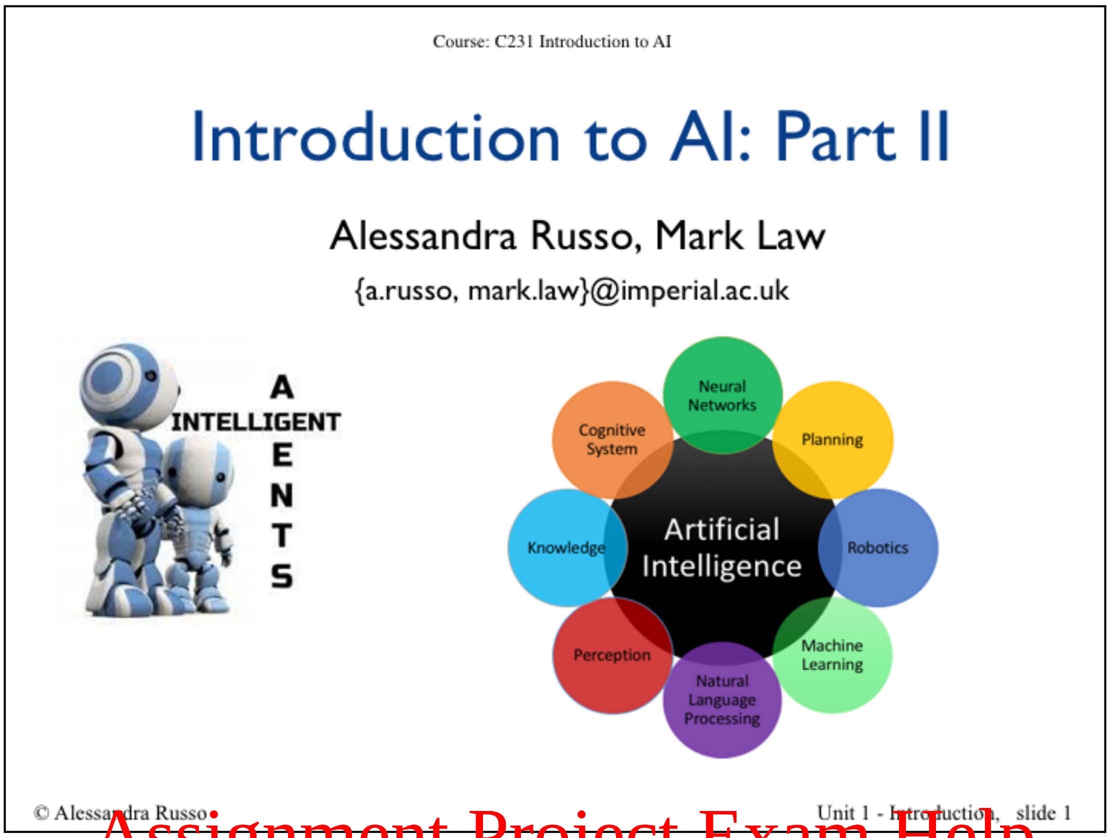


Course: C231 Introduction to AI

Introduction to AI: Part II

Alessandra Russo, Mark Law
{a.russo, mark.law}@imperial.ac.uk

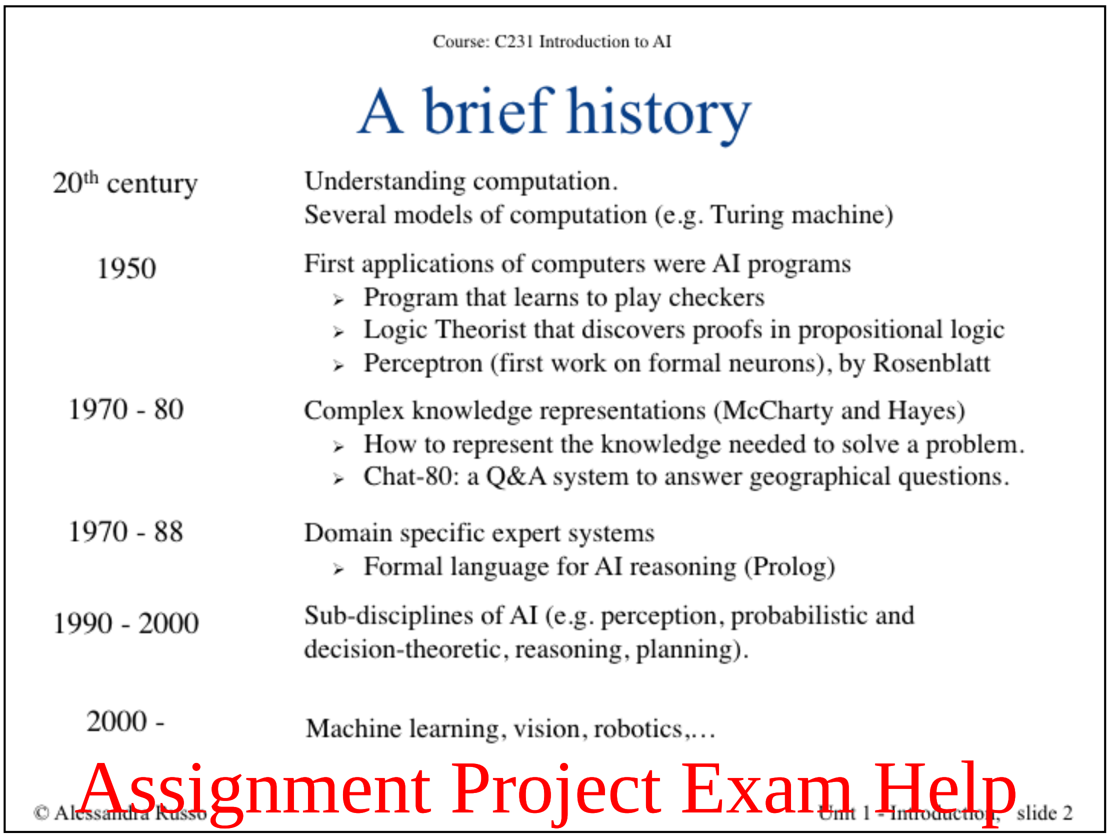
A
INTELLIGENT
E
N
T
S
Neural Networks
Cognitive System
Planning
Knowledge
Artificial Intelligence
Robotics
Perception
Machine Learning
Natural Language Processing

© Alessandra Russo Unit 1 - Introduction, slide 1

To introduce what this part of the course is going to cover, let's pose and look again at the meaning of the term "Artificial Intelligence". *Intelligence* can be defined in many ways and if you look around at various definition you will see that it surely includes the ability to understand, learn, acquire and use knowledge to solve given tasks, think abstractly and reason about problems in order to come up with best solutions and plans, to communicate and to perceive the external environment. Artificial intelligence is on the other hand the theory and development of computer software (or systems) able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages. In essence, the study and the development of **intelligence software agents**, machines that act intelligently, that manifest some of the capabilities that the human mind has (e.g., understand language, recognize pictures, solve problems, and learn). The wide field of Artificial Intelligence encompasses research areas and efforts in each of these different aspects of intelligence, and for each of these area the focus is on understanding the principles that underpin such capabilities as well as developing computational mechanisms that enable the realisation of efficient and effective computational systems that perform tasks commonly viewed as requiring intelligence.

In this part of the course we will concentrate on some of the fields of Artificial Intelligence listed above, specifically, on knowledge representation, reasoning, planning and problem solving. Fields such as learning, robotics, and neural networks will be covered in other courses in the third and four year of your degree. Some of the aspects that we cover in this course will provide you the foundations for courses taught in the third and four year that are related to AI. For instance, knowledge representation and reasoning and planning, will be useful in the logic-based learning and in the systems verification courses taught in the third year, as well as in the knowledge representation course taught in the forth year.

Part of the material included in this second part of the course is also based on the textbook "Artificial Intelligence, foundations of computational agents", by David Poole and Alan Mackworth.



Course: C231 Introduction to AI

A brief history

20th century	Understanding computation. Several models of computation (e.g. Turing machine)
1950	First applications of computers were AI programs ➢ Program that learns to play checkers ➢ Logic Theorist that discovers proofs in propositional logic ➢ Perceptron (first work on formal neurons), by Rosenblatt
1970 - 80	Complex knowledge representations (McCharty and Hayes) ➢ How to represent the knowledge needed to solve a problem. ➢ Chat-80: a Q&A system to answer geographical questions.
1970 - 88	Domain specific expert systems ➢ Formal language for AI reasoning (Prolog)
1990 - 2000	Sub-disciplines of AI (e.g. perception, probabilistic and decision-theoretic, reasoning, planning).
2000 -	Machine learning, vision, robotics,…

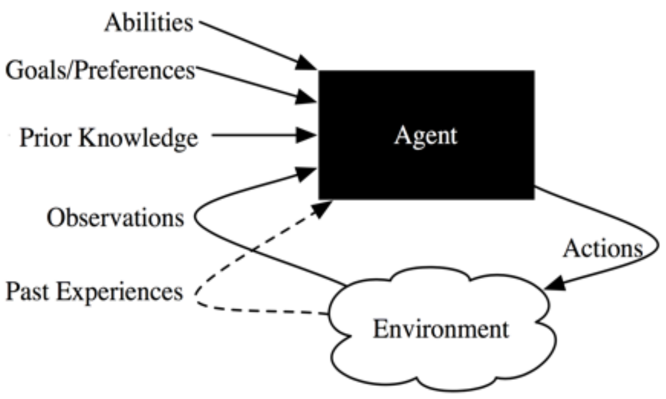
© Alessandra Russo Unit 1 - Introduction, slide 2

To give you a brief overview of the history of AI, the root of AI started long time ago (around 400 years ago) when philosopher such as Hobbes, started to study the notion of symbolic reasoning, as symbolic operations that capture the process of human thinking. In the early 1900, the advent of the Turing machine contributed further to the notion of symbolic operations, as concrete computational processes. Different models of computations were developed, among which the model of a Turing machine, an abstract machine that writes symbols on an infinitely long tape, and the lambda calculus of Church, a mathematical formalism for rewriting formulas. Once real computers were built (around 1950) some of the first applications of computers were AI programs, such as a checkers program (i.e. a program that learns to play checkers – or draughts) and the Logic Theorist, a program that discovers proofs in propositional logic. Together with high-level symbolic reasoning there was also some emphasis on low-level learning, which was initially inspired by the perceptron formal neuron model developed by Rosenblatt in 1958.

