



Introduction to Artificial Intelligence

Week 1

My Contact

- ▶ Email: m.makhmutov@innopolis.ru
 - ▶ Please preface all emails about the course with the subject [IntroAI] to allow for a quick response
- ▶ Office 418
 - ▶ Office hours – Monday 4:30 – 6:00 PM
 - ▶ This is the BEST method for us to make contact and for you to receive a solution
- ▶ Please do not use direct telegram messages, only in case of urgent situations
- ▶ FYI
 - ▶ I sleep at night
 - ▶ Usually I will ask for larger issues for you to come see me



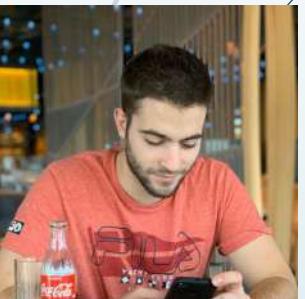
Team



- ▶ Munir Makhmutov – lecture instructor, TA (B22-CS-01, B22-CS-02)
 - ▶ m.makhmutov@innopolis.ru



- ▶ Darko Bozhinoski – tutorial instructor
 - ▶ d.bozhinoski@innopolis.ru



- ▶ Marko Pezer – TA (B22-CS-03, B22-CS-04, B22-CS-05)
 - ▶ m.pezer@innopolis.university



- ▶ Parth Kalkar – TA (B22-CS-06, B22-DSAI-01)
 - ▶ p.kalkar@innopolis.university

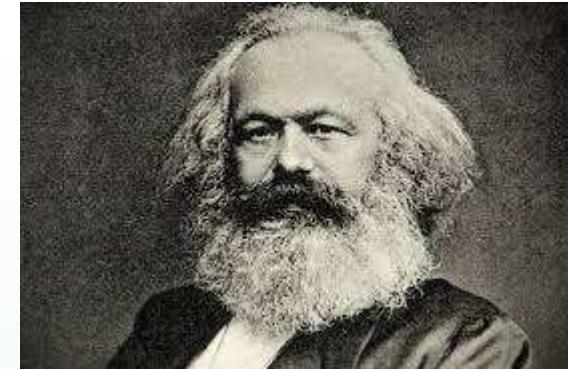


- ▶ ElDakrouy Karim – TA (B22-DSAI-02, B22-DSAI-03)
 - ▶ k.eldakrouy@innopolis.university



- ▶ Jayveersinh Raj – TA (B22-DSAI-04)
 - ▶ j.raj@innopolis.university

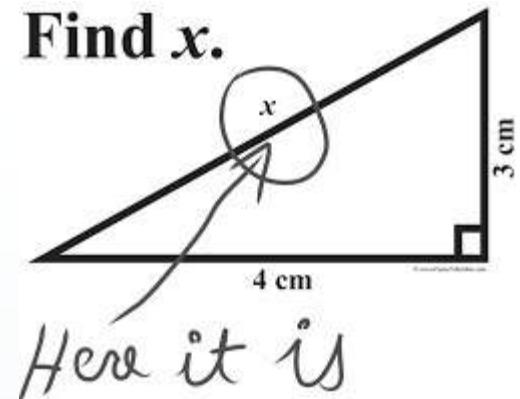
Marks Breakdown



- ▶ Lab Attendance – 10%
 - ▶ No lab during first and midterm weeks
 - ▶ Students are allowed to miss up to 2 labs without penalty to grade
 - ▶ I don't care if this is due to club reasons, illness, just don't feel like going
- ▶ Assignment 1 – 20%
- ▶ Assignment 2 – 20%
- ▶ Midterm Exam – 25%
- ▶ Final Exam – 25%
- ▶ Bonus – 5%
 - ▶ Given based on instructor's and TA's impression

Course Procedures

- ▶ As a matter of course I will not accept late assignments or missed tests without prior legitimate excuses given in a reasonable period before the due date
- ▶ Not legitimate
 - ▶ I was working on a project/hackathon for X and didn't do the work for this class
 - ▶ I was sick due to my own hand (*i.e.* too much parties)
 - ▶ My computer broke two weeks ago and I couldn't work on the assignment
- ▶ There is short time period between final exam and grades being locked – any requests beyond this period will be not heard
 - ▶ I do not regrade based on “but I need more marks to get the grade of X”
 - ▶ Legitimate reasons for regrade
 - ▶ Marks not added correctly
 - ▶ Mistake by the marker based in fact
 - ▶ note if we cannot understand what you wrote as you intended then we cannot regrade it
 - ▶ You may always request feedback or clarification
 - ▶ We want you to learn from any mistakes



Academic Misconduct

- ▶ **Academic misconduct** is any action or attempted action that may result in creating an unfair **academic** advantage for oneself or an unfair **academic** advantage or disadvantage for any other member or members of the **academic** community. – UC Berkeley
- ▶ I have a very long history in working on academic misconduct cases



- ▶ Cheating – copying or communicating answers to an evaluation
- ▶ Plagiarism – using someone else's work as your own
- ▶ Work for hires – hiring out your course work to a third party
- ▶ Submission of work for which credit has been obtained upon previously
- ▶ Possession of unauthorised aids
- ▶ Preventing any other students from accessing materials for an academic advantage
- ▶ Falsification of records or Impersonation
- ▶ Disruption to classrooms – e.g. pulling a fire alarm to get out of a test; failure to follow instructions during a test (i.e. pencils down)
- ▶ Accomplices – enabling or acting as a knowing or negligent party to any of the above

Academic Misconduct Policy at IU

- ▶ Innopolis policy (minimums)
 - ▶ First offence is 0 on the unit and note in record
 - ▶ Second offence is 0 in the course and note in record
 - ▶ Third is immediate expulsion
 - ▶ Even on a first offense on a major unit such as a final exam this may be enough to lead to an expulsion due to low grades
- ▶ Note – we reserve the right to increase penalties in egregious cases
- ▶ I reserve the right to use technical methods to detect plagiarism such as MOSS or Turn-it-in



What are You Allowed to Do

- ▶ Cite sources in essays, code, etc. when you use them
- ▶ Study for a test with other students in the class
- ▶ Ask another student to look at your code as there is a bug you can't figure out – just don't have them fix it for you though
- ▶ Demonstrate to another student the method of solving a problem conceptually – don't write their answer for them
- ▶ Ask for help from TAs and/or the Instructors when you are having difficulties

Course Materials

- ▶ Main Text
 - ▶ Artificial Intelligence a Modern Approach 3rd Edition – Russel and Norvig
- ▶ Secondary Text
 - ▶ Computational Intelligence For Modeling and Optimization - Ashlock

Tips for Success

- ▶ Read the syllabus
 - ▶ Lots of answers to many of life's (or at least the course administrations) questions are in it
- ▶ Read the book
- ▶ Attempt all questions
- ▶ Study a little each week – don't think you can just cram on the last day
- ▶ Form a regular study group and meet to discuss the assignments and tests
- ▶ Come to office hours after attempting the question and ideally have some plan of attack
 - ▶ if you're the TA/Professor what 'question' would you want to assist with more?
 - ▶ "I don't know how to do Question X!"
 - ▶ "I think Question X can be solved by Y method but I cannot see how to make step Z to occur - can you suggest how to accomplish this?"
- ▶ Try to find out where these techniques are being used for an issue/problem you care about

Attendance

- ▶ You are expected to attend lectures, tutorials and labs
- ▶ We reserve the right to maintain attendance lists in lecture and tutorial and forward them to the DoE
- ▶ Students with legitimate excuses due to actions or illness should be known to the DoE
- ▶ If there are any individual issues – please contact myself and the DoE in an email

Agreements

Introduction

The Introduction to the **Introduction to AI**

Artificial Intelligence

- ▶ Breaks down into four major definitions classes:
 - ▶ Thinking Humanly
 - ▶ Thinking Rationally
 - ▶ Acting Humanly
 - ▶ Acting Rationally
- ▶ Six major areas of concern (Russel & Norvig)
 - ▶ Natural Language Processing
 - ▶ Knowledge Representation
 - ▶ Automated Reasoning
 - ▶ Machine Learning
 - ▶ Computer Vision
 - ▶ Robotics

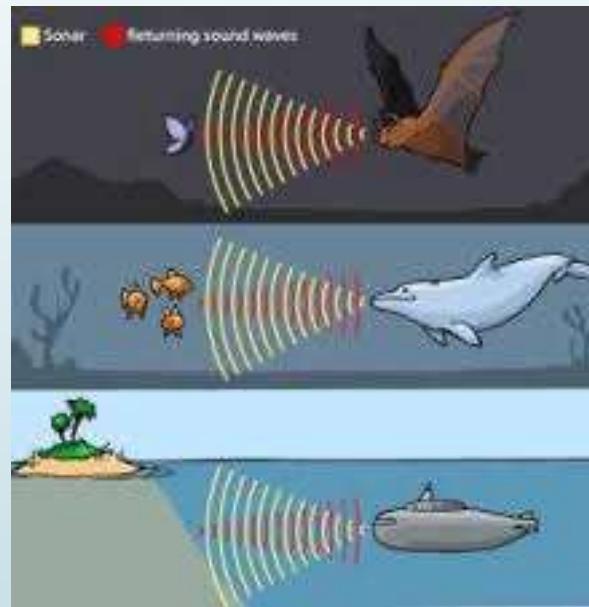


AI Onwards

► Computational Creativity



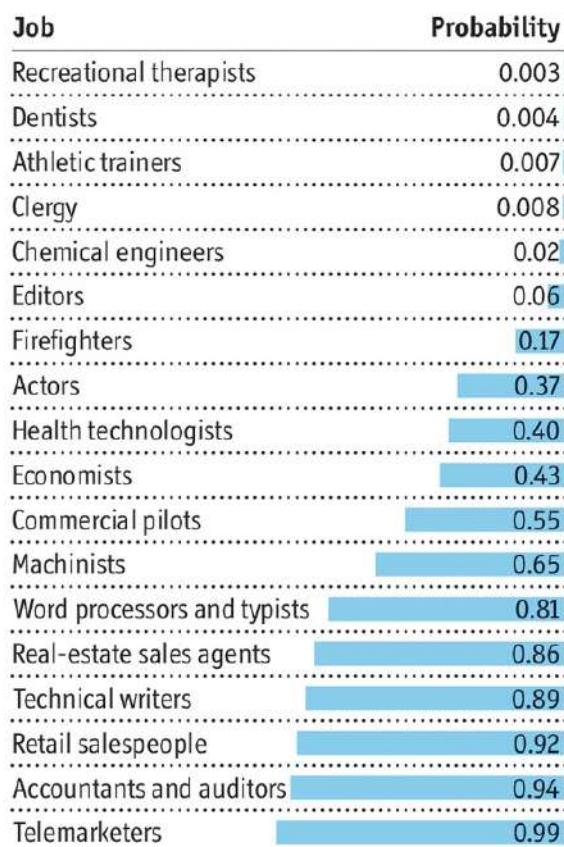
► Biomimicry



Why Should We Care?

Catalogue of fears

Probability of computerisation of different occupations, 2013
(1 = certain)



Source: "The Future of Employment: How Susceptible are Jobs to Computerisation?", by C. Frey and M. Osborne (2013)

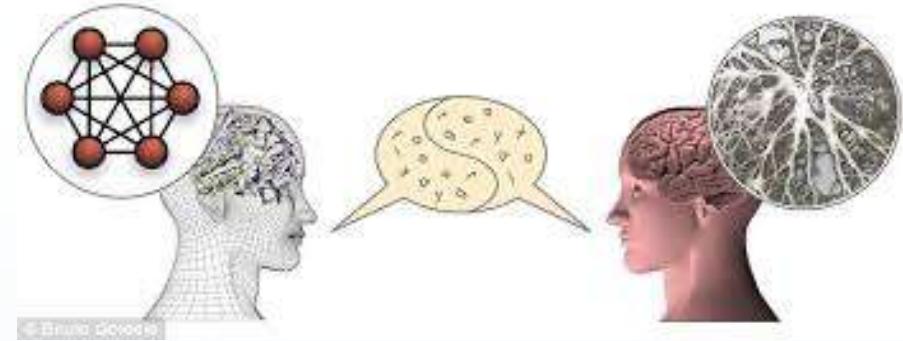
The future job situation:

- ▶ The Guardian, Sept. 13, 2016, "Robots will eliminate 6% of all US jobs by 2021, report says"
- ▶ BBC News, May 25, 2016, "Foxconn replaces '60,000 factory workers with robots"
- ▶ The Economist, Jun. 25, 2016, "The impact on jobs: Automation and anxiety"

Why Should We Care?

- ▶ Forbes Article - Automation, COVID, And The Future Of Work
 - ▶ Oct 2020 by Adi Gaskell
 - ▶ Questions the automation of jobs
 - ▶ Jobs in companies where AI has been engaged have actually INCREASED 15%
 - ▶ Firms productivity which engaged with AI was increased
 - ▶ This lead the firms to engage with new locations and move into new markets
 - ▶ Young people are the most vulnerable to employment issues
 - ▶ Most younger people are in jobs which are consumer facing
 - ▶ Retail, Services, etc.
 - ▶ These are also more gig work or temporary employment opportunities

Why Should We Care?



- ▶ Via the analysis of building something we learn more about it
- ▶ In order to understand our own process of thought
 - ▶ Build something which “thinks”
 - ▶ Explore what it means to be “rational”
 - ▶ What does it mean to be “intelligent”?
- ▶ Is there something beyond mechanical action to our brains
 - ▶ Free Will

Why Should We Care?



- ▶ Fear is defeated by knowledge
- ▶ If our future is going to be a terrifying world of the robots and AI
 - ▶ “I, for one, welcome our new overlords”
- ▶ The wars of the future will not be fought on the battlefield or at sea. They will be fought in space, or possibly on top of a very tall mountain. In either case, most of the actual fighting will be done by small robots. And as you go forth today remember always your duty is clear: To build and maintain those robots. - Simpsons

Intelligence is Situational



- ▶ Highly depends on cultural factors
- ▶ People of Indigenous tribes
 - ▶ Plants/Animals
 - ▶ Navigation of waters
- ▶ Methods of Transport
 - ▶ Horse
 - ▶ Do you know how to ride one?
 - ▶ Do you know how to feed, water, and resaddle one?
 - ▶ Do you know how to breed them?
 - ▶ Cars
 - ▶ Do you know how to drive?
 - ▶ Do you know how to refuel, put in washer fluid, do an oil change?
 - ▶ Do you know how an engine works enough to make repairs or build one?

