [the, cat, likes, some, dog] ;  
S = [the, cat, likes, mary] ;  
S = [the, cat, likes, john] ;  
S = [the, cat, kicks, the, cat] ;  
S = [the, cat, kicks, the, dog] ;  
S = [the, cat, kicks, some, cat] ;  
S = [the, cat, kicks, some, dog] ;  
S = [the, cat, kicks, mary] ;  
S = [the, cat, kicks, john]
```

Exercise: Definite Clause Grammars

s --> np, vp.

np --> art, noun.

np --> pn.

vp --> verb, np.

verb --> [likes].

verb --> [kicks].

verb --> [shoots].

art --> [the].

art --> [some].

pn --> [mary].

pn --> [john].

noun --> [cat].

noun --> [dog].

Extend this grammar so that it also accepts

[john,cries].

Example: Number agreement

Extend this grammar so that it accepts

[the, cats, like, mary],

but not

[the, cats, likes, mary].

Solution: Number agreement

% english.pl

s --> np(N), vp(N) .

np(N) --> art, noun(N) .

np(N) --> det(N), noun(N) .

np(N) --> pn(N) .

vp(N) --> tv(N), np(_) .

vp(N) --> iv(N) .

tv(singular) --> [likes] .

tv(plural) --> [like] .

tv(singular) --> [kicks] .

tv(plural) --> [kick] .

tv(singular) --> [shoots] .

tv(plural) --> [shoot] .

iv(singular) --> [cries] .

iv(plural) --> [cry] .

art --> [the] .

art --> [some] .

det(singular) --> [every] .

det(plural) --> [all] .

pn(singular) --> [mary] .

pn(singular) --> [john] .

noun(singular) --> [cat] .

noun(plural) --> [cats] .

noun(singular) --> [dog] .

noun(plural) --> [dogs] .

Compositional semantics

```
sentence(S) --> noun_phrase(X, P, S), verb_phrase(X, P).  
  
noun_phrase(X, P, S) --> determiner(X, P12, P, S), noun(X, P1),  
                           rel_clause(X, P1, P12).  
noun_phrase(X, P, P) --> proper_noun(X).  
  
verb_phrase(X, P) --> trans_verb(X, Y, P1), noun_phrase(Y, P1, P).  
verb_phrase(X, P) --> intrans_verb(X, P).  
  
rel_clause(X, P1, (P1, P2)) --> [that], verb_phrase(X, P2).  
rel_clause(_, P1, P1) --> [].  
  
determiner(_, P1, P, (P :- P1)) --> [every].  
  
trans_verb(X, Y, likes(X, Y)) --> [likes].  
trans_verb(X, Y, admires(X, Y)) --> [admires].  
  
intrans_verb(X, paints(X)) --> [paints].  
  
proper_noun(john) --> [john].          noun(X, man(X)) --> [man].  
proper_noun(annie) --> [annie].         noun(X, woman(X)) --> [woman].  
proper_noun(monet) --> [monet].
```

Compositional semantics

```
?- sentence(Sem, [john,paints], []).  
Sem = paints(john) .
```

```
?- sentence(Sem, [john,admires,annie], []).  
Sem = admires(john, annie) .
```

```
?- sentence(paints(john), S, []).  
S = [john, paints] .
```

```
?- sentence(Sem, [every,man,that,paints,admires,monet], []).  
Sem = (admires(_G3552, monet) :- man(_G3552), paints(_G3552)) .
```

```
?- sentence(Sem, [annie,likes,every,man,that,admires,monet], []).  
Sem = (likes(annie, _G3556) :- man(_G3556), admires(_G3556, monet)) .
```

Final course project

The final course project consists of creating your own chatbot that can answer questions about a certain topic (e.g. movies, the weather, railways time table).

You will be asked to demo an example interaction with your chatbot and explain

- the topic your chatbot can converse about;
- which, if any, web APIs you're using;
- which, if any, Prolog facilities you're using.

Example files will be provided and explained in class.

Q&A

Good luck!