All these examples of programs focused mainly on learning and search problems. It became then apparent that to provide intelligent solutions to problems it was essential to tackle the main problem of how to represent the knowledge needed to solve a given problem. In the 80s, AI research activity saw the proliferation of many proposals from simple to more complex knowledge representation and languages based on logic (see McCarthy and Hayes), which then naturally led to first systems able to understand simple, controlled natural language interactions. Early examples were Chat-80, which was a Q&A system designed to answer geographical questions. Expert systems then followed, where the aim was to capture a large body of knowledge of an expert in some domain so that a computer could carry out expert tasks. At the same time further work was done in AI in the area of automated reasoning and languages, and systems such as Prolog became established in the AI community. From 1990 onwards, specific sub-disciplines of AI such as perception, probabilistic and decision-theoretic reasoning, planning, machine learning, and many other fields started to grow.

Reasoning and learning have been since then at the center of AI research. You will see more about learning in subsequent courses in your third and four year. In the rest of this course we concentrate on the reasoning and knowledge representation and problem solving aspects of AI.

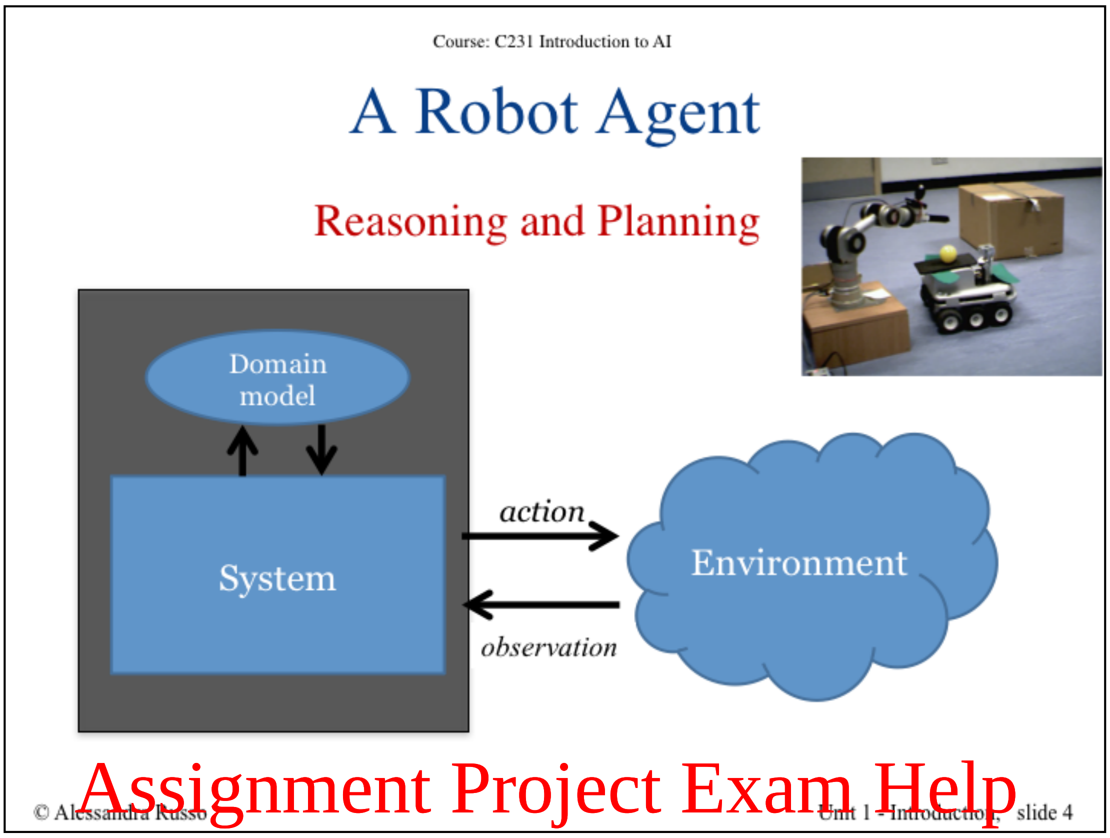
Intelligent Agent



- ❑ Robot agents
- ❑ Digital assistant
- ❑ Software agent
- ❑ …..

We have mentioned at the beginning that AI is, in essence, the study and the development of intelligence computational agents. What we aim to cover in this course is the study of principles that underlie intelligent computational agents. First of all, we should define what an intelligent computational agents is. Looking at the definition of David Pool in his text book on “Artificial Intelligent, foundations of computational agents”, an intelligent agent is an agent that acts in an environment. At any time, it decides what actions to perform, based on **observations** of the current environment, **prior knowledge** about the environment and task that it has to accomplish, **goals** that it must try to achieve, also with **preferences** over states of the world, and of course its own **abilities,** that is primitive actions that it is capable of carry out. In an intelligent agent, decisions may also be informed by **past experience**, that is knowledge about previous actions and observations of effects these actions in given states of the environment. Past experience can be used by the agent as new learned knowledge to improve the execution of its task. Prior knowledge is, in general, knowledge that cannot be learned through experience.

We can have different types of agents. An agent could, for example, incorporates a computational engine with physical sensors and actuators and act in an environment which is a physical setting. This is the case of a *robot agent*. It could be an advice-giving computer – *a digital assistant*– where the environment the agent has to interact with is the human. The human provides the perceptual information and carries out the task. Or, we could have a software agent. This is an agent that acts in a purely computational environment (e.g., smart meter controller).


Course: C231 Introduction to AI

A Robot Agent

Reasoning and Planning

© Alessandra Russo

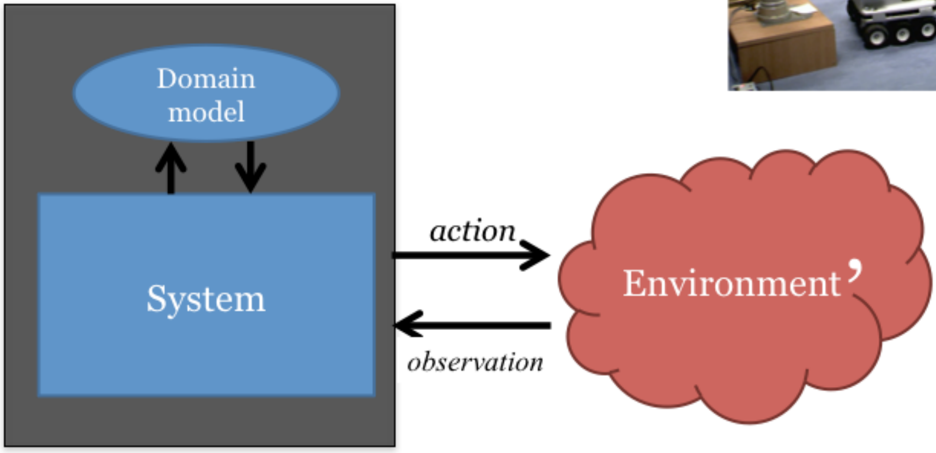
Unit 1 - Introduction, slide 4

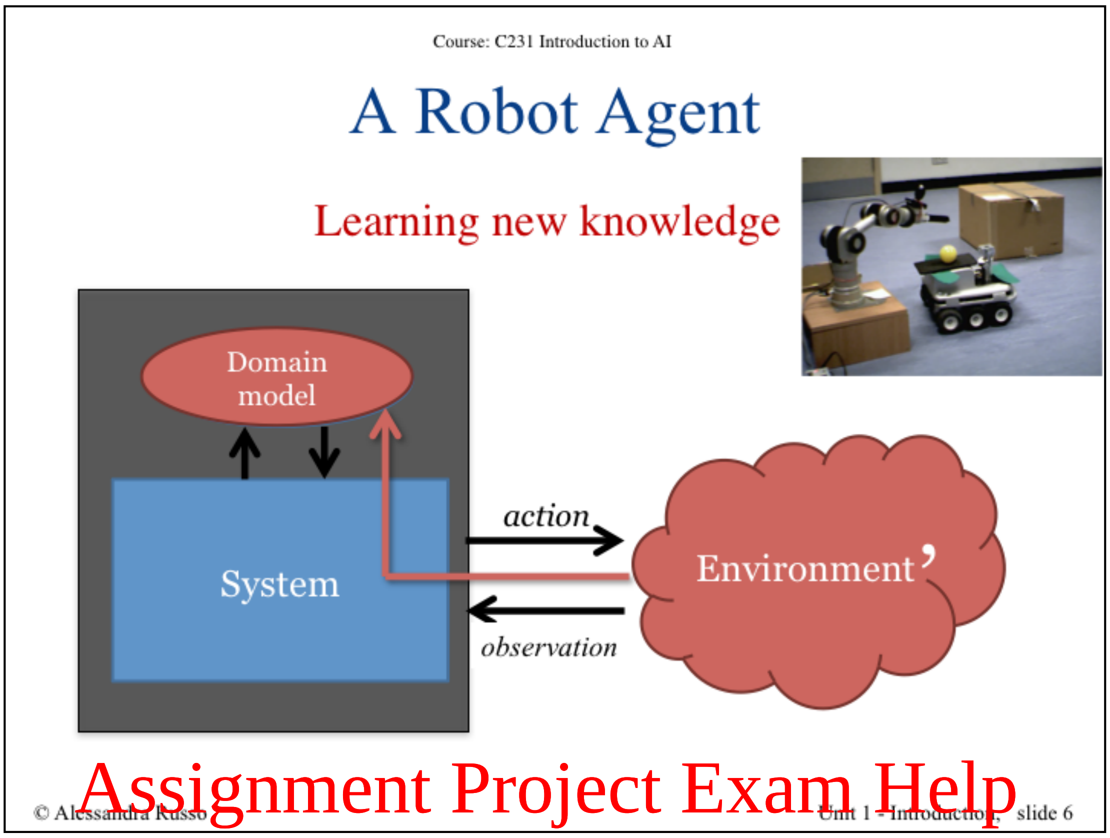
Consider for instance the case of a robot agent that has to perform the task of carrying a ball from one place to another in an room environment (goal). Its observations are success and failure of its moving actions. Past experiences are logs of previous actions performed in given states of the worlds and success and failure of those actions. Its prior knowledge is the effects of each of its actions, and abilities are the actions of "pickup" an object, "move" and "putdown" an object.