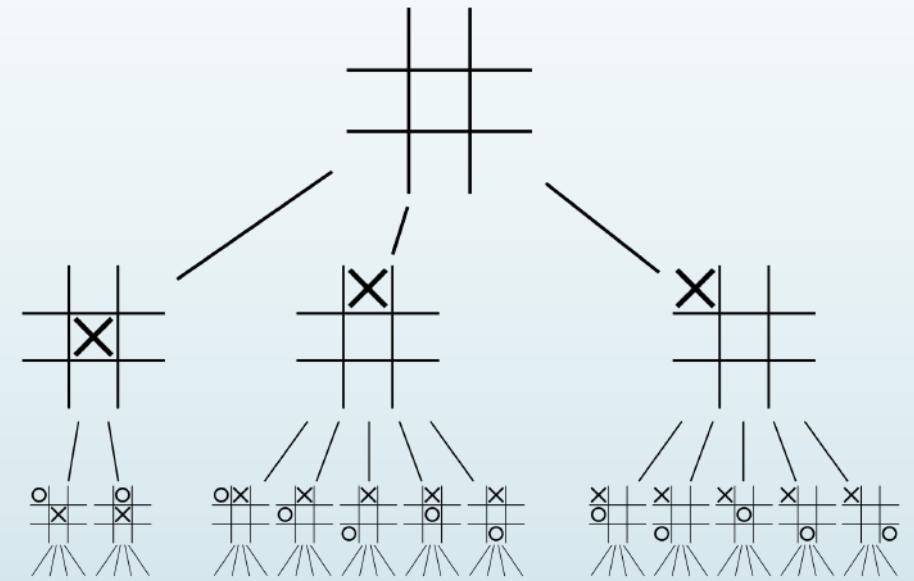
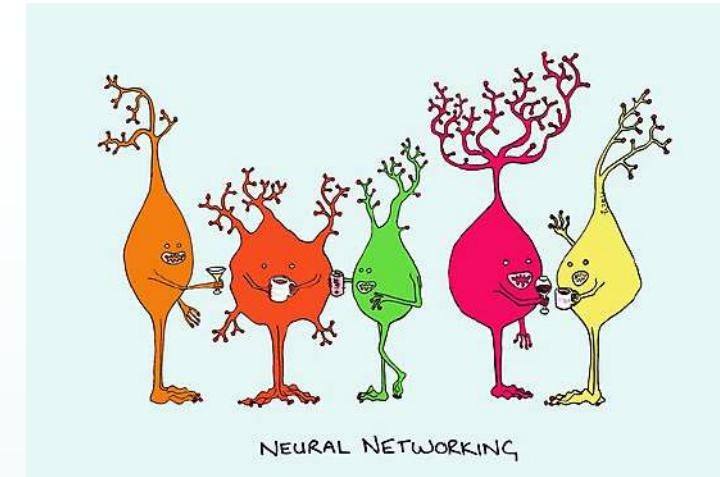
Self Driving Cars – Offloading Intelligence into Technology

- ▶ AI is taking the act of driving a car away from us
- ▶ Gradual Process
 - ▶ Powered Steering
 - ▶ Clutch assists
 - ▶ Automatic Transmissions
 - ▶ ABS
 - ▶ Four wheel skid prevention systems
 - ▶ Backup Cameras
 - ▶ Drive-by-wire
- ▶ What was once a skill required to live will become a domain-specific skill
 - ▶ Much like horses are to us now a skill which is reserved for those who are interested
 - ▶ Special teams of off-board drivers who will fix what cannot be done by AI now



Human Decision Making

- ▶ Decision Trees
- ▶ Game Trees
- ▶ Expert Systems
- ▶ Neural Networks



Bioinspired



- ▶ Humans are perhaps not only the one type of intelligence we mimic
- ▶ Swarms
 - ▶ Birds Flocking
 - ▶ Ant Colonies seeking food
 - ▶ Bees finding flowers
- ▶ Evolution and Breeding
 - ▶ Genetics
- ▶ Movement in Robotics



Embodied AIs

- ▶ A robot without a program is soulless
- ▶ An AI without a body is just a ghost
- ▶ AIs now have more physical (robot) or digital (agent) embodiment
 - ▶ Siri/Galaxy/Cortana/Alexa/Alisa – Digital Embodiment
 - ▶ Video Game Characters
- ▶ Humans love to make non-Human things have human characteristics



Greek Mythos

- ▶ Ovid's narrative poem Metamorphoses
 - ▶ Pygmalion (sculptor) fell in love with a statue he had carved
 - ▶ Statue comes to life due to inference of the gods (Aphrodite)
- ▶ Daedalus (inventor) put voice to a statue using quicksilver (i.e. mercury)
- ▶ Talos is an artificial man made of Bronze (Hephaestus)
- ▶ Pandora made of clay (Hephaestus at request of Zeus)



Strong & Weak AI

- ▶ Strong AI
 - ▶ Human-based design
 - ▶ Human-based constructions
- ▶ Weak AI
 - ▶ Problem Solving for specific tasks
 - ▶ May or may not have human aspects
 - ▶ Majority of AI work is on this issue

Welcome to the Course

- ▶ Think about what products you use every day will be obsolete due to AI in the near future
 - ▶ Will this be good for humanity or bad – make a list of pros and cons
 - ▶ What is Intelligence?



AI, Automation, 4th Industrial Revolution, Jobs



Darko Bozhinoski,
Ph.D. in Computer Science

About Me



Lived and worked in
6 countries

Started Ph.D.

Ph.D. in Computer Science

Gran Sasso Science Institute,
L'Aquila, Italy



Nov 2013



Dec 2017

Postdoc

IRIDIA, Artificial Intelligence,
ULB, Brussels, Belgium



Jan 2018

Researcher

Cognitive Robotics, TU
Delft, Delft, Netherlands



Jan 2019

Senior Researcher

FNRS, Belgium



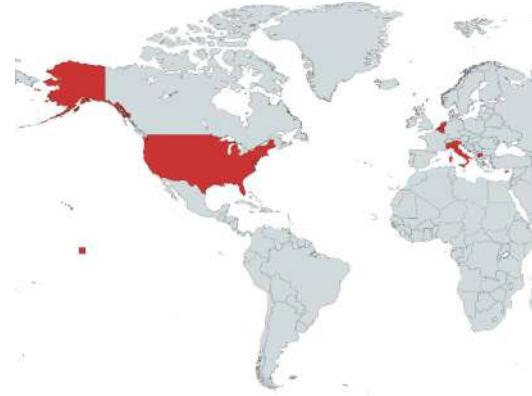
Oct 2021

Visiting Professor

InnoPolis, Russia



March 2022



Lived and worked in
6 countries

Research Focus

Autonomous systems

Model-driven engineering

Self-adaptive systems

Formal verification

Single and Multi-robot systems

Robot Swarms

Academic Partners

UNIVERSITÉ LIBRE DE BRUXELLES



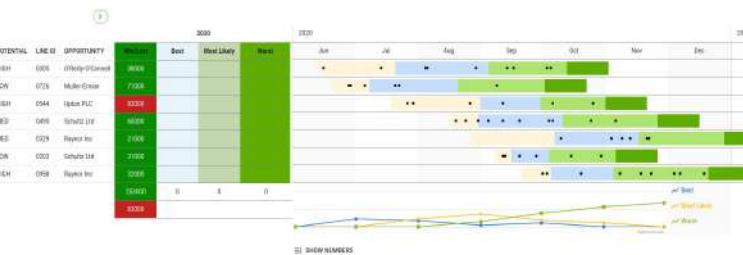
Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle



Industrial Projects

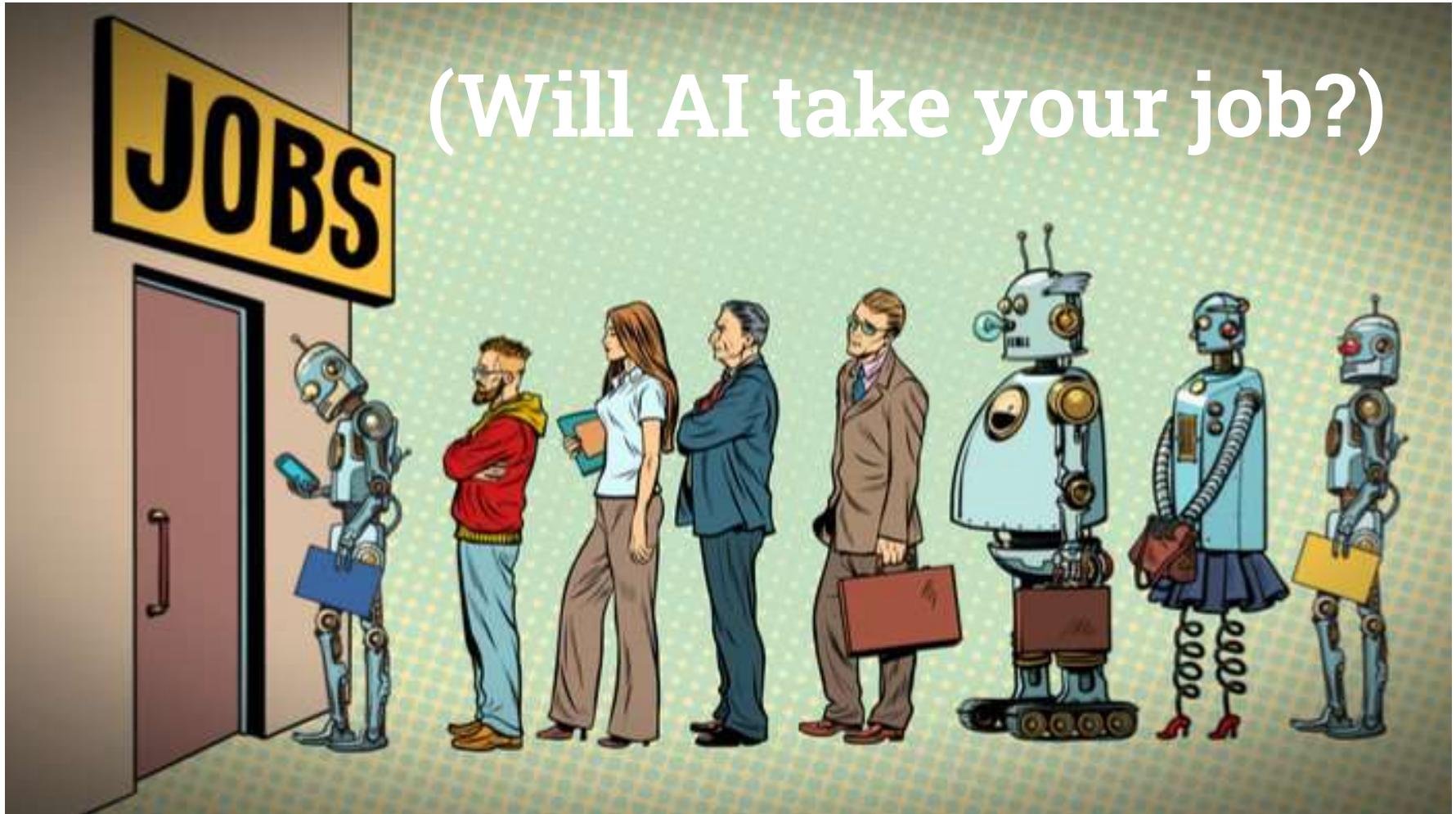
The screenshot shows the homepage of BFM.mk. At the top, there's a navigation bar with links for 'HOME', 'ARTICLES', 'VIDEOS', 'OPINION', 'INTERVIEWS', 'PHOTOS', 'TOP STORIES', 'SEARCH', and 'LOG IN'. Below the navigation is a large banner for 'GLAMIRA' featuring a woman's face. To the left, there's a sidebar with a 'COVID-19' section and a 'TOP NEWS' section. The main content area has several news articles with images and titles, such as 'Kazineti 226 возачи во Скопје' and 'БМРО-ДЛНИЕ побара јавна расправа за најавената реформа на системот на основното образование'. At the bottom, there's a footer with social media links for Facebook, YouTube, Instagram, and Twitter, along with a 'Tides' logo.

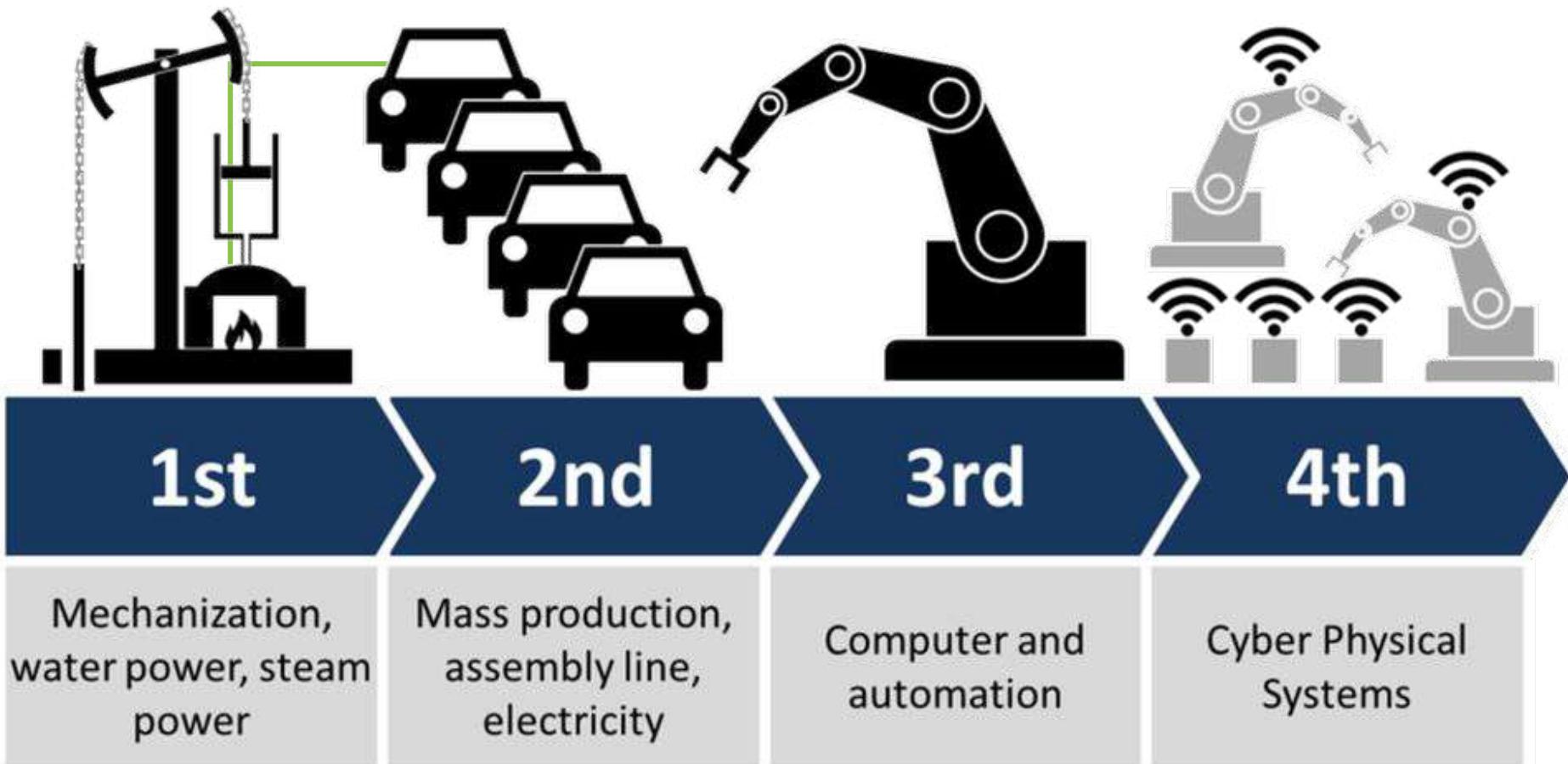
Collaboration with startups;