The agent in this case is intelligent because it is able to reason about the environment and construct plans, but also learn better models of its environment based on observations and past experience and use this knowledge to decide what is the best action to perform at each current time point given the state in which the environment and the robot are. Past experience and observations, as the environment changes, provide new evidence to the robot, which it then uses to learn new knowledge about the environment.

A Robot Agent

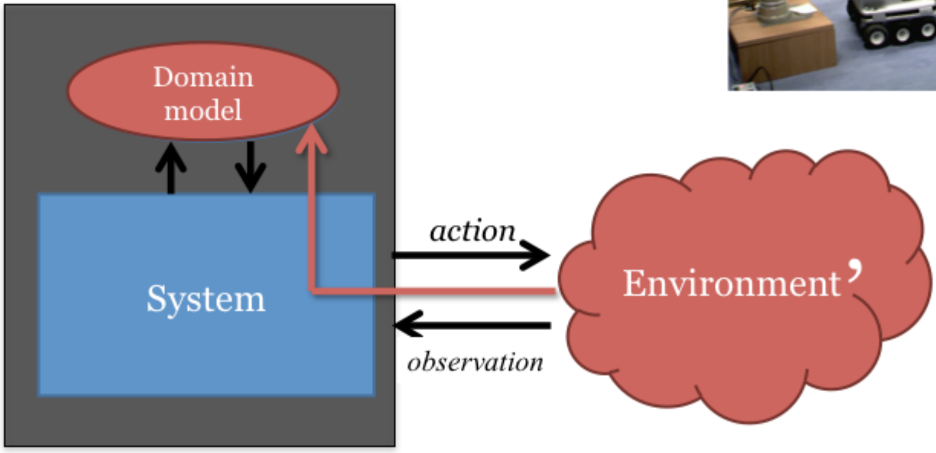
Exogenous Events

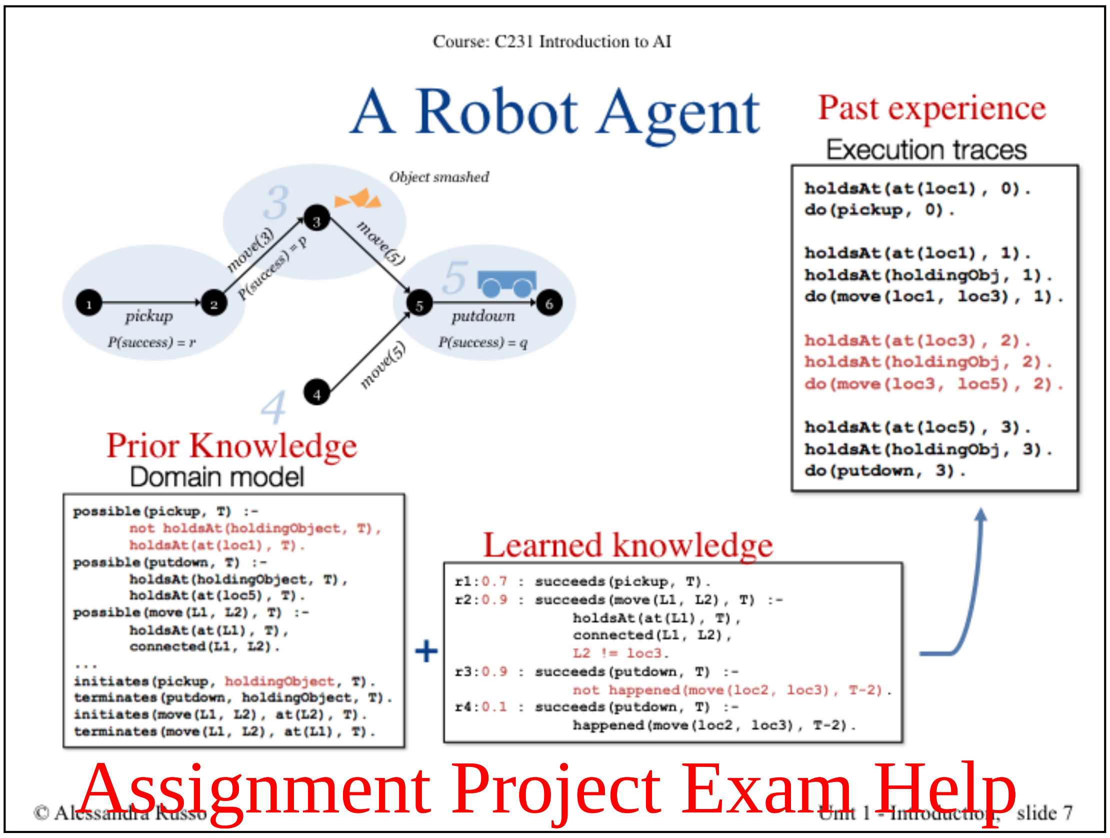




A Robot Agent

Learning new knowledge





3
Object smashed
move(3)
P(success) = p
move(5)
5
pickup
P(success) = r
putdown
P(success) = q
move(5)
4

A Robot Agent

Past experience

Execution traces

```
holdsAt(at(loc1), 0).
do(pickup, 0).

holdsAt(at(loc1), 1).
holdsAt(holdingObj, 1).
do(move(loc1, loc3), 1).

holdsAt(at(loc3), 2).
holdsAt(holdingObj, 2).
do(move(loc3, loc5), 2).

holdsAt(at(loc5), 3).
holdsAt(holdingObj, 3).
do(putdown, 3).
```

Prior Knowledge

Domain model

```
possible(pickup, T) :-
        not holdsAt(holdingObject, T),
        holdsAt(at(loc1), T).
possible(putdown, T) :-
        holdsAt(holdingObject, T),
        holdsAt(at(loc5), T).
possible(move(L1, L2), T) :-
        holdsAt(at(L1), T),
        connected(L1, L2).
...
initiates(pickup, holdingObject, T).
terminates(putdown, holdingObject, T).
initiates(move(L1, L2), at(L2), T).
terminates(move(L1, L2), at(L1), T).
```

+

Learned knowledge

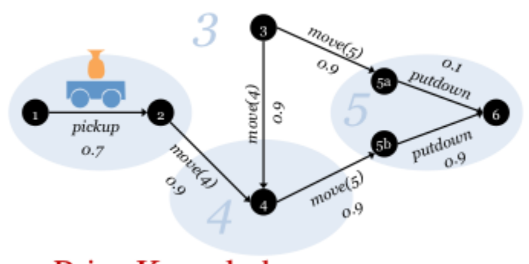
```
r1:0.7 : succeeds(pickup, T).
r2:0.9 : succeeds(move(L1, L2), T) :-
              holdsAt(at(L1), T),
              connected(L1, L2),
              L2 != loc3.
r3:0.9 : succeeds(putdown, T) :-
              not happened(move(loc2, loc3), T-2).
r4:0.1 : succeeds(putdown, T) :-
              happened(move(loc2, loc3), T-2).
```

This is just an example of how a prior (or domain) knowledge can be represented in the case of the robot agent moving in an environment. You don't need to know this representation of understand the full details here. The important message is the fact that the robot collects logs of executions over time some of which lead to successful task and some of which do not (as for example moving to a position 3 in the environment where the object breaks). The knowledge learned about what action is successful under what conditions and what is the likelihood of each of these learned rules.

The probabilities are learned so explain as much as possible the collected observations.

Course: C231 Introduction to AI

A Robot Agent



Past experience
Execution traces

```
holdsAt(at(loc1), 0).
do(pickup, 0).

holdsAt(at(loc1), 1).
holdsAt(holdingObj, 1).
do(move(loc1, loc3), 1).

holdsAt(at(loc3), 2).
holdsAt(holdingObj, 2).
do(move(loc3, loc5), 2).

holdsAt(at(loc5), 3).
holdsAt(holdingObj, 3).
do(putdown, 3).
```