JOBS

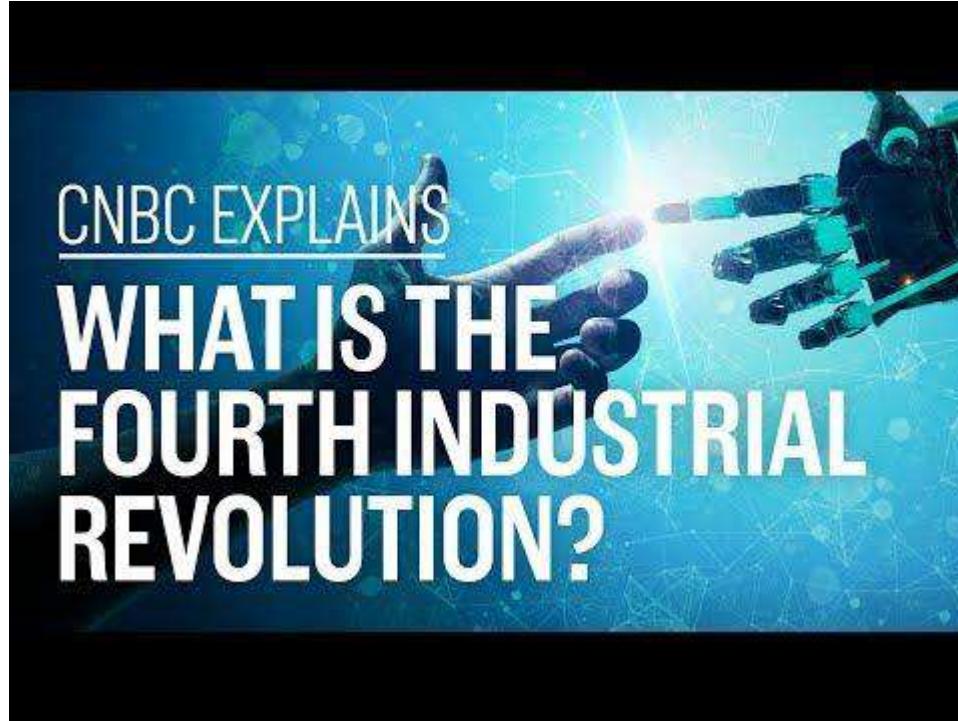
(Will AI take your job?)





CNBC EXPLAINS

WHAT IS THE FOURTH INDUSTRIAL REVOLUTION?



Consequences of the 4th Industrial Revolution

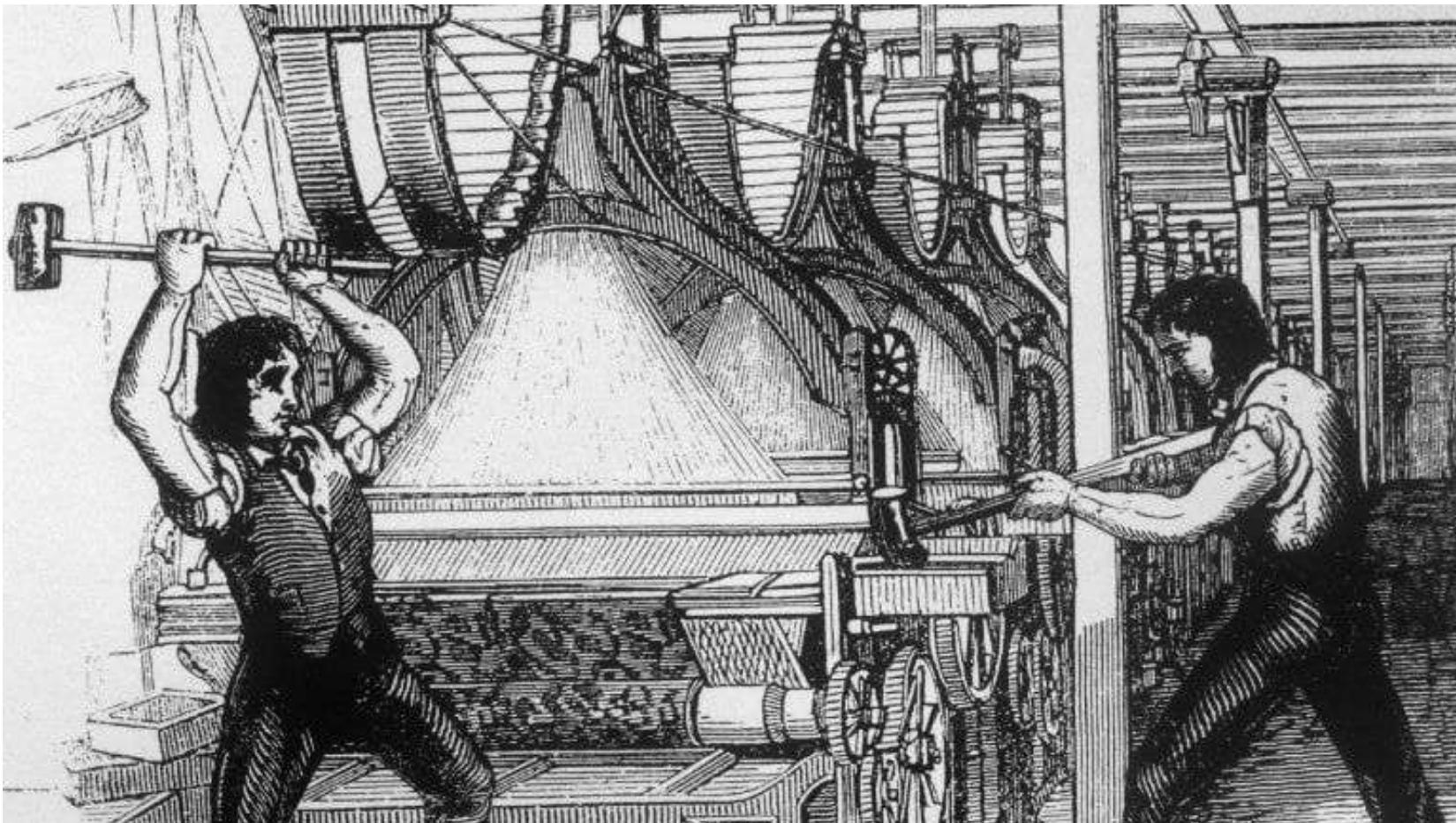
- ❖ Create bigger inequality
- ❖ Innovators, investors and shareholders benefit the most from the innovation
- ❖ AI will eliminate some jobs
- ❖ Create demand for new skills that many workers don't have
- ❖ Privacy concerns

Journalists and expert commentators tend to **overstate** the extent of machine substitution for human labor and ignore the strong complementarities between automation and labor that increase productivity, raise earnings, and augment demand for labor.

Why Are There Still So Many Jobs? The History and Future of
Workplace Automation
American Economic Association



**Have we been here
before?**



Luddite movement of the early 19th century, in which a group of English textile artisans protested the automation of textile production by seeking to destroy some of the machines.

In the 1960s that president Lyndon B Johnson created a special “Blue-Ribbon National Commission on Technology, Automation, and Economic Progress” to examine the consequences for jobs and living standards. The national commission eventually concluded that mass unemployment was not a risk.

“The basic fact is that technology eliminates jobs, not work”

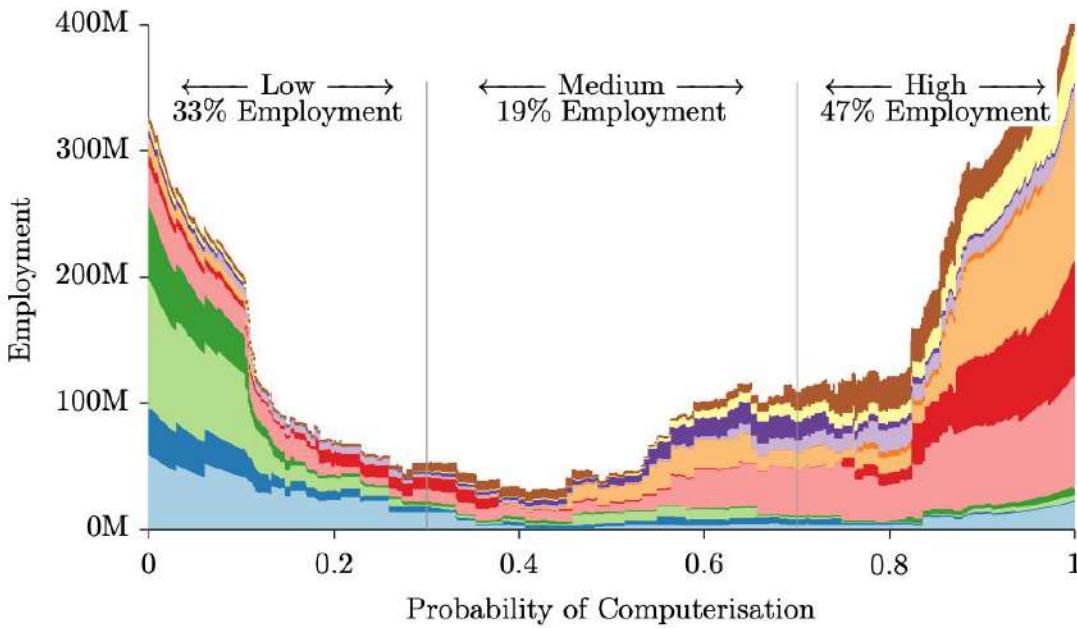


The past two centuries of Automation
and technological progress have not
made human labor obsolete.



Should we worry?

- Management, Business, and Financial
- Computer, Engineering, and Science
- Education, Legal, Community Service, Arts, and Media
- Healthcare Practitioners and Technical
- Service
- Sales and Related
- Office and Administrative Support
- Farming, Fishing, and Forestry
- Construction and Extraction
- Installation, Maintenance, and Repair
- Production
- Transportation and Material Moving



47 percent of U.S. jobs are at risk from automation

THE FUTURE OF EMPLOYMENT:
HOW SUSCEPTIBLE ARE JOBS TO
COMPUTERISATION?

Conceptualizing automation

Jobs are made up of many tasks and that while automation and computerization can substitute for some of them, understanding the interaction between technology and employment requires thinking about the range of tasks involved in jobs, and how human labor can often complement new technology (is it not a simple substitution.)

The task categorisation of Autor, et al. (2003) distinguishes between workplace tasks using a two-by-two matrix, with routine versus non-routine tasks on one axis, and manual versus cognitive tasks on the other.

Conceptualizing automation



Jobs are made up of many tasks and that while automation and computerization can substitute for some of them, understanding the interaction between technology and employment requires

Smarter together: Why artificial intelligence needs human-centered design



Conceptualizing automation

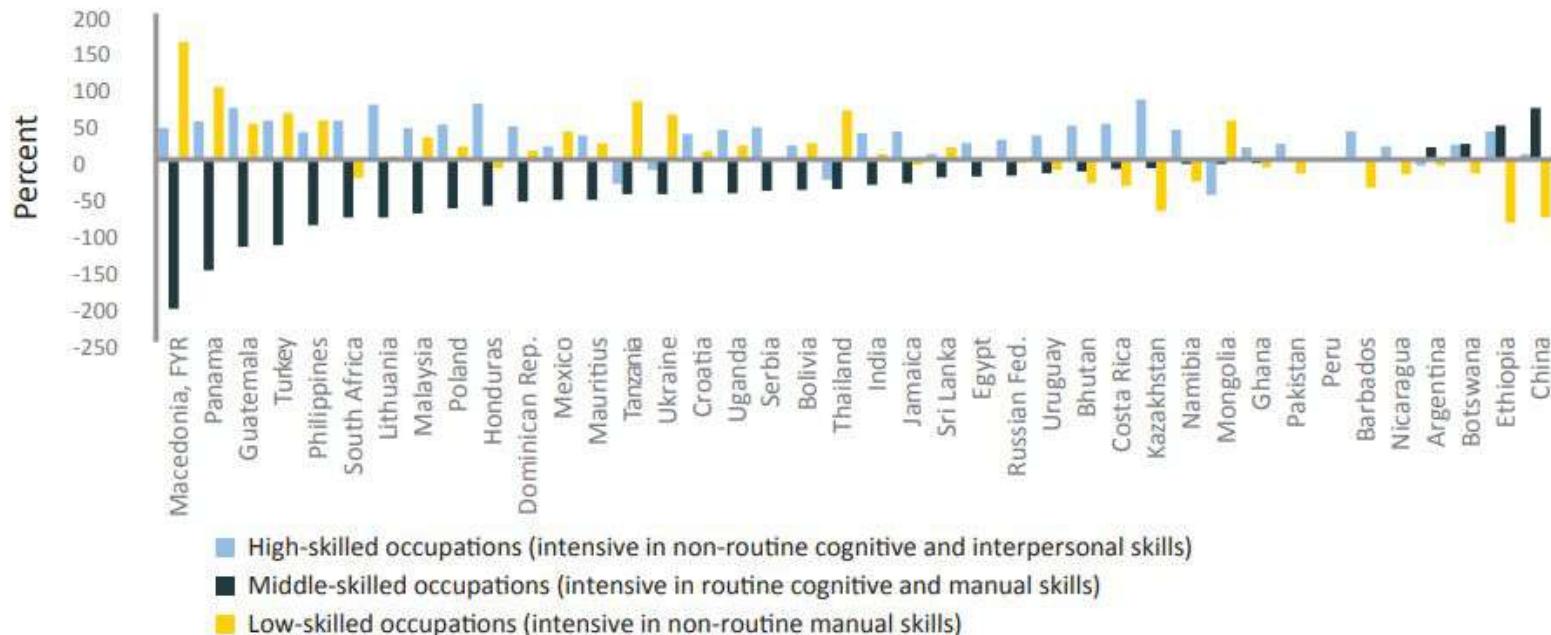
Automation has historically increased workers' productive and enabled humans to transition out of physically difficult, mundane, or menial labor, and in so doing, raised the standard of living.