Prior Knowledge
Domain model

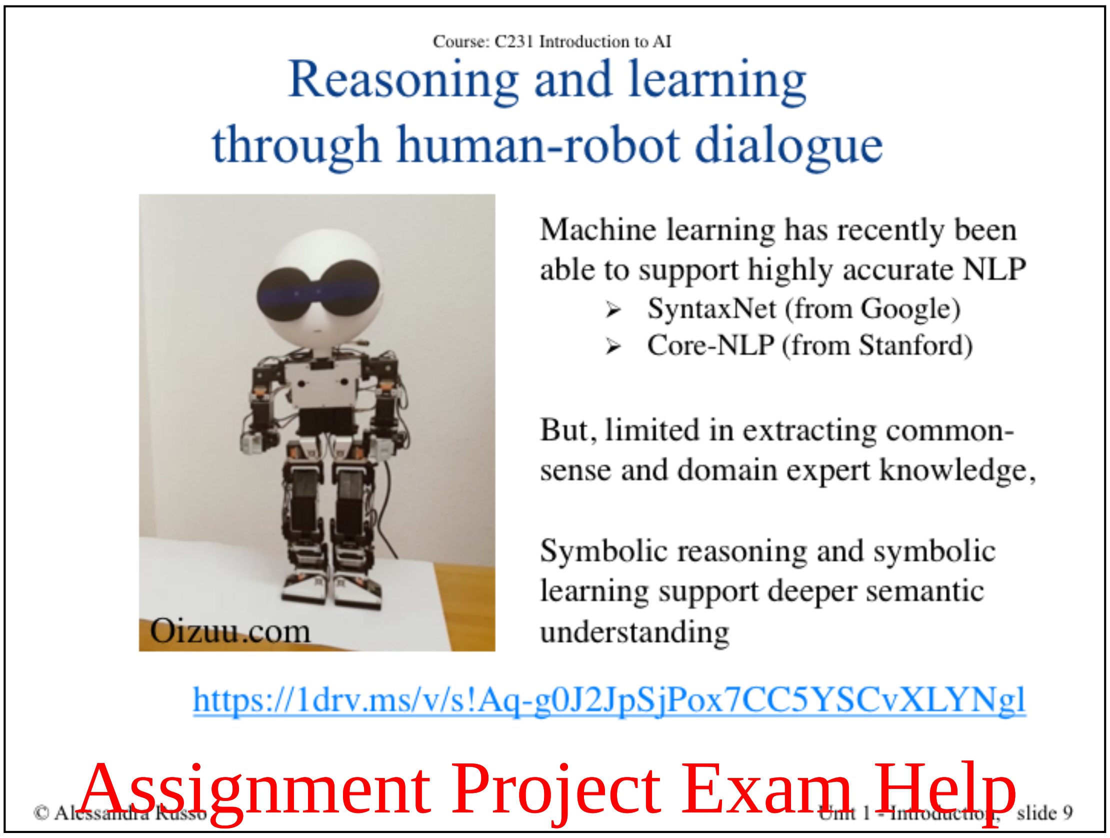
```
possible(pickup, T) :-
        not holdsAt(holdingObject, T),
        holdsAt(at(loc1), T).
possible(putdown, T) :-
        holdsAt(holdingObject, T),
        holdsAt(at(loc5), T).
possible(move(L1, L2), T) :-
        holdsAt(at(L1), T),
        connected(L1, L2).
...
initiates(pickup, holdingObject, T).
terminates(putdown, holdingObject, T).
initiates(move(L1, L2), at(L2), T).
terminates(move(L1, L2), at(L1), T).
```

+

Learned knowledge

```
r1:0.7 : succeeds(pickup, T).
r2:0.9 : succeeds(move(L1, L2), T) :-
              holdsAt(at(L1), T),
              connected(L1, L2),
              L2 != loc3.
r3:0.9 : succeeds(putdown, T) :-
              not happened(move(loc2, loc3), T-2).
r4:0.1 : succeeds(putdown, T) :-
              happened(move(loc2, loc3), T-2).
```

The learned knowledge may represent a new model of the environments and actions to perform in this environment.



Reasoning and learning through human-robot dialogue

Machine learning has recently been able to support highly accurate NLP
- SyntaxNet (from Google)
- Core-NLP (from Stanford)

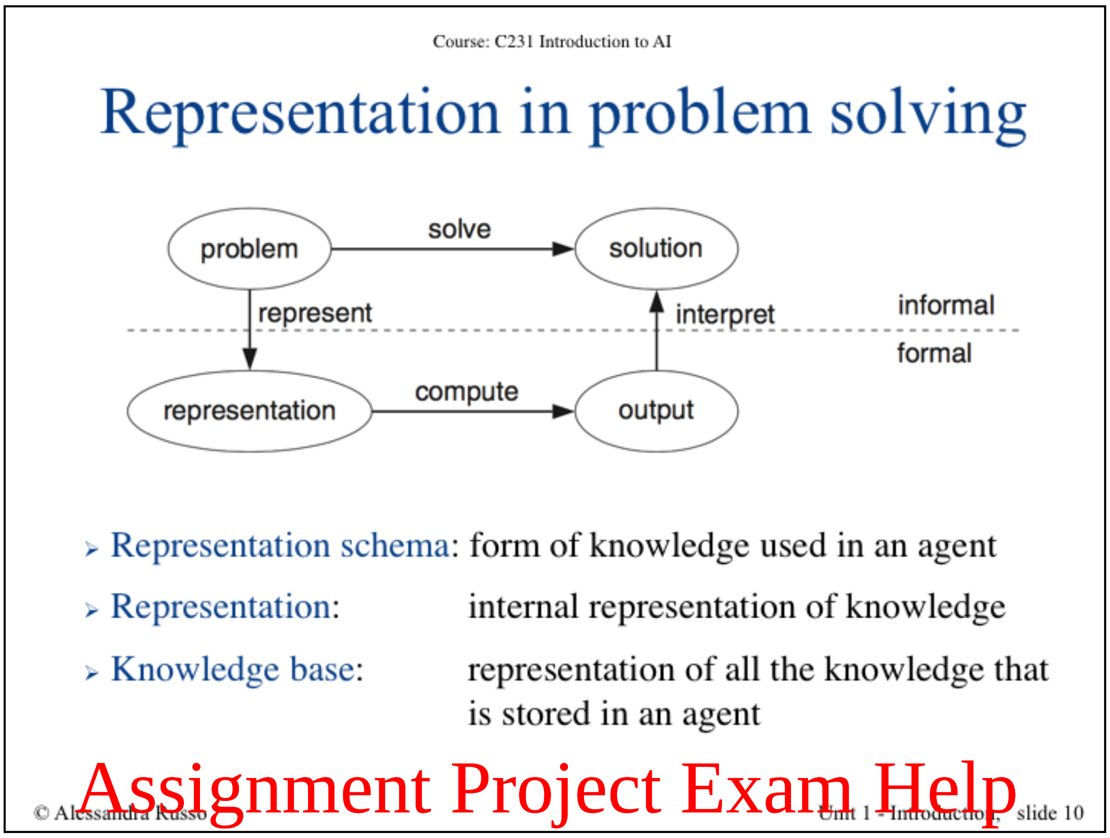
But, limited in extracting common-sense and domain expert knowledge,

Symbolic reasoning and symbolic learning support deeper semantic understanding

https://1drv.ms/v/s!Aq-g0J2JpSjPox7CC5YSCvXLYNgl

This is an example of knowledge representation and AI solutions such as symbolic machine learning, could be used to allow a robot agent to sustain human-machine dialogue, and to learn common-sense knowledge or domain specific knowledge through dialogue.

This work was done as student project by an MSc student in summer 2016.



Course: C231 Introduction to AI

Representation in problem solving

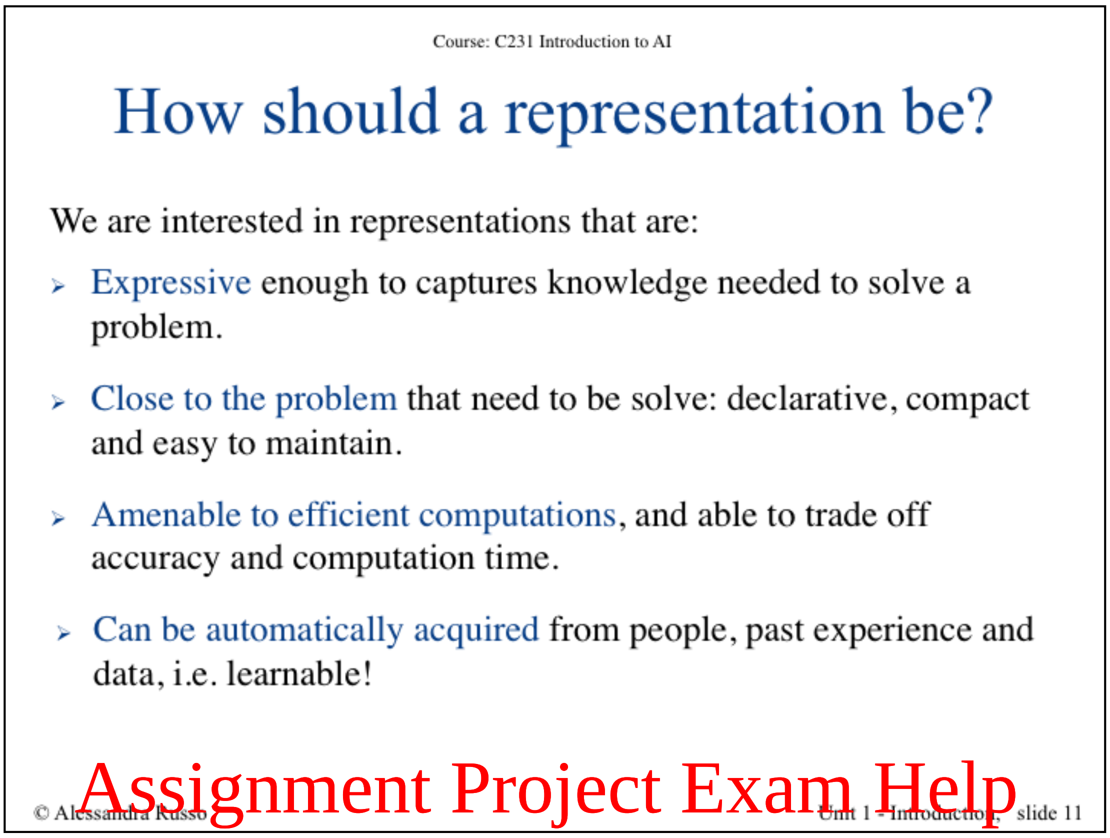
$$\begin{tikzcd}
\text{problem} \arrow[r, "\text{solve}"] \arrow[d, "\text{represent}"] & \text{solution} & \text{informal} \\
\text{representation} \arrow[r, "\text{compute}"] & \text{output} \arrow[u, "\text{interpret}"'] & \text{formal}
\end{tikzcd}$$

- Representation schema: form of knowledge used in an agent
- Representation: internal representation of knowledge
- Knowledge base: representation of all the knowledge that is stored in an agent

© Alessandra Russo Unit 1 - Introduction, slide 10

This slide describes the general framework for solving problems by computer. To solve a problem, we must first define the task and determine what constitutes a solution. This is normally done informally (see top part of the diagram above). We need then to represent the problem in a language with which a computer can reason. Use the computer to compute an output. This can be an answer, which can be presented to a user (as it is the case of a digital assistant) or a sequence of actions to be carried out in the environment (as it is the case for a robot agent). The output has then to be interpreted as a solution to the problem.

Knowledge is the information needed to solve a problem in a given domain. So, as part of the design (or modelling) of a computational agent capable of solving a problem, we must define how the knowledge will be represented in the agent. There are different forms of representation of knowledge. These are referred to as **knowledge schemas**. You have, for instance, seen in the first part of the course the logic programming formalism. This is an example of representation schema, whereby knowledge (or information) is stored or represented into an agent as set of clauses or Prolog rules. Many different forms of representation of knowledge can be used, e.g. symbolic, probabilistic, statistical, etc. Each of these forms may be more suited to certain tasks and not others. For instance, in domains of "certainty", a symbolic, logic-based form of representation of knowledge is appropriate. Different are the cases of approximate domains and uncertainty. Probabilistic forms of representation of knowledge might be more suitable. Once we fix a representation schema, then the formalisation of a piece of knowledge in that schema is called **representation** of the knowledge. Different schemas give rise to different representations of the knowledge related to a given problem. The representation of the entire knowledge that is stored in an agent is instead referred to as **knowledge base** of the agent.



Course: C231 Introduction to AI

How should a representation be?

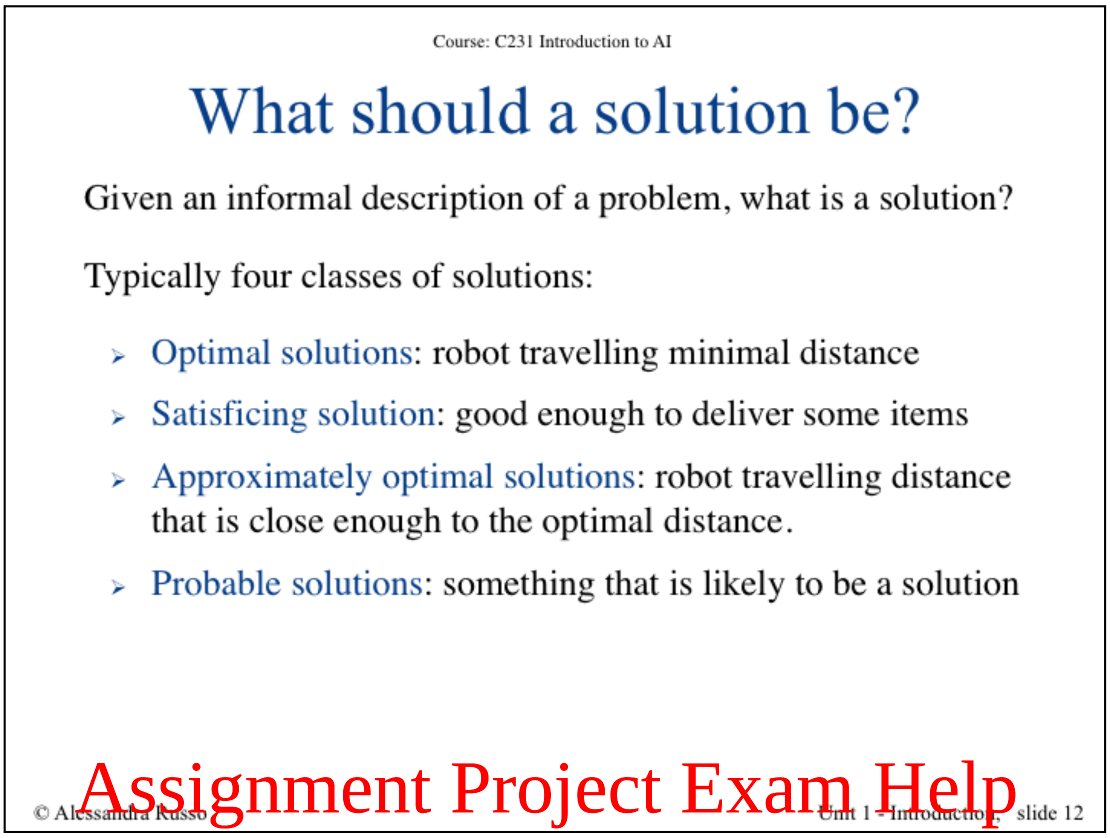
We are interested in representations that are:

- Expressive enough to captures knowledge needed to solve a problem.
- Close to the problem that need to be solve: declarative, compact and easy to maintain.
- Amenable to efficient computations, and able to trade off accuracy and computation time.
- Can be automatically acquired from people, past experience and data, i.e. learnable!

© Alessandra Russo Unit 1 - Introduction, slide 11

A representation should be

- *rich enough to express the knowledge needed to solve the problem*. In the first part of this course, you have seen logic programming as a symbolic approach for representing knowledge. This is a reasonably expressing language where, for instance, common sense notions (typically used in digital assistant), default assumptions, used in all types of agents, could be expressed by means of negation as failure. In this second part of the course, we will look a much more expressive language, called **Answer Set Programming** (ASP), that provides a truly declarative approach for representing knowledge, hard constraints, weak constraints, and optimisation statements.
- *close to the problem.* Declarative formalisms such as logic programming, ASP and logic in general, are as close to a problem description as possible. ASP in particular is known to be more declarative than logic programming as it does not mix in its representation style procedural aspects for solving a problem, as well as declarative aspects for defining the problem. It is purely descriptive and can be used to describe a problem solution in terms of its properties and specification. Because of this, ASP is also easy to relate to a given problem domain than Prolog representations. It is also very compact in its representation and natural. We will see later on why this is the case.
- *amenable to efficient computation*, which usually means that it is able to express features of the problem that can be exploited for computational gain and able to trade off accuracy and computation time. Depending on the type of formalisms, accuracy of solutions may or may not be something that can be trade. For instance, in logic programming, the representation is underpinned by a precise semantic and solutions, when they exist, are always guaranteed to be 100% precise. In the case of ASP, instead, optimisation statements can be used to generate optimal and suboptimal solutions. Computation time can be traded off with accuracy. Fast computations may look for sub-optimal solutions instead of "the" optimal solutions.
- *able to be acquired from people, data and past experiences*. This feature refers to the ability of an agent to learn knowledge automatically. In the logic-based learning course, which is taught in the third year, you will be able to see how you can automatically learn knowledge represented as ASP programs from given data and past experiences.


Course: C231 Introduction to AI

What should a solution be?

Given an informal description of a problem, what is a solution?

Typically four classes of solutions:

- Optimal solutions: robot travelling minimal distance
- Satisficing solution: good enough to deliver some items
- Approximately optimal solutions: robot travelling distance that is close enough to the optimal distance.
- Probable solutions: something that is likely to be a solution

© Alessandra Russo Unit 1 - Introduction, slide 12

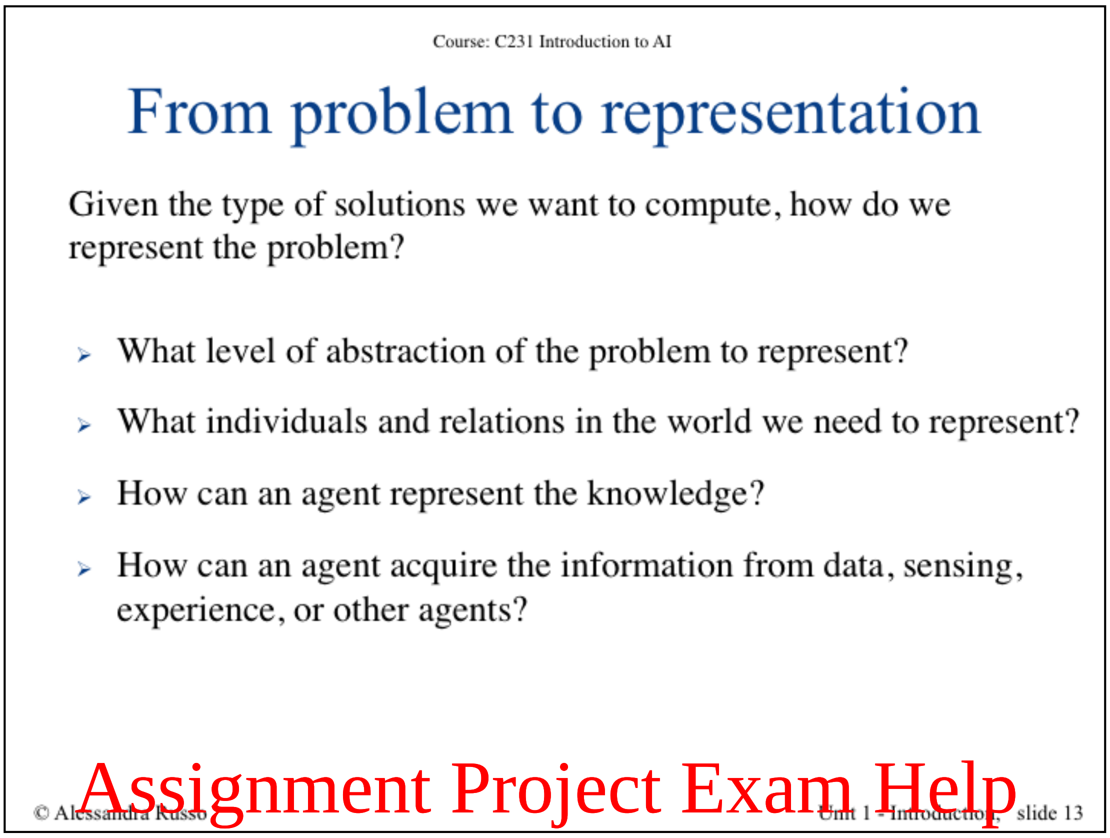
Before representing the knowledge that is needed to solve a problem in a given domain, it is important to define what would constitute a solution of the given problem. Typically, there are four common classes of solutions:

Optimal Solutions: this are the ones that are the best solutions according to some criteria for measuring the quality of solutions. These criteria can be *qualitative* (also said ordinal) measures (e.g. a robot that takes out "as much trash" as possible, the more trash the better is the solution), or *quantitative* (also said cardinal) measures (e.g. a delivery robot, has to deliver as many packets but minimizing the distance travelled.