Conceptualizing automation

Automation has historically increased workers' productive and enabled humans to transition out of physically difficult, mundane, or menial labor, and in so doing, raised the standard of living.

In the United States between 1900 and 2000, farming went from being the main employer in the economy, with 41 percent of all jobs, to employing only 2 percent of workers, according to data from the U.S. Department of Agriculture.

Figure 1. Annual Average Change in Employment Share, circa 1995–2012



Source: World Bank 2016.

FIGURE 22

Top 20 job roles in increasing and decreasing demand across industries

↗ Increasing demand

1	Data Analysts and Scientists
2	AI and Machine Learning Specialists
3	Big Data Specialists
4	Digital Marketing and Strategy Specialists
5	Process Automation Specialists
6	Business Development Professionals
7	Digital Transformation Specialists
8	Information Security Analysts
9	Software and Applications Developers
10	Internet of Things Specialists
11	Project Managers
12	Business Services and Administration Managers
13	Database and Network Professionals
14	Robotics Engineers
15	Strategic Advisors
16	Management and Organization Analysts
17	FinTech Engineers
18	Mechanics and Machinery Repairers
19	Organizational Development Specialists
20	Risk Management Specialists

↘ Decreasing demand

1	Data Entry Clerks
2	Administrative and Executive Secretaries
3	Accounting, Bookkeeping and Payroll Clerks
4	Accountants and Auditors
5	Assembly and Factory Workers
6	Business Services and Administration Managers
7	Client Information and Customer Service Workers
8	General and Operations Managers
9	Mechanics and Machinery Repairers
10	Material-Recording and Stock-Keeping Clerks
11	Financial Analysts
12	Postal Service Clerks
13	Sales Rep., Wholesale and Manuf., Tech. and Sci. Products
14	Relationship Managers
15	Bank Tellers and Related Clerks
16	Door-To-Door Sales, News and Street Vendors
17	Electronics and Telecoms Installers and Repairers
18	Human Resources Specialists
19	Training and Development Specialists
20	Construction Laborers

Source

Future of Jobs Survey 2020, World Economic Forum.

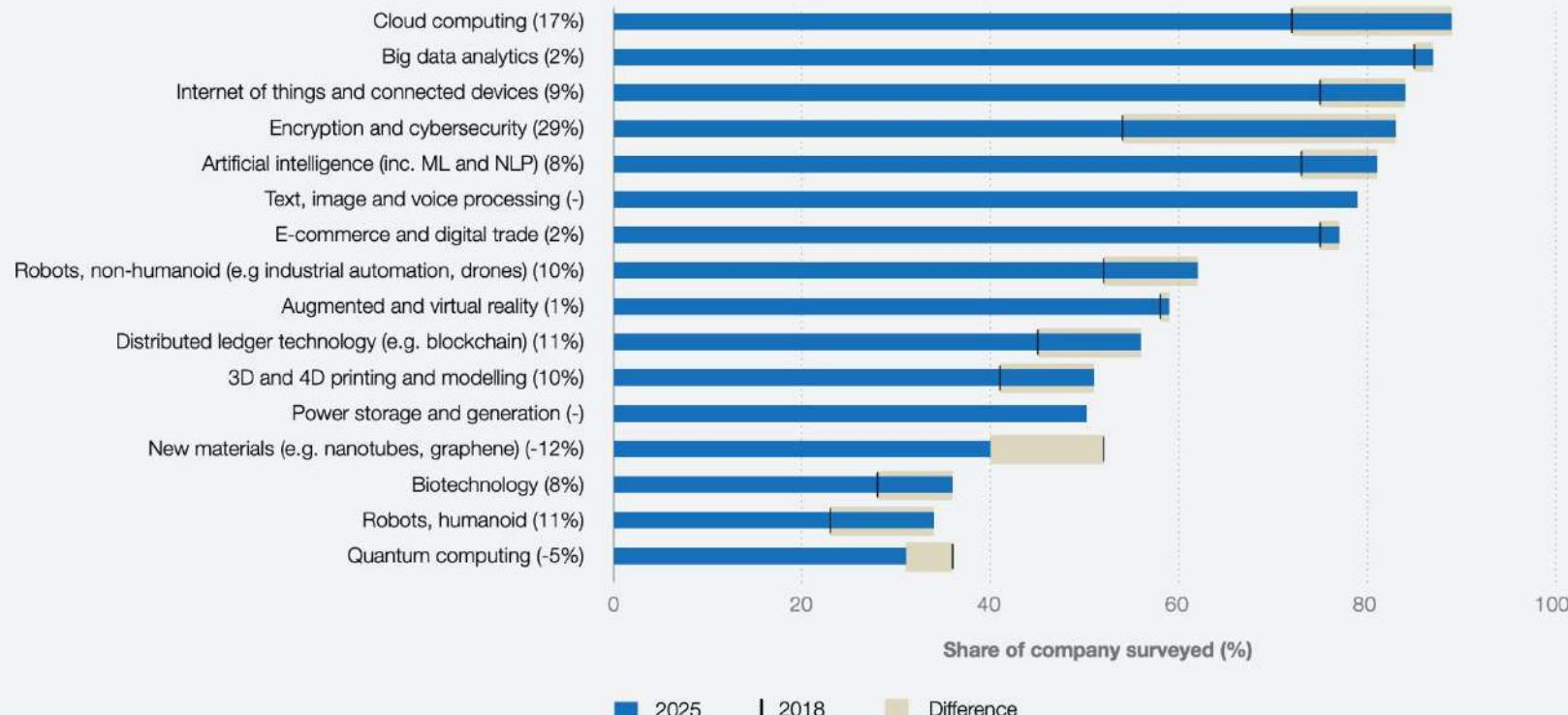
FIGURE 23

Emerging roles clustered into the jobs of tomorrow



FIGURE 18

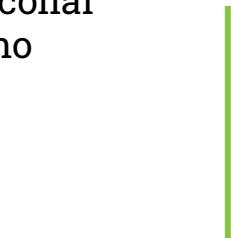
Technologies likely to be adopted by 2025 (by share of companies surveyed)





Who is affected?

In the recent past, those hardest hit were men with low levels of education who dominated manufacturing and other blue-collar jobs, and women with intermediate levels of education who dominated clerical and administrative positions.



<https://willrobotstakemyjob.com/>

TABLE I. O*NET variables that serve as indicators of bottlenecks to computerisation.

Computerisation bottleneck	O*NET Variable	O*NET Description
Perception and Manipulation	Finger Dexterity	The ability to make precisely coordinated movements of the fingers of one or both hands to grasp, manipulate, or assemble very small objects.
	Manual Dexterity	The ability to quickly move your hand, your hand together with your arm, or your two hands to grasp, manipulate, or assemble objects.
	Cramped Work Space, Awkward Positions	How often does this job require working in cramped work spaces that requires getting into awkward positions?
Creative Intelligence	Originality	The ability to come up with unusual or clever ideas about a given topic or situation, or to develop creative ways to solve a problem.
	Fine Arts	Knowledge of theory and techniques required to compose, produce, and perform works of music, dance, visual arts, drama, and sculpture.
Social Intelligence	Social Perceptiveness	Being aware of others' reactions and understanding why they react as they do.
	Negotiation	Bringing others together and trying to reconcile differences.
	Persuasion	Persuading others to change their minds or behavior.
	Assisting and Caring for Others	Providing personal assistance, medical attention, emotional support, or other personal care to others such as coworkers, customers, or patients.

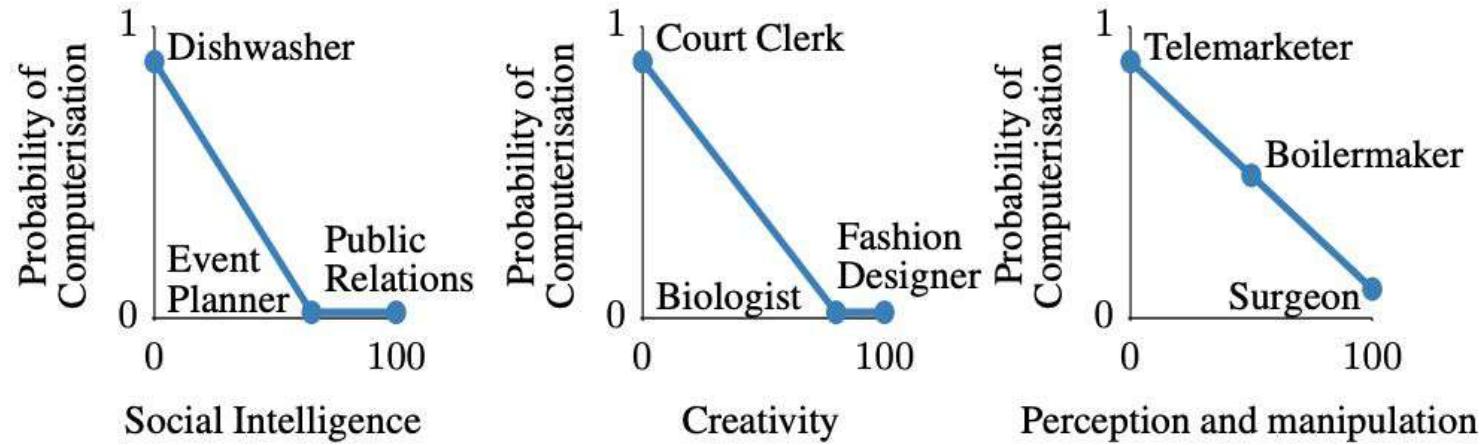


FIGURE I. A sketch of how the probability of computerisation might vary as a function of bottleneck variables.

What can we say about the current technologies?

While in the past, technology was increasingly able to perform routine manual and cognitive tasks, in the current digital and computing revolution, machines can also perform some nonroutine tasks that had been reserved to humans.

AI is remarkably effective in conducting specific tasks rather than replicating human intelligence

The replacement of labor by machines takes time and depends on circumstances specific to a given context. Technological innovations tend to occur in developed countries, and their adoption in developing countries usually occurs with a time lag.

Man Automated His Job One Year Ago And Is Still Earning \$90,000



Dominic Smithers



Published 13:07, 13 January 2022 at GMT
| Last updated 13:07, 13 January 2022 at GMT



The [pandemic](#) forced millions of people to change their working habits, moving from the office to the bedroom and only seeing colleagues over Zoom.

But while many of us struggled to adapt to the 'new normal', one man managed to make it work to his advantage.

An [IT specialist](#) recently revealed that he was able to make his \$90,000 (£65,603) a year job fully automated.



How to mitigate the negative effects of technological change?



How to mitigate the negative effects of technological change?



How to maximize the potential social
gains from technological change?

Government perspective

- ❖ Policies that will facilitate labor market flexibility and mobility
- ❖ Policies that introduce and strengthen safety nets and social protection
- ❖ Improved education and training process.
 - Complementing fundamental education with active labor market policies,
 - workforce training
 - lifelong learning opportunities that engage workers to participate in the changing labor market

Company Perspective

- ❖ Effective transformation of the production process
- ❖ Catching up with latest technologies
- ❖ Strategies for retraining the company stuff

Amazon will retrain one-third of its U.S. employees to get ahead of tech changes

\$700 million initiative that will cover 100,000 workers by 2025.

Beth Galetti, Amazon's senior vice president for HR. "We think it's important to invest in our employees, and to help them gain new skills and create more professional options for themselves."

Business

Amazon will retrain one-third of its U.S. employees to get ahead of tech changes

Individual Perspective



Change is the only constant.

Heraclitus

REFERENCES

1. Why Are There Still So Many Jobs? The History and Future of Workplace Automation
2. The future of employment: How susceptible are jobs to computerisation?
3. The Future of Work: Race with—not against—the Machine



Introduction to Artificial Intelligence

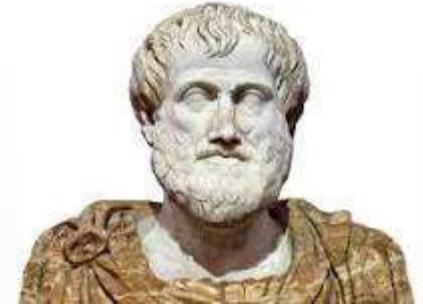
Week 2



2

What is Intelligence?