Satisficing solutions: sometime it is not necessary to find the best optimal solution but just some solution. A satisficing solution is one that is good enough according to some notion of what is an adequate solution. For example, a person may tell a robot that it must take all of trash out, or tell it to take out three items of trash. Or, in the case of a natural language interactions with digital personal assistant, satisficing solutions are those systems that are able to answer most of the questions in a satisficing manner.

Approximately optimal solution: these are solutions that are close enough to optimal solutions. Agents may not need optimal solutions to problems; they only need to get close enough. For example, a delivery robot may not need to travel the optimal distance to take out the trash but may only need to be within, say, 10% of the optimal distance.

Probable solution: A probable solution is one that, even though it may not actually be a solution to the problem, is likely to be a solution. This is one way to approximate, in a precise manner, a satisficing solution. For example, in the case where the delivery robot could drop the trash or fail to pick it up when it attempts to, we may need the robot to be 80% sure that it has picked up some items of trash.



Course: C231 Introduction to AI

From problem to representation

Given the type of solutions we want to compute, how do we represent the problem?

- What level of abstraction of the problem to represent?
- What individuals and relations in the world we need to represent?
- How can an agent represent the knowledge?
- How can an agent acquire the information from data, sensing, experience, or other agents?

© Alessandra Russo Unit 1 - Introduction, slide 13

Once we have an idea of the nature of a solution, we can choose the level of representation of the problem so that a machine can solve the problem and compute the type of required solutions. The representation of the problem constitutes essentially **a model of the world** in which an intelligent agent has to operate to solve its task.

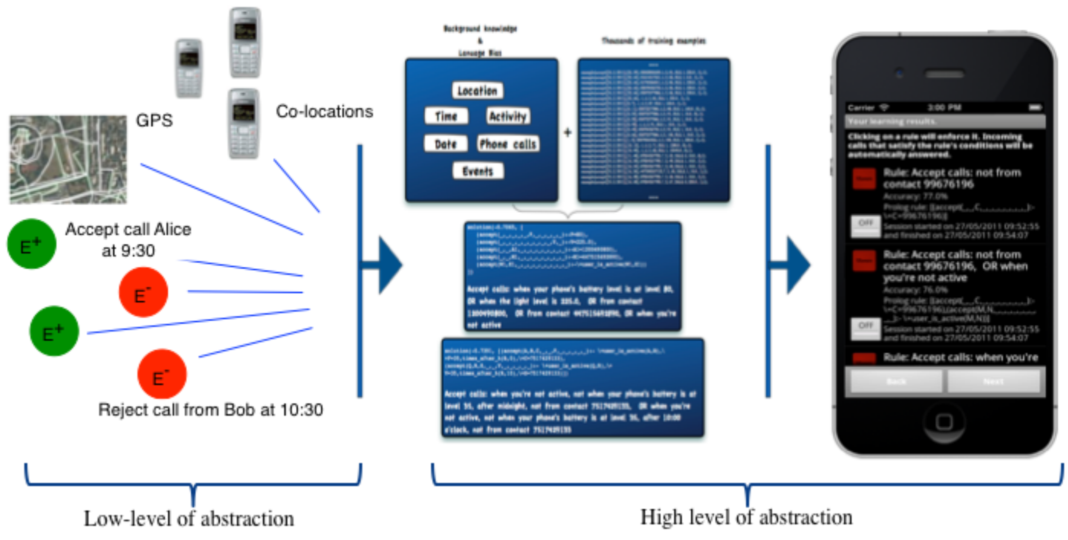
So, we can define **a model** to be a representation of the specifics of what is true and false in the world.

The world does not have to be modeled at the most detailed level to be useful. All models are abstractions; they represent only part of the world and leave out many of the details. An agent can have a very simplistic model of the world, or it can have a very detailed model of the world. *A lower-level abstraction includes more details than a higher-level abstraction.* An agent can have multiple, even contradictory, models of the world (see argumentation models in the four year course).

In general, there isn't a best level of abstraction, and the best way to judge whether a model is a good representation is if it is useful to solve a given task.

Course: C231 Introduction to AI

Choosing level of abstraction



Model the problem with multiple levels of abstraction.

Choosing an appropriate level of abstraction is difficult:

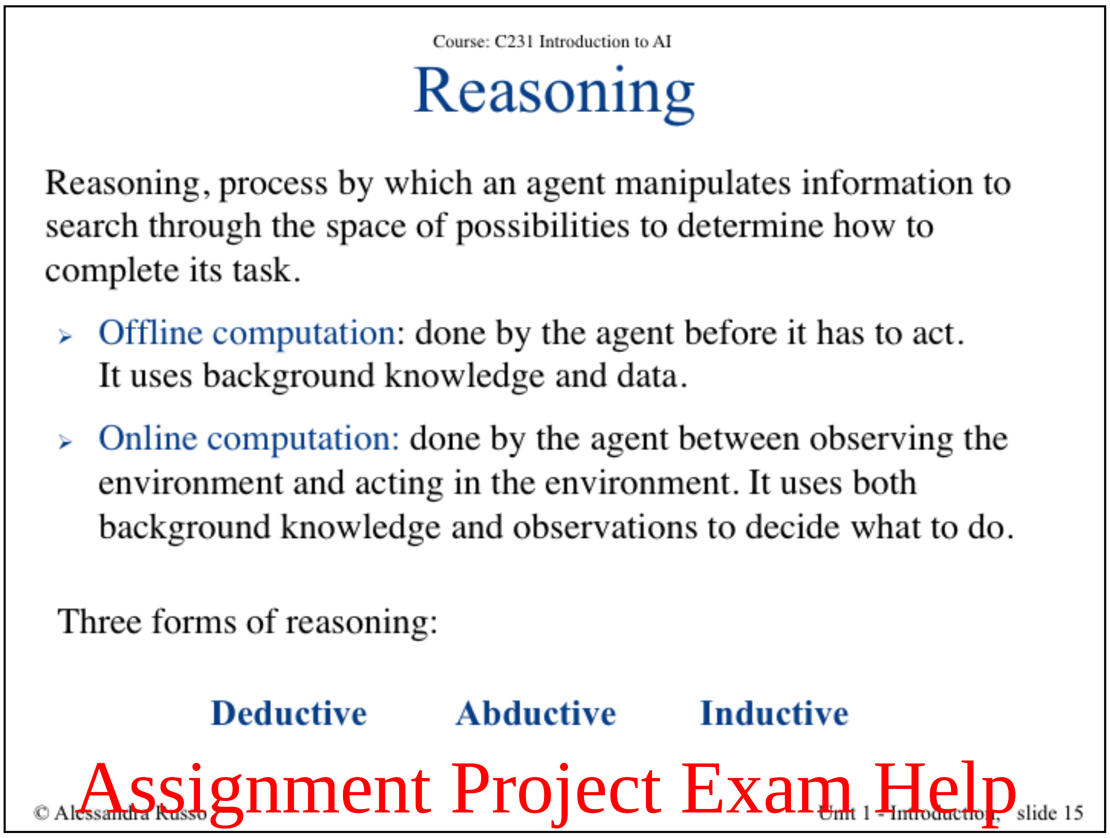
• high-level description, such the natural language example you have seen in the short video, as a form of explanation that a system can be give to a human, is easier for a human to understand the system's solution and also easier for a human to describe the problem and give feedback. But it is harder for a machine to fully comprehend due to the high level of ambiguity and also the common-sense knowledge that is often implicitly assumed by humans when they communicate in NL. Despite the tremendous progress witnessed so far in NLP , we are still far for representing a problem in NL directly, as machines are still not able to rigorously reason directly with NL.

• a low-level description can be more accurate and more predictive. Often high-level descriptions abstract away details that may be important for actually solving the problem. So certain type of problems can be solved only if detailed models are given, but

• the lower the level, the more difficult it is to reason with. This is because a solution at a lower level of detail involves more steps and many more possible courses of action exist from which to choose.

• you may not know the information needed for a low-level description as it is too detailed and environment changes at micro-level more than at abstract level. Consider for instance the GPS location of a phone. At a low-level representation of the problem, GPS data change very frequently whereas at a higher-level of representation the state property of a human walking persists. So reasoning about human behaviors at low-level of representation is clearly harder that reasoning using the more abstract representation of similar concepts (e.g. person walking).

The example given in this slide shows different level of abstraction in the context of learning (privacy) policies about human behaviors with mobile phones. On the left, data are collected by sensors. The low-level representation is quite detailed, imagine for instance GPS coordinates versus GPS names. But all these data can be used by an intelligent agent to learn user behaviors from contextual low-level data information and user's past actions. The outcome is high-level representation of policies that the system can communicate back to the user for validation purposes. These policies are then used by the system to self-adapt (at run-time). Models in these cases can have a mix level of representation. It is often a good idea to model a given problem at multiple levels of abstraction.


Course: C231 Introduction to AI

Reasoning

Reasoning, process by which an agent manipulates information to search through the space of possibilities to determine how to complete its task.

- Offline computation: done by the agent before it has to act. It uses background knowledge and data.
- Online computation: done by the agent between observing the environment and acting in the environment. It uses both background knowledge and observations to decide what to do.