Philosophy



- ▶ Epistemology – fr. Greek - from Greek ἐπιστήμη, epistēmē, meaning "knowledge", and λόγος, logos, meaning "logical discourse"
- ▶ Study of how we know what we know
- ▶ Many epistemologists see a separation of Truth/Belief/Justification that gives Knowledge
 - ▶ Truth is the objective reality
 - ▶ Belief is the subjective idea of what reality is
 - ▶ Justification is an explanation
 - ▶ Knowledge is True Belief with Justification (comes from Socrates)
 - ▶ Example: I had a coffee this morning to keep myself awake for class

Gettier Cases

- ▶ Smith and Jones apply for a job
- ▶ Smith asks Jones for some change and knows he has 10 Rouble coins in his pocket
- ▶ Smith is informed that it is Jones who will be given the job
 - ▶ Jones has ten coins in his pocket, Jones will get the job
 - ▶ Smith now believes that the man with 10 coins in his pocket will get the job
- ▶ Assume there is a change of plan and Smith is selected; also unknown to Smith to him was that in his pocket are 10 coins
- ▶ So, the above propositions where True, were beliefs, were justified
- ▶ But Smith didn't KNOW!
 - ▶ Includes a justified false beliefs
 - ▶ A Justification must also be necessarily the cause

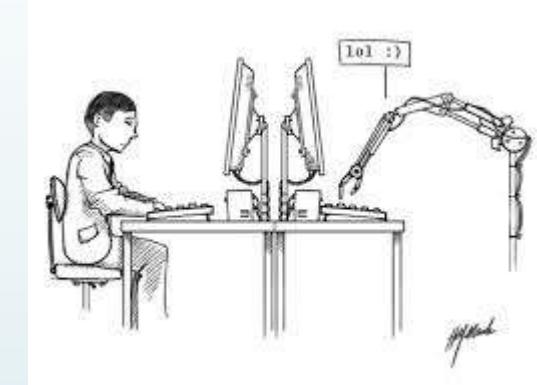
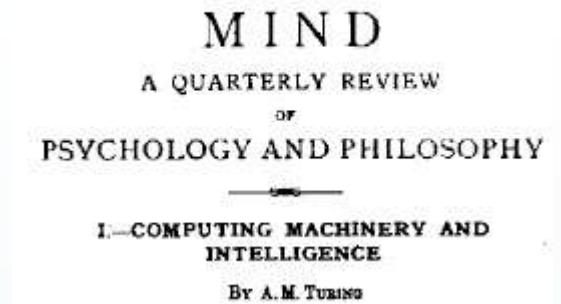
Alan Mathison Turing OBE FRS

- ▶ Born 23 June 1912 to an English civil servant for India
- ▶ Studied mathematics at King's College, Cambridge, and was elected a fellow
- ▶ Developed the Universal Turing Machine as part of the Church-Turing Thesis
- ▶ PhD from Princeton; offered a postdoctoral position by John von Neumann but instead returns to England to Cambridge
- ▶ Multiple Philosophical fights with Ludwig Wittgenstein
- ▶ Went on to work at Bletchley Park
 - ▶ Developed the Bombe decoder
 - ▶ *The Applications of Probability to Cryptography and Paper on Statistics of Repetitions* not released by UK Govt. for 70 years
- ▶ Developed and popularized the Turing Test
- ▶ Went on to studies of new computers, biology, and mathematics
- ▶ He was prosecuted in 1952, public government apology 2009, full pardon in 2013
- ▶ Died 1954 of cyanide poisoning (Apple) – most likely at his own hand



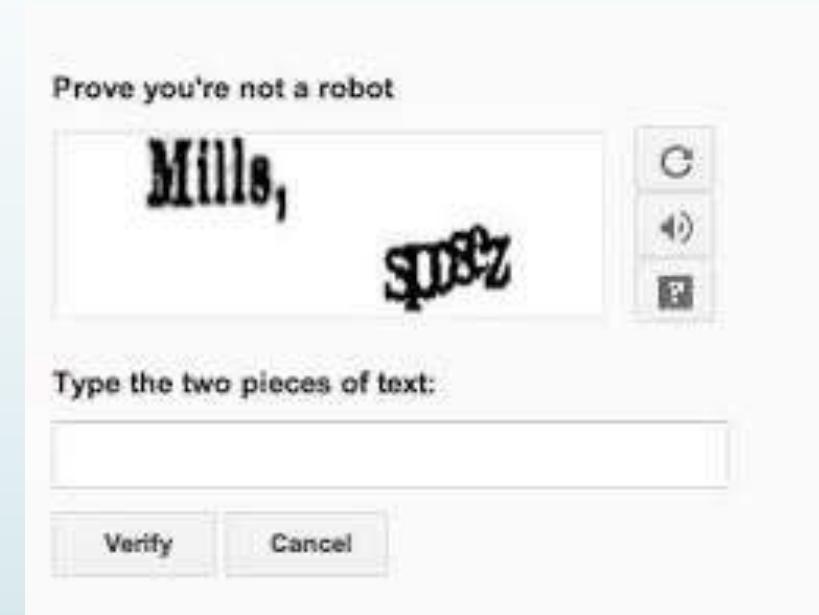
Turing Test

- ▶ Published in 1952
- ▶ Interrogator
- ▶ Human v. Computer
- ▶ Can the Interrogator detect the computer
- ▶ Solved in some limited areas
- ▶ Inverse of this test used as a human detector



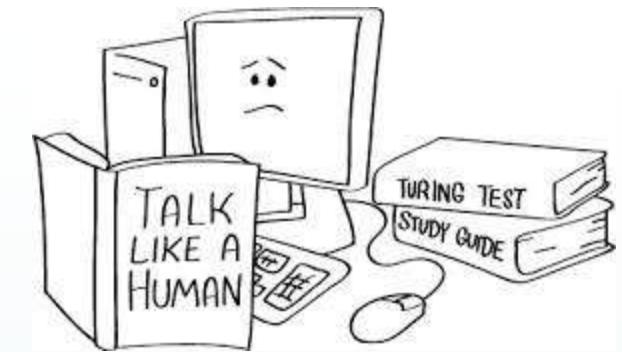
CAPTCHA aka completely automated public Turing test to tell computers and humans apart

- ▶ Utilized in a number of systems
 - ▶ Email
 - ▶ Online ordering
 - ▶ Account creation
- ▶ Cost effective
- ▶ Has been utilized in order to decode optical character recognition (OCR) issues



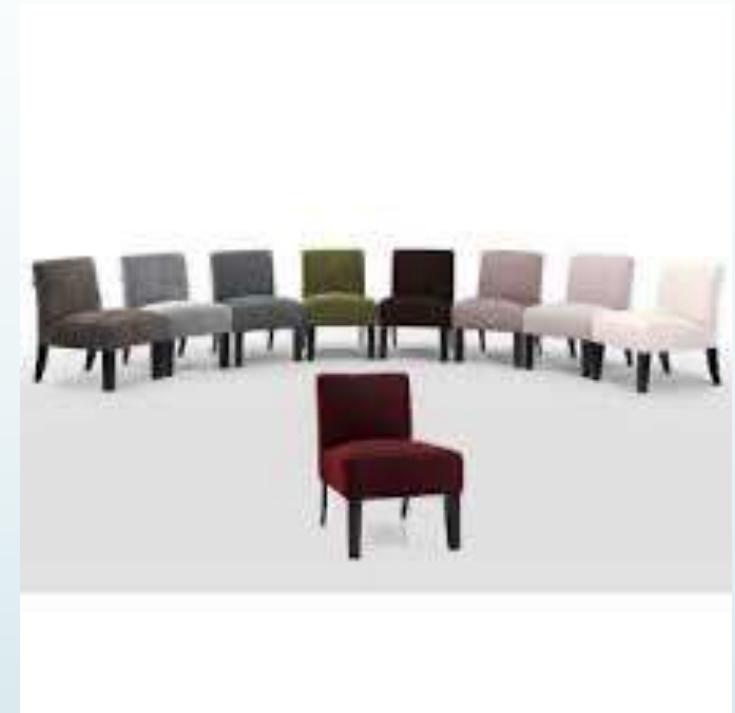
Computers don't think like us?

- ▶ Image Recognition
 - ▶ Segmentation
 - ▶ Detection
 - ▶ Conceptualization
 - ▶ Verbalization

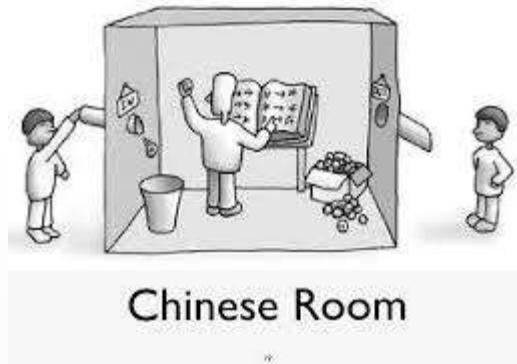


Syntax v. Semantics

- Chair has a meaning conceptually and as an object



Chinese Room Argument



- ▶ Thought experiment
- ▶ Imagine yourself as someone who has no knowledge of the language of Chinese
 - ▶ Rather easy for some of us – but feel free to substitute any language you don't know
- ▶ You are locked in the room with a large book of instructions, a pile of paper, and two slots labeled in and out
- ▶ From the in slot comes a page of Chinese characters, the instructions tell you what to do and to write before putting it on the out slot

Does This Mean That You KNOW Chinese?

- ▶ You don't know what the characters mean – semantics
- ▶ You know actions to be produced on symbols – syntax
- ▶ You will be slow to begin with, but as you learn the rule system you become better and better at your actions
- ▶ This is how a computer makes actions but is it all it does?

Arguments Against the Room

- ▶ You are not processing meaningless symbols
 - ▶ They have a meaning internally and a different external meaning
- ▶ This might be only part of the bigger system
 - ▶ Many Chinese rooms
 - ▶ Who writes the rule book?
 - ▶ Can the rules be changed?
 - ▶ Development of rules is now part of many AI



13

First-order and Propositional Logic

Why Do We Care?

- ▶ Utilized in a number of AI systems
- ▶ Allows us to have a basic idea of ‘knowledge’
- ▶ Categories
- ▶ Binary predicates
 - ▶ Part of the set/Not part of the set

Operations

- ▶ For Every \forall
 - ▶ For Every X its value is 3; $\forall x=3$
- ▶ There exists \exists
 - ▶ There is an X which is 3; $\exists x=3$
- ▶ Conjunction \wedge
 - ▶ X and Y; $x \wedge y$
- ▶ Disjunction \vee
 - ▶ X or Y; $x \vee y$
- ▶ Implication \rightarrow
 - ▶ If x then y; $x \rightarrow y$
- ▶ Biconditional \leftrightarrow
 - ▶ X if and only Y; $x \leftrightarrow y$
- ▶ Such that (s.t.) or :

Example Propositions

- ▶ $\text{REDHAIR}(X)$ – X has red hair
- ▶ $\text{EVIL}(X)$ – X is evil
- ▶ $\text{GLASSES}(X)$ – X wears glasses
- ▶ $\text{FEMALE}(X)$ – X is female

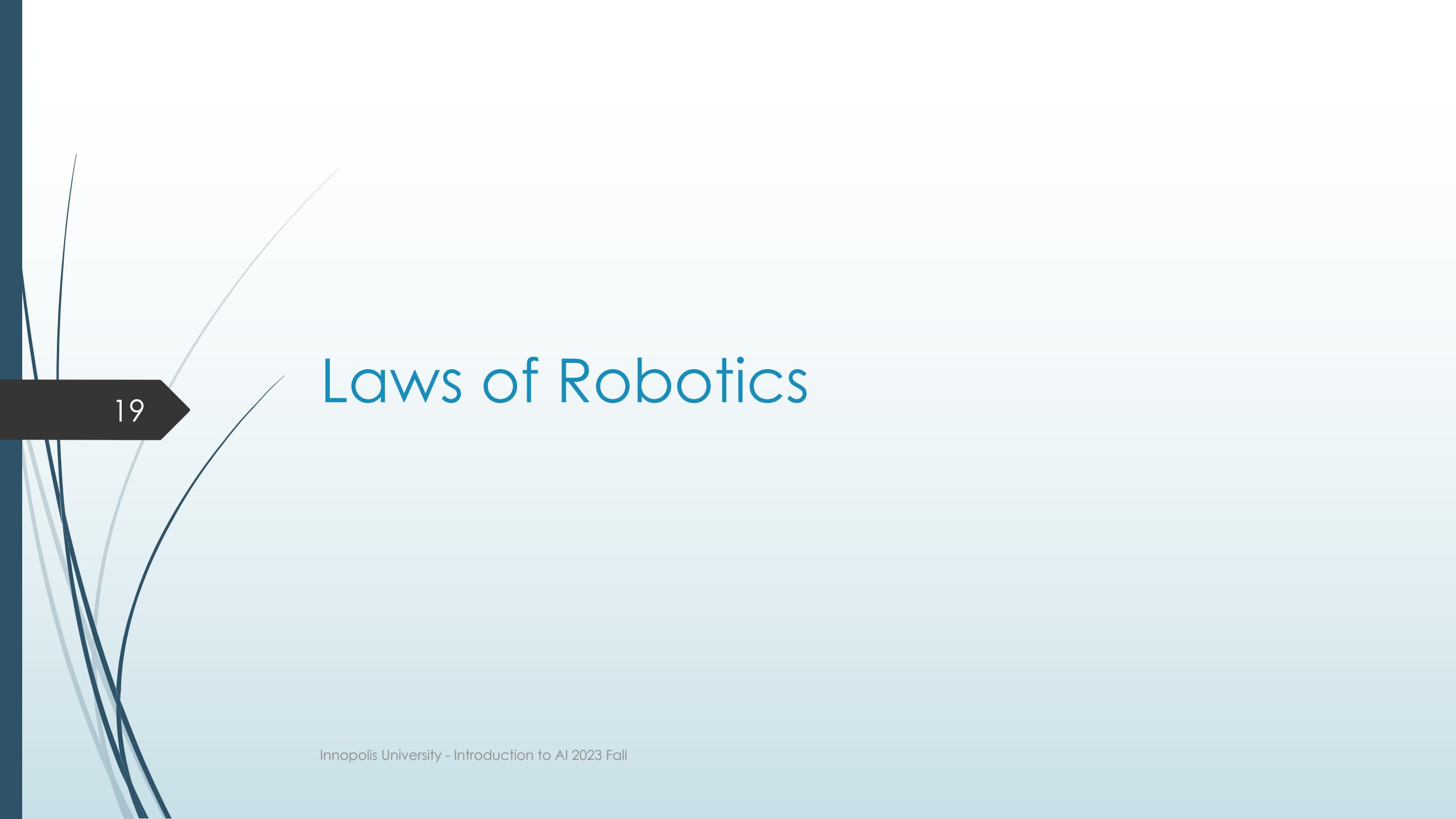
Example

- ▶ All female redheads are evil
- ▶ $\forall X, \text{REDHAIR}(X) \wedge \text{FEMALE}(X) \rightarrow \text{EVIL}(X)$

- ▶ $\forall X \text{ FEMALE}(X) \rightarrow \exists X \text{ REDHAIR}(X) \vee \text{EVIL}(X)$
- ▶ For all females there exists one who has red hair or who is evil

More...

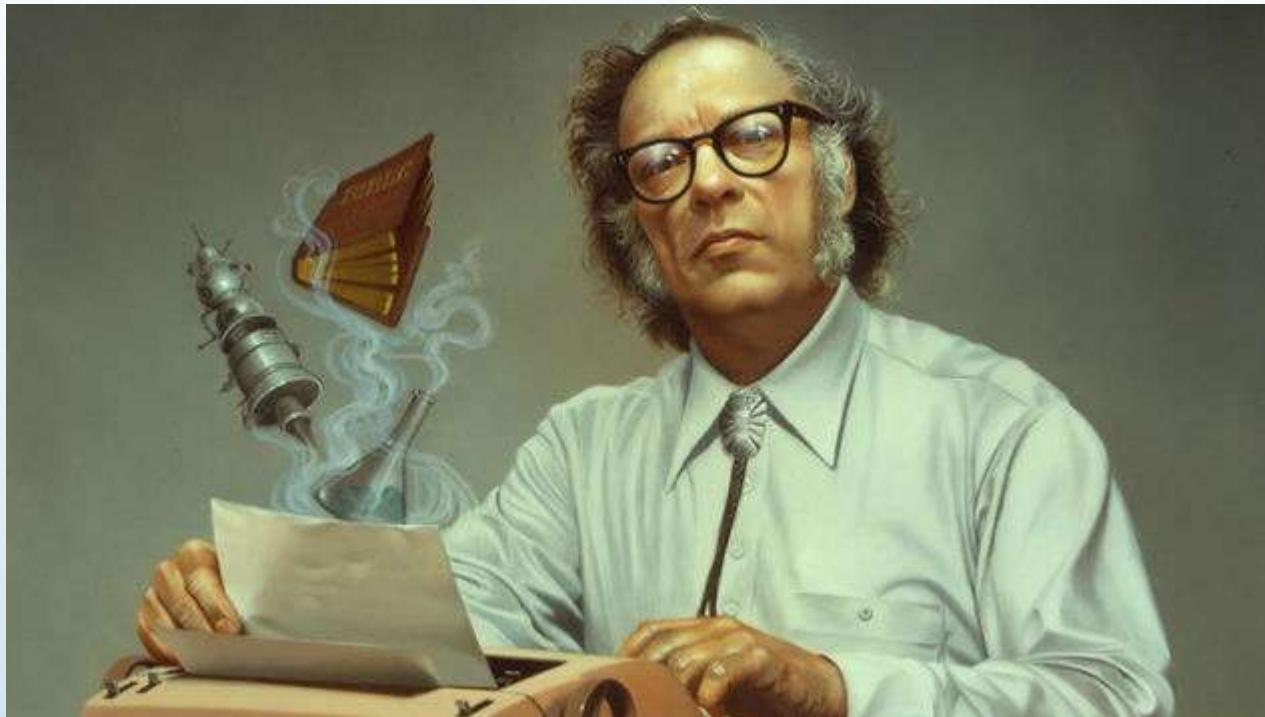
- ▶ This week tutorial and lab will contain more information about first-order and propositional logic



19

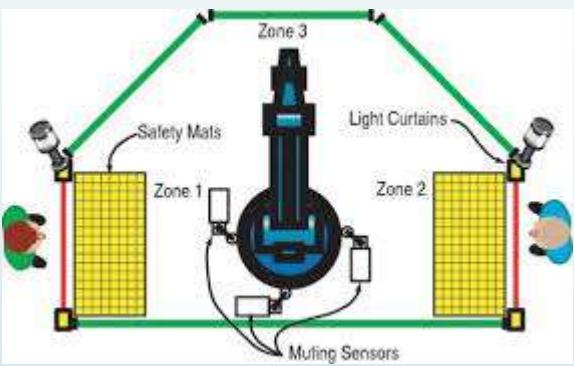
Laws of Robotics

Isaac Asimov



“A robot may not injure a human being or, through inaction, allow a human being to come to harm.”

LAW ONE – DO NO HARM



“A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.”

LAW TWO – OBEY HUMANS



“A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.”

LAW THREE - SELF PRESERVATION



Robots that Kill

- ▶ Where does this place military and police drones
 - ▶ Police/Military are required to use force which is reasonable at the time for the action (police bound by constitutional laws and militaries by international law)
 - ▶ Dallas police utilized a robot with a grenade strapped on it to subdue a suspect who shot at officers
 - ▶ Numerous militaries with semi autonomous drones with missiles

Zeroth Law

- ▶ A robot may not injure HUMANITY, or, through inaction, allow HUMANITY to come to harm
- ▶ What does this say for ethical calculus?

Logic

**Darko Bozhinoski,
Ph.D. in Computer Science**

Lecture Objectives

- ❖ Propositional Logic
- ❖ First-Order Logic

PROPOSITIONAL LOGIC

- Sentences considered in propositional logic are not arbitrary sentences but are the ones that are either true or false, but not both. This kind of sentences are called propositions. By reading a sentence and basing your conclusion directly from the facts mentioned in the sentence only, you can conclude the truth or falsity of the statement.

Examples ?? (Discussion)

Examples

- ❖ Grass is green
- ❖ $2 + 5 = 5$
- ❖ $x + 3 = 3$
- ❖ $x + 3 > 5$
- ❖ John is in the classroom.
- ❖ Mary is enrolled in 270A.

- A statement is a sentence that is either true or false, but not both.
- Statements:
 - It is raining.
 - I am carrying an umbrella.
- Non-statements:
 - a. (Question): Why are you late?
 - b. (Command): Open the door.
 - c. (Wish): If only I had studied a little harder...
 - d. (Something Vague): $x + y = 4$.

COMPOUND STATEMENTS

A **compound statement** is one with two or more simple statements as parts - **components**. A component of a compound is any whole statement that is part of a larger statement.

Simple sentences which are true or false are basic propositions.

Complex sentences are constructed from basic propositions by combining them with **connectives**.

- **propositions** and **connectives** are the basic elements of propositional logic. **Five basic connectives** here:

NOT	\neg	or \sim
AND	\wedge	
OR	\vee	
IF THEN	\rightarrow	
IF \neg and ONLY IF	\leftrightarrow	

Logical Operators

- Binary operators
 - Conjunction – “and”.
 - Disjunction – “or”.
- Unary operator
 - Negation – “not”.
- Other operators
 - XOR – “exclusive or”
 - NAND – “not both”
 - NOR – “neither”

Elements of Propositional Logic

Simple sentences.

A **simple statement** is one that does not contain any other statement as a part. We will use the lower-case letters, p , q , r , ..., as symbols for simple statements.

If p or q , then r

Therefore, if not r , then not p and not q .

E.g. If Jane is a math major or Jane is a computer science major, then Jane will take Math 150.

Jane is a computer science major.

Therefore, Jane will take Math 150.

Examples

- P means “It is hot.”
- Q means “It is humid.”
- R means “It is raining.”
- $(P \wedge Q) \rightarrow R$
“If it is hot and humid, then it is raining”
- $Q \rightarrow P$
“If it is humid, then it is hot”
- A better way:

Hot = “It is hot”

Humid = “It is humid”

Raining = “It is raining”

Translation into Propositional Logic

- If it rains, then the game will be cancelled.
- If the game is cancelled, then we clean house.
- Can we conclude?
 - If it rains, then we clean house.
- $p = \text{it rains}$, $q = \text{game cancelled}$ $r = \text{we clean house}$.
- If p then q . $\neg p \text{ or } q$
- If q then r . $\neg q \text{ or } r$
- if p then r . $\neg p \text{ or } r$ (resolution)

Translation into Propositional Logic

- ❖ Basic statements

- p = “It is raining.”
 - q = “I am carrying an umbrella.”

- ❖ Compound statements

- $p \wedge q$ = “It is raining and I am carrying an umbrella.”
 - $p \vee q$ = “ It is raining or I am carrying an umbrella.”
 - $\sim p$ = “It is not raining.”

TRUTH VALUES

Each simple statement comprising a compound statement is a statement in its own right, that is it has a truth value, that is it can be true or false but not both.

The truth values of each individual statements affect the truth value of the compound statement of which they are part of.

NEGATION

If p is a statement variable, the negation of p is “not p ” or “It is not the case that p ” and is denoted as $\sim p$. It has opposite truth values as p . Wherever p is true, $\sim p$ is false and vice versa. The truth values for negation summarized in a truth table are

p	$\sim p$
T	F
F	T

LOGICAL AND

If p and q are statement variables, the conjunction of p and q is “ p and q ” denoted $p \wedge q$. It is true when, and only when, both p and q are true. If either p or q is false, or if both are false, $p \wedge q$ is false. The truth values for and summarized in a truth table are

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

LOGICAL OR

If p and q are statement variables, the disjunction of p and q is “ p or q ” denoted $p \vee q$. It is true when at least one of p or q is true and is false only when both p and q are false. The truth values for p or q summarized in a truth table are

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

TRUTH TABLES

The general rule for constructing truth tables is:

1. List all the variables mentioned in the statement in separate columns and then list all the combinations of truth values for them.
2. Add columns for operations mentioned in the statement with all the combinations of truth values for them starting from inside out the parenthesis if there exist any.
3. Evaluate the final result

Exercise. Construct truth table for:

- p OR (p AND q)

TRUTH TABLES

The general rule for constructing truth tables is:

1. List all the variables mentioned in the statement in separate columns and then list all the combinations of truth values for them.
2. Add columns for operations mentioned in the statement with all the combinations of truth values for them starting from inside out the parenthesis if there exist any.
3. Evaluate the final result

Exercise. Construct truth table for:

- p OR (p AND q)

Result: <https://web.stanford.edu/class/cs103/tools/truth-table-tool/>

Logical Equivalence

- ❖ Two statements are said to be **logically equivalent** if and only if, they've identical truth values for each possible substitution of statements for their statement variables.
- ❖ It is denoted as $P \equiv Q$

Follow this procedure:

- ❖ Construct the truth table for P
- ❖ Construct the truth table for Q using the same statement variables for identical component statements.
- ❖ Check each combination of truth values of the statement variables:
 - a. If in each row the truth value of P is the same as the truth value of Q, then P and Q are logically equivalent.
 - b. Otherwise, P and Q are not logically equivalent.

Are $(p \wedge q) \vee (\sim p \wedge \sim q)$ and $(p \vee \sim q) \wedge (\sim p \vee q)$ logically equivalent?

$$(p \wedge q) \vee (\sim p \wedge \sim q) \equiv (p \vee \sim q) \wedge (\sim p \vee q)$$

p	q	$(p \wedge q) \vee (\sim p \wedge \sim q)$	$(p \vee \sim q) \wedge (\sim p \vee q)$
T	T	T	T
T	F	F	F
F	T	F	F
F	F	T	T

De Morgan's law

- $\neg(p \wedge q) \equiv \neg p \vee \neg q$ *the negation of and is logically equivalent to the or of the negated components*
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$ *the negation of or is logically equivalent to the and of the negated components*

Tautology and Contradiction

A **tautology** is a logical expression that is true regardless of the values of its variables. Thus, for a tautology, the value of the expression is TRUE in every row of the truth table. The column of a tautology in a truth table contains only T's.

Example??

A proposition which is always false is called a **contradiction**. The column of a contradiction in a truth table contains only F's.

Example??

Tautology and Contradiction

A **tautology** is a logical expression that is true regardless of the values of its variables. Thus, for a tautology, the value of the expression is TRUE in every row of the truth table. The column of a tautology in a truth table contains only T's.

Example: $P \vee (\text{NOT } P)$

A proposition which is always false is called a **contradiction**. The column of a contradiction in a truth table contains only F's.

Example: $P \wedge (\text{NOT } P)$

Conditional statements

If p and q are statement variables, the **conditional** of q by p is “if p then q ” or “ p implies q ” and is denoted by $p \rightarrow q$.

It is false when p is true and q is false.

A conditional statement is **vacuously true** or **true by default** if its hypothesis is false.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	F	T
F	T	T

$$p \rightarrow q \equiv \neg p \vee q$$

“If p then q ” is logically equivalent to “not p or q ”.

The **negation** of a conditional statement. The negation of “if p then q ” is logically equivalent to “ p and not q ”.

$$\neg(p \rightarrow q) \equiv p \wedge \neg q$$

The proposition $\neg q \rightarrow \neg p$ is called the **contrapositive** of the proposition $p \rightarrow q$. They are logically equivalent. $p \rightarrow q \equiv \neg q \rightarrow \neg p$

Symbolically the contrapositive of

A **conditional statement is logically equivalent to its contrapositive**.

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of \wedge
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of \vee
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of \wedge
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of \vee
$\neg(\neg \alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$	De Morgan
$\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of \wedge over \vee
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of \vee over \wedge

Figure 7.11 Standard logical equivalences. The symbols α , β , and γ stand for arbitrary sentences of propositional logic.

Propositional logic (Recap)

- **Logical constants:** true, false
- **Propositional symbols:** P, Q, S, ... (**atomic sentences**)
- Wrapping **parentheses**: (...)
- Sentences are combined by **connectives**:

\wedge ...and [conjunction]

\vee ...or [disjunction]

\Rightarrow ...implies [implication / conditional]

\Leftrightarrow ..is equivalent [biconditional]

\neg ...not [negation]

- **Literal:** atomic sentence or negated atomic sentence

Propositional Logic (Recap)

- A simple language useful for showing key ideas and definitions
- User defines a set of propositional symbols, like P and Q.
- User defines the **semantics** of each propositional symbol:
 - P means “It is hot”
 - Q means “It is humid”
 - R means “It is raining”
- A sentence (well formed formula) is defined as follows:
 - A symbol is a sentence
 - If S is a sentence, then $\neg S$ is a sentence
 - If S is a sentence, then (S) is a sentence
 - If S and T are sentences, then $(S \vee T)$, $(S \wedge T)$, $(S \rightarrow T)$, and $(S \leftrightarrow T)$ are sentences
 - A sentence results from a finite number of applications of the above rules

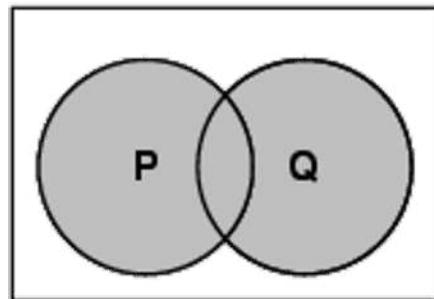
Truth tables

The five logical operators:

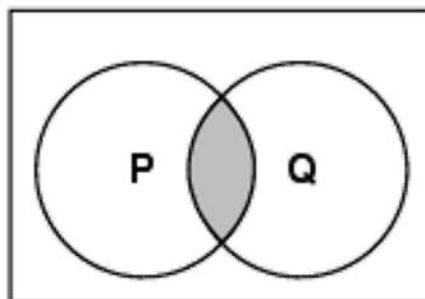
P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>

Truth Tables through SETs

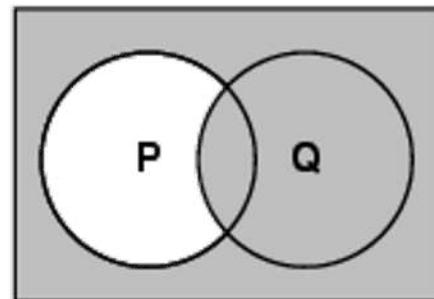
$P \vee Q$



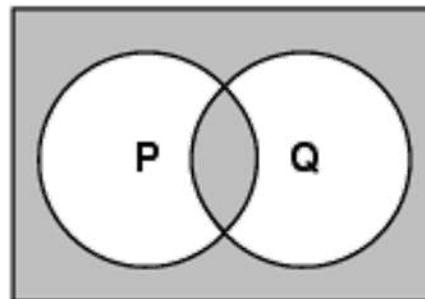
$P \wedge Q$



$P \Rightarrow Q$



$P \Leftrightarrow Q$



Exercise

Are the statements $P \rightarrow(Q \vee R)$ and $(P \rightarrow Q) \vee(P \rightarrow R)$ logically equivalent? Explain your answer.