Three forms of reasoning:

Deductive **Abductive** **Inductive**

© Alessandra Russo Unit 1 - Introduction, slide 15

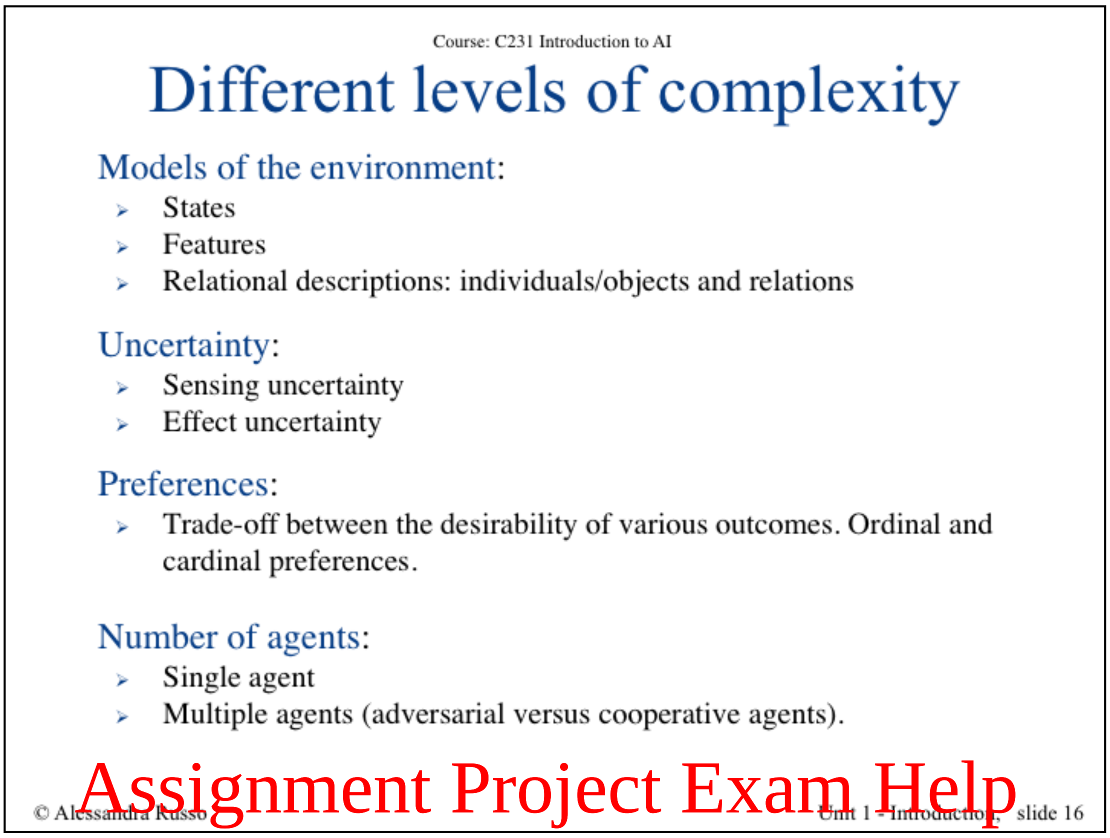
The term "reasoning" is used to denote the process made by a computational agent when searching for possible ways to determine how to complete its task. Depending of the tasks, for instance, these possible ways can be actions, as it is in the case of a delivery robot that has to choose the next physical action needed to complete its task, or computing (diagnostic) answers, as it is in the case of a diagnostic assistant that has to compute possible diagnosis for some given symptoms.

The reasoning process, in general, can be performed by a computational agent either "offline" or "online".

Offline reasoning assumes that the background knowledge needed to solve the task is precomputed. Agents receive, at design time, this knowledge and (environment) data and perform reasoning to solve its task. This is for instance the case of computational agents such as a manufacturing robots. The knowledge is all precompiled so that the agent can perform simple offline reasoning and perform its task in a predefined constrained environment. Examples are car painting robots. They are very specialized agents that do not adapt well to different environments or to changing tasks.

Online reasoning, instead, allow agents to "sense" the environment in which they operate so to collect online data during the execution and use these data, together with the given background knowledge, to decide at execution time what action to take. The reasoning in this case is performed between the sensing of environment data and the action. This is for instance the case of both a delivery robot or a diagnostic assistant. In the latter case the environment would be the human who provide information to the systems as new sensed data.

In general, a reasoning process can be of different types: deductive, abductive and inductive. In this course we will cover the first two types of reasoning in detail. Inductive reasoning will instead be covered in the 3rd year course on "Logic-based learning".


Course: C231 Introduction to AI

Different levels of complexity

Models of the environment:

- States
- Features
- Relational descriptions: individuals/objects and relations

Uncertainty:

- Sensing uncertainty
- Effect uncertainty

Preferences:

- Trade-off between the desirability of various outcomes. Ordinal and cardinal preferences.

Number of agents:

- Single agent
- Multiple agents (adversarial versus cooperative agents).

© Alessandra Russo Unit 1 - Introduction, slide 16

When designing a computational agent, we have to make decisions on different aspects of the system that can affect the level of computational complexity of the agent.

In this slide we highlight some of the key aspects that we will need to take into account when we develop an AI solution to a given problem:

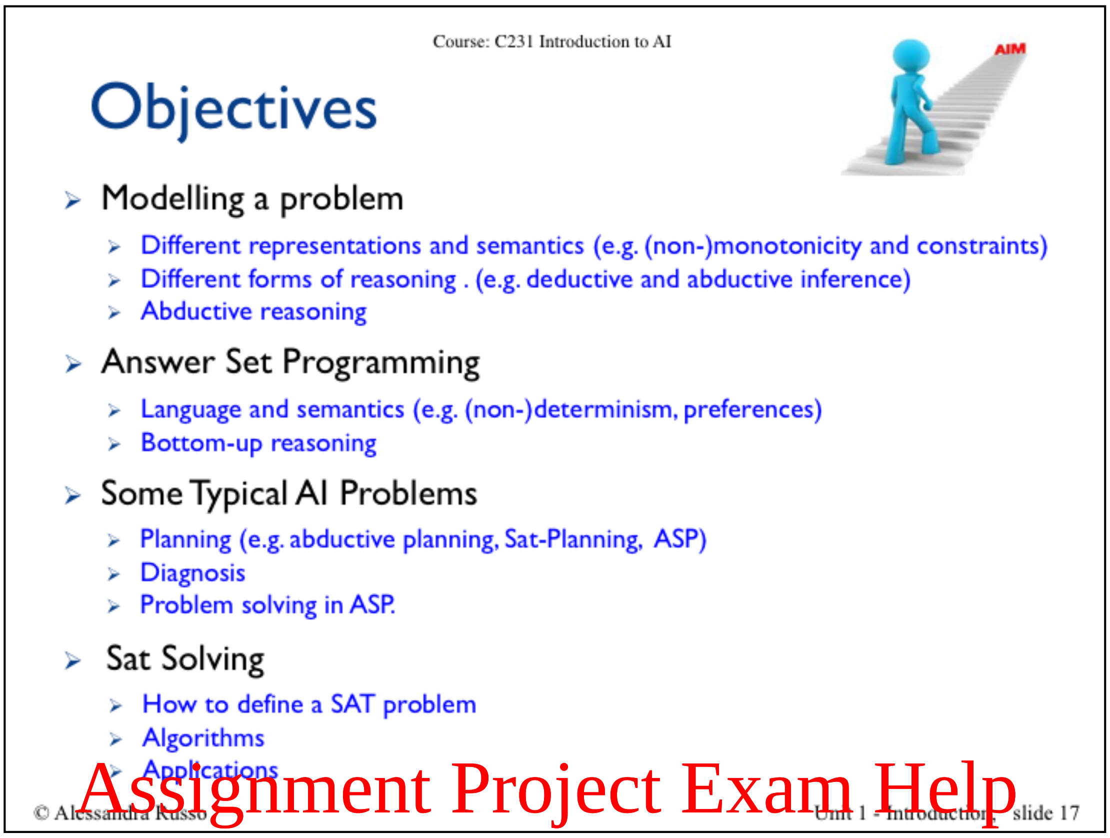
1) **Model of the environment**. These can be expressed in terms of states, features and/or individuals and relations between individuals. For example, in the robot agent shown in slide 4, the model of the environment is expressed in terms of states and the agent reasons about the states the environment can be in when it has to decide which action to take. In the case of a diagnostic assistant, it might be more appropriate to consider features and features values instead of individual states, as they may be too many.

2) **Uncertainty**. This can be of two types, uncertainty on the state and uncertainty of effects of actions. In the first case, perception of the state in which the environment is maybe "certain" and the agent known exactly in which state the environment is (we call this **full observable state**), or the perception of the state the environment may be noisy, in which case that can be more than one possible state of the environment that can lead to the same agent's perception (we call this **partially observable state**). For instance, in the case of a diagnostic assistant, the agent cannot really sense the disease of the patient, (patient is in this case the environment and the diseases possible states), but multiple diseases may give the same symptoms. So the agent can only reason with the set of possible diseases. Uncertainty may also be on effects of actions. Effects of actions can be deterministic (next state uniquely determined by the current state and the action) or effects maybe stochastic (when we have possible next states and the agent knows some probability distribution over the next states). This is the case in slides 7 and 8.

3) **Preferences**. These occur when there are trade-off between the desirability of possible outcomes. For instance, in the robot agent it is more important to carry the item without breaking than to take the shortest path in the environment, as this might lead to states where there is a risk of dropping the item. They can be ordinal, when we need just a ranking or ordering among goals or actions; and cardinal, when the magnitude of the values matters (e.g. costs).

4) **Number of agents**. Some problem need to be solve by an agent taking into account also the reasoning/acting of other agents in the environment. The other agents may act to trick or maybe willing to

cooperate. You have seen examples in the first part of this course.