Exercise

Are the statements $P \rightarrow(Q \vee R)$ (1) and $(P \rightarrow Q) \vee(P \rightarrow R)$ (2) logically equivalent?
Explain your answer.

Solution 1. Truth Table.

Solution 2.

Step 1. (Implication Elimination) (1) $P \rightarrow(Q \vee R) \equiv \neg p \vee(q \wedge r)$.

Step 2. (Implication Elimination) (2) $(p \rightarrow q) \wedge(p \rightarrow r) \equiv (\neg p \vee q) \wedge(\neg p \vee r)$

Step 3. (Distribution law) (2) $(\neg p \vee q) \wedge(\neg p \vee r) \equiv \neg p \vee(q \wedge r)$ (1)

How can we express “Every person is mortal.” with propositional logic?

How can we express “Every person is mortal.” with propositional logic?

WE CAN NOT. WE NEED FIRST-ORDER LOGIC!!!

First-Order Logic

Propositional logic is a weak language

- Hard to identify “individuals” (e.g., Mary, 3)
- Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)
- Generalizations, patterns, regularities can’t easily be represented (e.g., “all triangles have 3 sides”)
- First-Order Logic (abbreviated FOL) is expressive enough to concisely represent this kind of information

FOL adds relations, variables, and quantifiers, e.g.,

- “Every elephant is gray”: $\forall x (\text{elephant}(x) \rightarrow \text{gray}(x))$
- “There is a white alligator”: $\exists x (\text{alligator}(X) \wedge \text{white}(X))$

First-order logic

- First-order logic (FOL) models the world in terms of
 - **Objects**, which are things with individual identities
 - **Properties** of objects that distinguish them from others
 - **Relations** that hold among sets of objects
 - **Functions**, which are a subset of relations where there is only one “value” for any given “input”
- Examples:
 - Objects: Students, lectures, companies, cars ...
 - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
 - Properties: blue, oval, even, large, ...
 - Functions: father-of, best-friend, second-half, more-than ...

FOL Provides

- **Variable symbols**
 - E.g., x , y , foo
- **Connectives**
 - Same as in propositional logic: not (\neg), and (\wedge), or (\vee), implies (\rightarrow), iff (\leftrightarrow)
- **Quantifiers**
 - Universal $\forall x$ or (\mathbf{Ax})
 - Existential $\exists x$ or (\mathbf{Ex})

Sentences: built from terms and atoms

- **quantified sentences** adds quantifiers \forall and \exists
 - $\forall x \text{ loves}(x, \text{mother}(x))$
 - $\exists x \text{ number}(x) \wedge \text{greater}(x, 100) \wedge \text{prime}(x)$
- A **well-formed formula (wff)** is a sentence containing no “free” variables, i.e., all variables are “bound” by either a universal or existential quantifiers
 - ($\forall x$)P(x,y) has x bound as a universally quantified variable, but y is free

Quantifiers

- **Universal quantification**

- $(\forall x)P(x)$ means P holds for **all** values of x in domain associated with variable
 - E.g., $(\forall x) \text{ dolphin}(x) \rightarrow \text{mammal}(x)$

- **Existential quantification**

- $(\exists x)P(x)$ means P holds for **some** value of x in domain associated with variable
 - E.g., $(\exists x) \text{ mammal}(x) \wedge \text{lays_eggs}(x)$
 - This lets us make a statement about some object without naming it

Quantifier Scope

- FOL sentences have structure, like programs
- In particular, the variables in a sentence have a scope
- For example, suppose we want to say
 - “everyone who is alive loves someone”
 - $(\forall x) \text{alive}(x) \rightarrow (\exists y) \text{loves}(x,y)$
- Here’s how we scope the variables

$$(\forall x) \text{alive}(x) \rightarrow (\exists y) \text{loves}(x,y)$$


— Scope of x

— Scope of y

Quantifier Scope

- **Switching order of universal quantifiers does *not* change the meaning**
 - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x) P(x,y)$
 - “Dogs hate cats” (i.e., “all dogs hate all cats”)
- **You can switch order of existential quantifiers**
 - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x) P(x,y)$
 - “A cat killed a dog”
- **Switching order of universal and existential quantifiers does change meaning:**
 - Everyone likes someone: $(\forall x)(\exists y) \text{ likes}(x,y)$
 - Someone is liked by everyone: $(\exists y)(\forall x) \text{ likes}(x,y)$

Connections between \forall and \exists

- We can relate sentences involving \forall and \exists using extensions to **De Morgan's laws**:

$$1. (\forall x) \neg P(x) \leftrightarrow \neg(\exists x) P(x)$$

$$2. \neg(\forall x) P \leftrightarrow (\exists x) \neg P(x)$$

$$3. (\forall x) P(x) \leftrightarrow \neg(\exists x) \neg P(x)$$

$$4. (\exists x) P(x) \leftrightarrow \neg(\forall x) \neg P(x)$$

- Examples

1. All dogs don't like cats \leftrightarrow No dogs like cats

2. Not all dogs dance \leftrightarrow There is a dog that doesn't dance

3. All dogs sleep \leftrightarrow There is no dog that doesn't sleep

4. There is a dog that talks \leftrightarrow Not all dogs can't talk

Exercise I

Translate these English statements into First Order Logic.

1. Every student takes Introduction to AI.
2. There is some course that every student has not taken.

Exercise I - Solutions

Translate these English statements into First Order Logic.

1. Every student takes Introduction to AI.

Solution: $\forall x, \text{Student}(x) \Rightarrow \text{Takes}(x, \text{Introduction to AI})$

2. There is some course that every student has not taken.

Solution: $\exists x, \text{Course}(x) [\forall y, \text{Student}(y) \Rightarrow \neg \text{Taken}(y, x)]$

Exercise II

This exercise uses the function `Borders(x,y)` and `Country(x)` whose arguments are geographical regions, along with constant symbols for various regions. We give an English sentence and a number of candidate logical expressions. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; or (2) is syntactically valid but does not express the meaning of the English sentence.

All countries that border Ecuador are in South America.

1. $\forall c \text{Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \wedge \text{In}(c, \text{SouthAmerica})$
2. $\forall c \text{Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})$
3. $\forall c \text{Country}(c) \Rightarrow [\text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})]$

Exercise II - Solution

1. $\forall c \text{Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \wedge \text{In}(c, \text{SouthAmerica})$

Answer: Does not correctly express the sentence. It reads:
“Every country borders Ecuador and is in South America”

2. $\forall c \text{Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})$

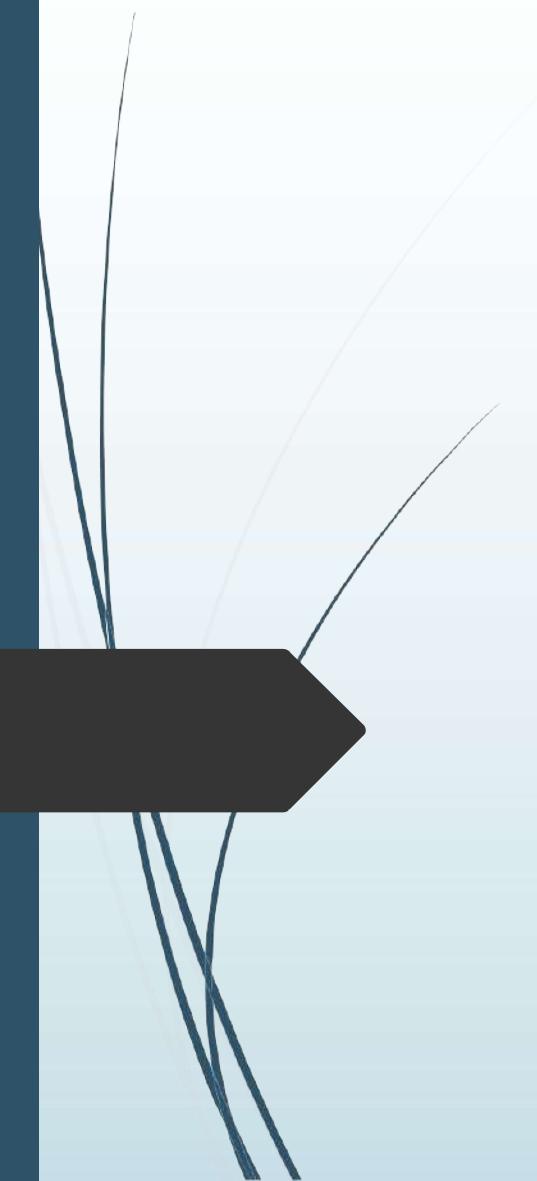
Answer: Correctly express the English sentence.

3. $\forall c \text{Country}(c) \Rightarrow [\text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})] \equiv$
 $\forall c \neg \text{Country}(c) \vee [\neg \text{Border}(c, \text{Ecuador}) \vee \text{In}(c, \text{SouthAmerica})] \equiv$
 $\forall c [\neg \text{Country}(c) \vee \neg \text{Border}(c, \text{Ecuador})] \vee \text{In}(c, \text{SouthAmerica}) \equiv$
 $\forall c \neg [(\text{Country}(c) \wedge \neg \text{Border}(c, \text{Ecuador})) \vee \text{In}(c, \text{SouthAmerica})] \equiv$
 $\forall c \text{Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})$ -

Answer: Equivalent with 2.

Three wooden blocks are arranged horizontally, spelling out "Q & A". The blocks are light-colored wood with dark, bold, sans-serif letters. The letter "Q" is on the left, "&" is in the middle, and "A" is on the right. They are placed on a dark brown wooden surface. The background is a soft-focus, warm-toned outdoor scene with green and yellow foliage.

Q & A



Introduction to Artificial Intelligence

AI and Logics - Week 2

About me

Name:

Office hours:

Contact:

3

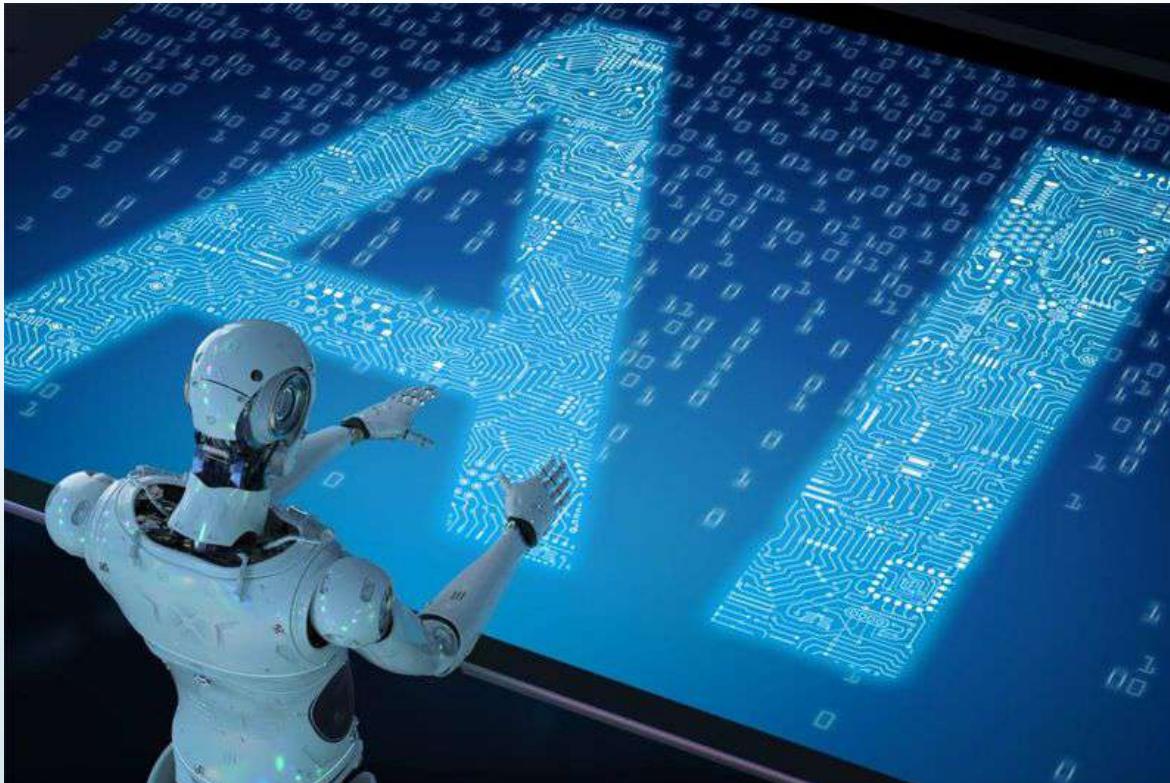
Agreements

Agenda

1. Discussion about AI
2. Tasks about Logics

How AI is changing the world?

5



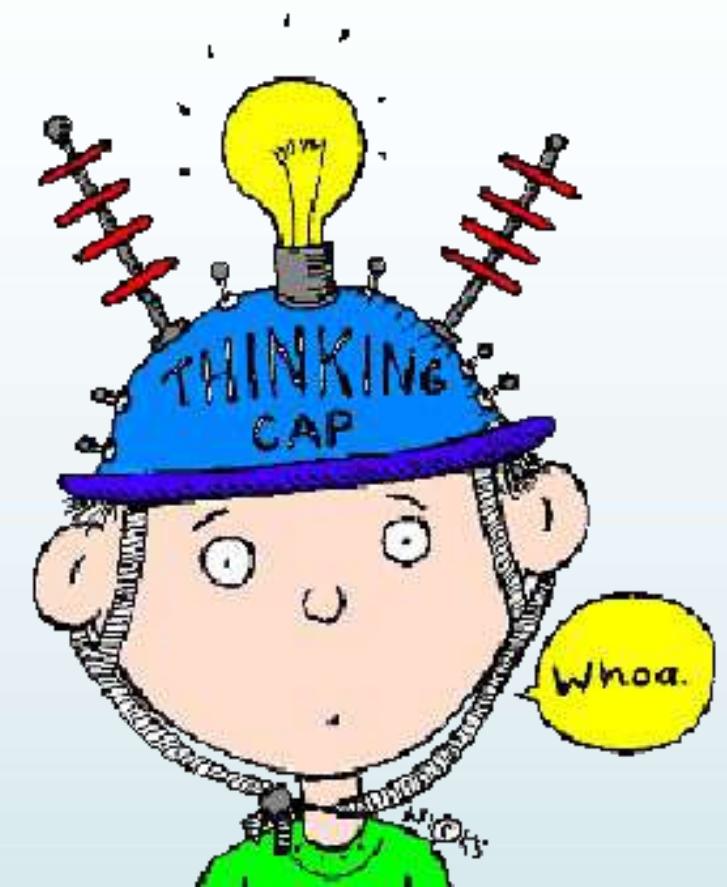
Due to AI, which things will be
obsolete in the near future?