Course: C231 Introduction to AI

Objectives

- Modelling a problem
  - Different representations and semantics (e.g. (non-)monotonicity and constraints)
  - Different forms of reasoning . (e.g. deductive and abductive inference)
  - Abductive reasoning
- Answer Set Programming
  - Language and semantics (e.g. (non-)determinism, preferences)
  - Bottom-up reasoning
- Some Typical AI Problems
  - Planning (e.g. abductive planning, Sat-Planning, ASP)
  - Diagnosis
  - Problem solving in ASP.
- Sat Solving
  - How to define a SAT problem
  - Algorithms
  - Applications

© Alessandra Russo   Unit 1 - Introduction   slide 17

We have seen that from an AI point of view, **representation** of a problem and **reasoning** are key elements. In the first part of this course, you have covered different forms of representation considering in particular logic programming, as a main formalism for representing a problem domain and an agent's knowledge. You have also covered different semantics and looked at the deductive inference mechanisms typical for Prolog (i.e. SLD and SLDNF). These are normally goal-directed inference systems.

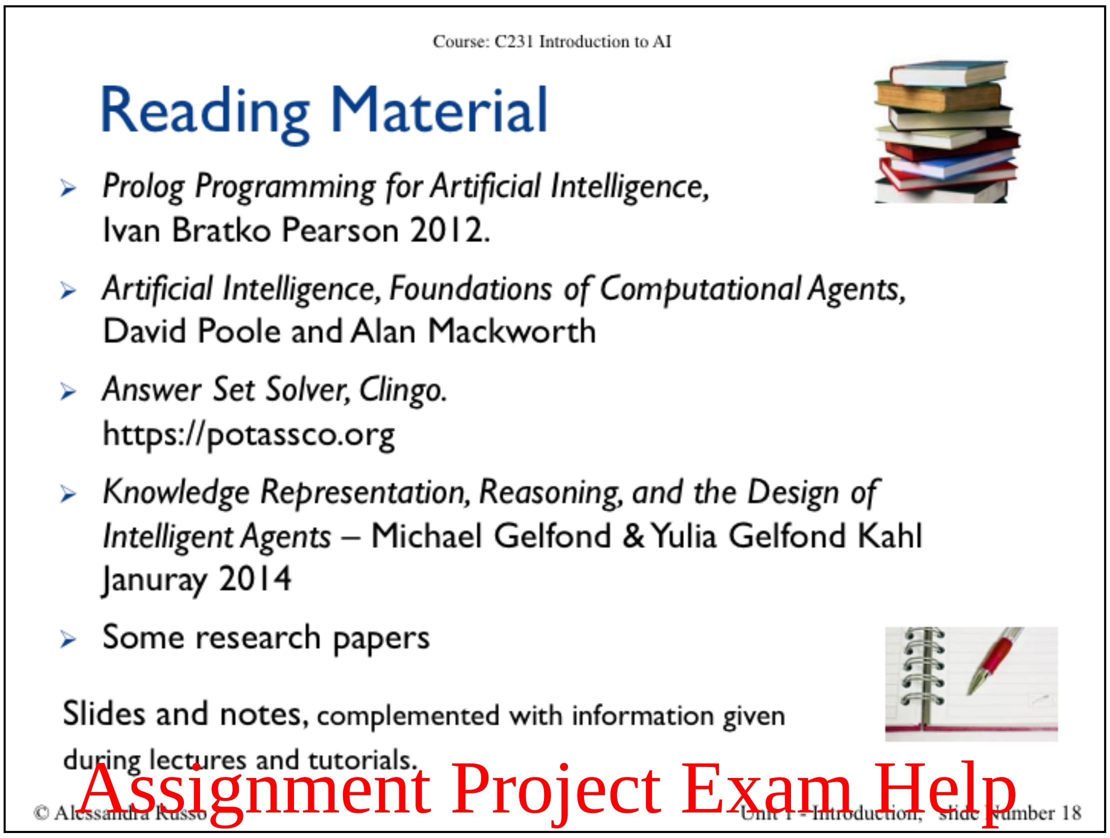
In this second part of this course we will look at a different form of inference, called **abductive inference**. This is typically used to solve AI problems such as diagnosis (as it is the case for a diagnostic assistant agent), explanation generation (which is at the core of the current "Explainable AI" trend) and off-line planning. Abductive reasoning can be computed in different ways depending on the type representation used to express the problem. We will cover two ways for computing abductive reasoning, a top-down approach that uses logic programming as representation formalism, and a bottom-up approach that uses Answer Set Programming.

We will also in particular cover **Answer Set Programming** (ASP), considered to be currently one of the most efficient declarative programming languages, also particularly suited for representing and solving AI problems. ASP uses a different semantics and a different reasoning process for computing solutions of a given problem. The reasoning process of ASP builds upon another traditional AI task, called **SAT solving**. So, we will also cover basic notions of SAT solving and explore more expressing mechanisms for representing and solving a SAT problem, including related algorithms. The formalisms covered in this case with be closer to full classical logic, than logic programming.

The **aims** of this second part of the course are:

- learn bottom-up formalism and semantics for representing knowledge in a computational agents
- learn foundation principles of planning and abductive reasoning
- familiarize with existing solvers such as ASP solver and SAT solvers.
- use solvers to solve planning problems.

At the end of this course you will have acquired fundamental principles, technologies in the AI areas of planning, knowledge, perception and cognitive computing, which will provide you with the essential building blocks for the AI courses in your third and forth year.



Course: C231 Introduction to AI

Reading Material

- *Prolog Programming for Artificial Intelligence,* Ivan Bratko Pearson 2012.
- *Artificial Intelligence, Foundations of Computational Agents,* David Poole and Alan Mackworth
- *Answer Set Solver, Clingo.* https://potassco.org
- *Knowledge Representation, Reasoning, and the Design of Intelligent Agents* – Michael Gelfond & Yulia Gelfond Kahl Januray 2014
- Some research papers

Slides and notes, complemented with information given during lectures and tutorials.

© Alessandra Russo Unit 1 - Introduction, slide number 18

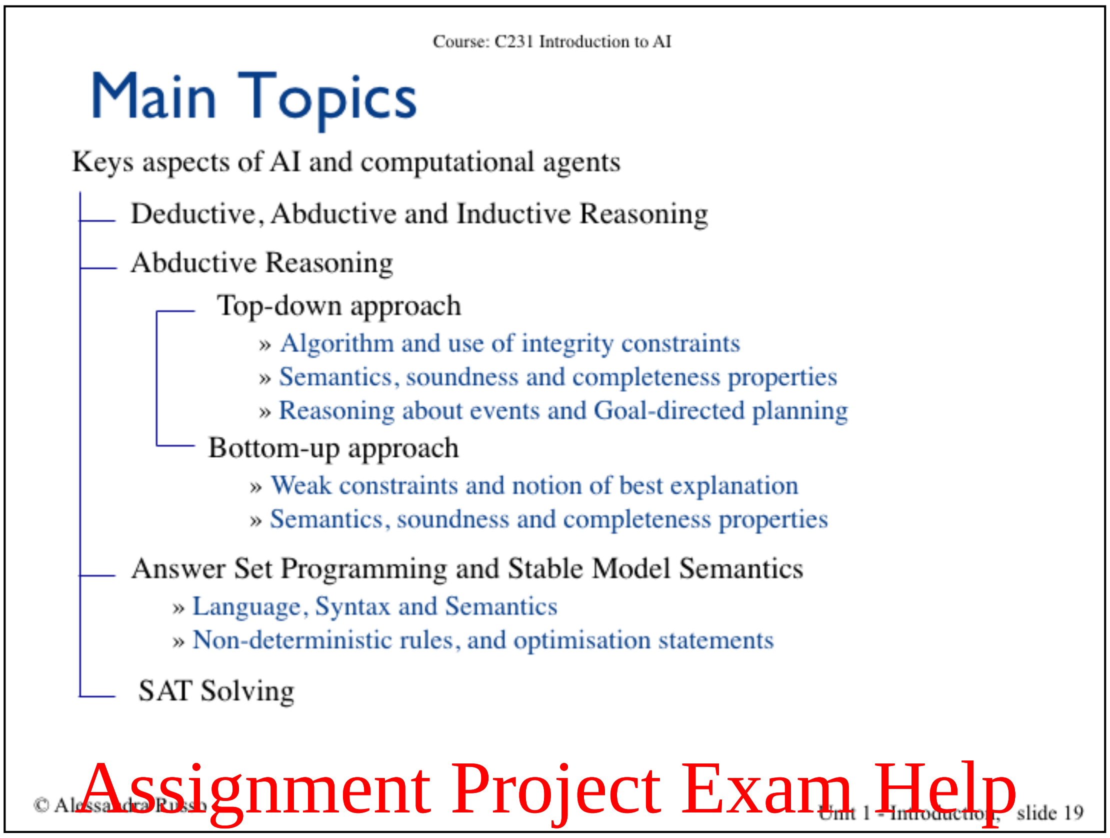
The first textbook is a good introduction to logic programming and basic notions of Prolog that we will be referring to during this course.

The second textbook is one of the main textbooks of this course. Notes on this lecture as well as other lectures to come are based on this textbook. So it's good read if you want to extend your knowledge on the topics covered in this course.

In the third reference is the link to ASP, where you will find the Clingo system and related manual. You will need to familiarise with the manual and system available on this Website as you will be doing some exercises and tutorials using Clingo. Note that the material is available on a Potsdam university's link, because a research group at this university has developed the Clingo system, which is now one of the most used programming environments for solving problems written in Answer Set Programming.

The fourth reference is another general textbook on knowledge representation. Some of these topics will covered in more detailed in the knowledge representation course in the forth year.

I will also suggest papers as additional reading if and where needed, during the course.


Course: C231 Introduction to AI

Main Topics

Keys aspects of AI and computational agents

- Deductive, Abductive and Inductive Reasoning
- Abductive Reasoning
  - Top-down approach
    - » Algorithm and use of integrity constraints
    - » Semantics, soundness and completeness properties
    - » Reasoning about events and Goal-directed planning
  - Bottom-up approach
    - » Weak constraints and notion of best explanation
    - » Semantics, soundness and completeness properties
- Answer Set Programming and Stable Model Semantics
  - » Language, Syntax and Semantics
  - » Non-deterministic rules, and optimisation statements
- SAT Solving

© Alessandra Russo Unit 1 - Introduction, slide 19

The above list of topics is not meant to follow any chronological order in which we will be covering them.