Reason about the following Questions

The thing you identify to be obsoleted by AI in the near future:

1. Will bring a positive change towards humanity or a negative change?
2. Explain the pros and cons of this change.

Wear your thinking cap
and discuss the question
in the next slide

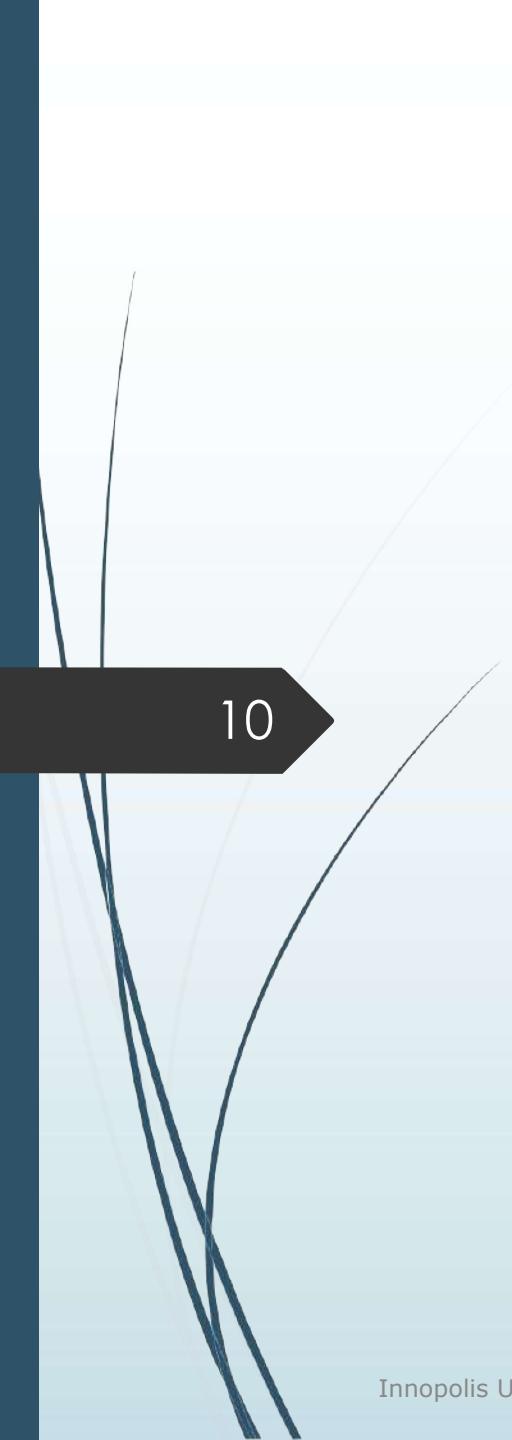


Can AI be smarter than Humans?

If Yes, How do you justify?

If No, How do you rationalize?





10

Let's Dive into Logics

Propositional Logic - Operations

For Every \forall

- For Every X its value is 3; $\forall x=3$

There exists \exists

- There is an X which is 3; $\exists x=3$

Conjunction \wedge

- X and Y; $x \wedge y$

Disjunction \vee

- X or Y; $x \vee y$

Implication \rightarrow

- If x then y; $x \rightarrow y$

Biconditional \leftrightarrow

- X if and only if Y; $x \leftrightarrow y$

Such that (s.t.)

Logic Examples

- $\text{REDHAIR}(X)$ – X has red hair
- $\text{EVIL}(X)$ – X is evil
- $\text{GLASSES}(X)$ – X wears glasses
- $\text{FEMALE}(X)$ – X is female

13

Logics Task 1

Convert the natural language into propositional language:

For all Innopolis students, there exists one, who is from Kazan or who hates programming.

Logics Task 1 - Solution

Convert the natural language into propositional language:

For all Innopolis students, there exists one, who is from Kazan or who hates programming.

$$\forall X \text{INNO_STUDENTS}(X) \rightarrow \exists X \text{KAZAN}(X) \vee \text{PROGRAMMING_HATER}(X)$$

Logics Task 2

Convert the propositional language into natural language:

$$\forall x \text{ STUDENTS}(x) \wedge \text{PROGRAMMER}(x) \wedge \text{KAZAN}(x) \rightarrow \exists x \text{ INNO_STUDENT}(x)$$

Logics Task 2 - Solution

Convert the propositional language into natural language:

$$\forall X \text{ STUDENTS}(X) \wedge \text{PROGRAMMER}(X) \wedge \text{KAZAN}(X) \rightarrow \exists X \text{ INNO_STUDENT}(X)$$

For all students who are programmers and from Kazan, there exists an Innopolis student.

Logics Task 3

Convert the propositional language into truth table:

$$\forall X \text{ STUDENTS}(X) \wedge \text{PROGRAMMER}(X) \wedge \text{KAZAN}(X) \rightarrow \exists X \text{ INNO_STUDENT}(X)$$

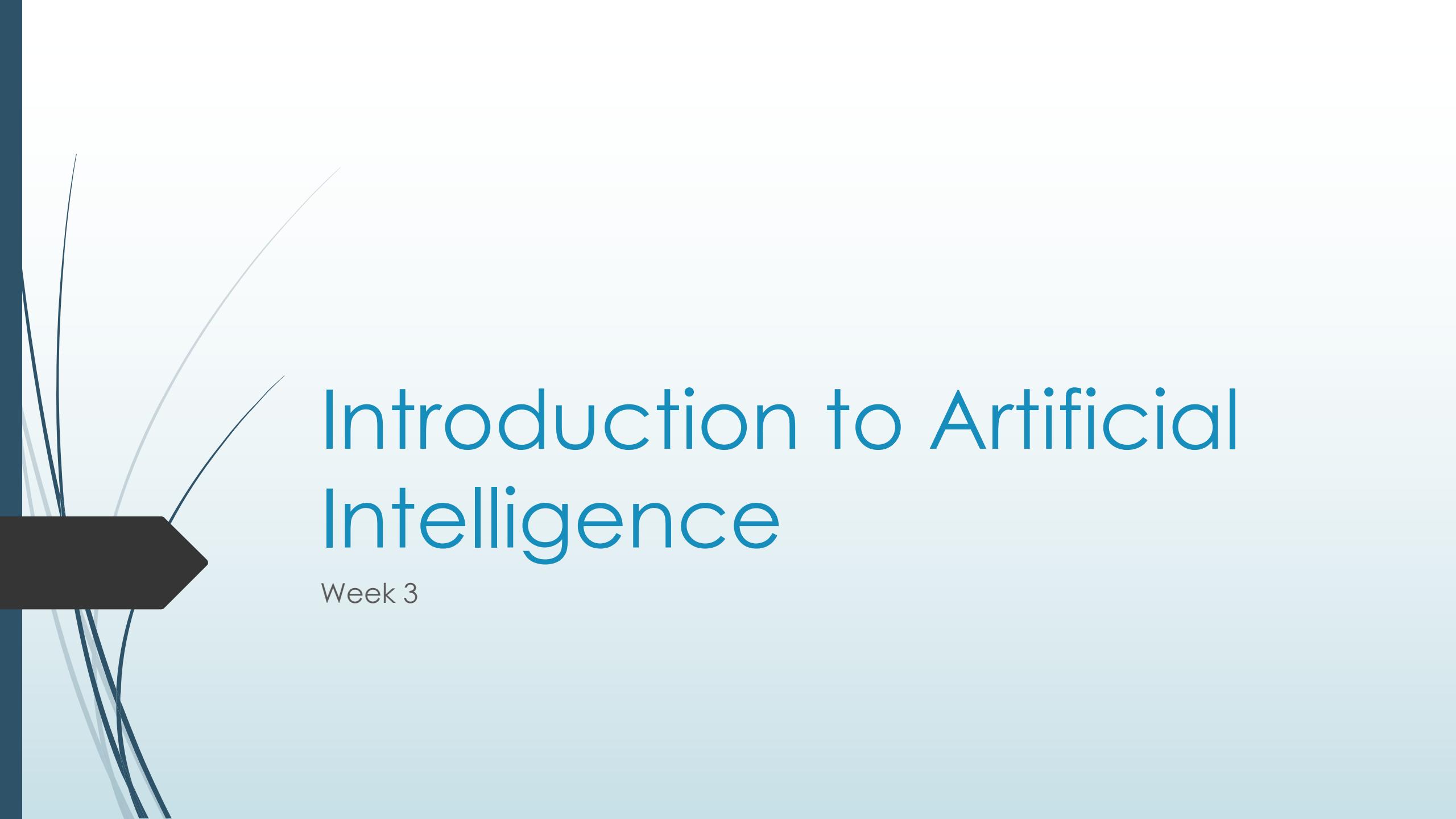
Logics Task 3 - Solution

Convert the propositional language into truth table:

$$\forall X \text{ STUDENTS}(X) \wedge \text{PROGRAMMER}(X) \wedge \text{KAZAN}(X) \rightarrow \exists X \text{ INNO_STUDENT}(X)$$

STUDENTS	PROGRAMMER	KAZAN	INNOSTUDENT	Expression
T	T	T	F	F
ANYTHING ELSE				T

Thanks for participating



Introduction to Artificial Intelligence

Week 3

2

Agents

Mr. Anderson....



What is an Agent?

- ▶ An agent is anything which:
 - ▶ Lives in an environment
 - ▶ Can perceive the environment via its sensors
 - ▶ Can act upon the environment via its actuators
- ▶ Agents also have memory of the current and past perceptions of the environment
 - ▶ Percept – the current state
 - ▶ Percept Sequence – the complete history of Percepts



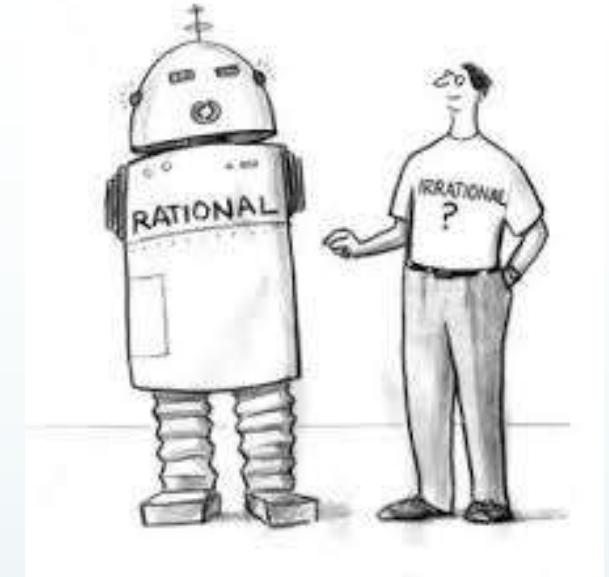
Humans as Agents

- ▶ Environments
 - ▶ Home
 - ▶ School
 - ▶ Work
- ▶ Sensors
 - ▶ Eyes
 - ▶ Ears
 - ▶ Mouth
 - ▶ Nose
 - ▶ Skin
- ▶ Actuators
 - ▶ Hands
 - ▶ Feet
 - ▶ Speech



Rationality

- ▶ What is rational at any given time depends upon:
 - ▶ The performance measure defines success
 - ▶ The agent's prior knowledge of the environment
 - ▶ The actions which an agent can perform
 - ▶ The agent's percept sequence to date
- ▶ Definition:
 - ▶ For Each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has



Agents are Rational, Not Necessarily Omniscient

- ▶ A jester knocks over the king's wine
 - ▶ "What an idiot!"
 - ▶ Did the jester make a cartwheel and miss the landing?
 - ▶ Did the jester see it being poisoned and the king did not?
- ▶ Agents need to gather information before making a decision of an action
 - ▶ "Why did you spill my wine jester?!"
- ▶ Note that agents may perceive differently from each other
- ▶ Agents have a degree of autonomy
 - ▶ The King does not control every action of the jester



PEAS – Performance Measure, Environment, Actuators, Sensors



- ▶ Agent Type
 - ▶ Jester
- ▶ Performance measure
 - ▶ The King is Happy
- ▶ Environment
 - ▶ The Court
- ▶ Actuators
 - ▶ Tumbling
 - ▶ Juggling
 - ▶ Joke Telling
- ▶ Sensors
 - ▶ Hearing Laughter
 - ▶ Seeing Smiles

Properties of the Environment (1)



- ▶ Fully Observable v. Partially Observable
 - ▶ Can the agent see everything in the world via its sensors?
 - ▶ Does it miss elements which are relevant to the decision making process?
- ▶ Single v. Multiple Agent
 - ▶ Is there only one agent within the space?
 - ▶ Does there need to be some form of competitive or cooperation between agents?
 - ▶ How do the agents communicate?

Properties of the Environment (2)



- ▶ Deterministic v. Stochastic
 - ▶ Can we determine the next state of the environment given the previous state and the action we are applying into it?
 - ▶ We can be certain in environments which are fully observable and deterministic
- ▶ Episodic v. Sequential
 - ▶ Episodic environments allow for atomic actions not dependant upon those previous
 - ▶ Sequential actions are those which the current decision has consequences on future actions

Properties of the Environment (3)



- ▶ Static v. Dynamic
 - ▶ Does the environment change as the agent is thinking about its actions?
 - ▶ Semi-dynamic if the environment does not change as the time passes by but the agent's performance score does
- ▶ Discrete v. Continuous
 - ▶ Are there states in the environment, or does the environment smoothly act over time
 - ▶ Chess – discrete movements
 - ▶ Jenga – continuous actions of physics on the blocks
- ▶ Known v. Unknown
 - ▶ Does the designer of the agent have full knowledge of the rules of the environment?

The Worst Case

- ▶ An environment which is partially observable, multiagent, stochastic, sequential, dynamic, continuous, and unknown
- ▶ Ordering in a new restaurant in a new country
 - ▶ Can't see how they make the food
 - ▶ Have to deal with the waiter, other customers
 - ▶ Can't predict when the food will come or how exactly it will taste
 - ▶ Depending on what I order, I might order more or might order an ambulance
 - ▶ Maybe they don't have anymore what I would like to order
 - ▶ Time keeps on ticking
 - ▶ Don't know the cultural rules



Testing an AI

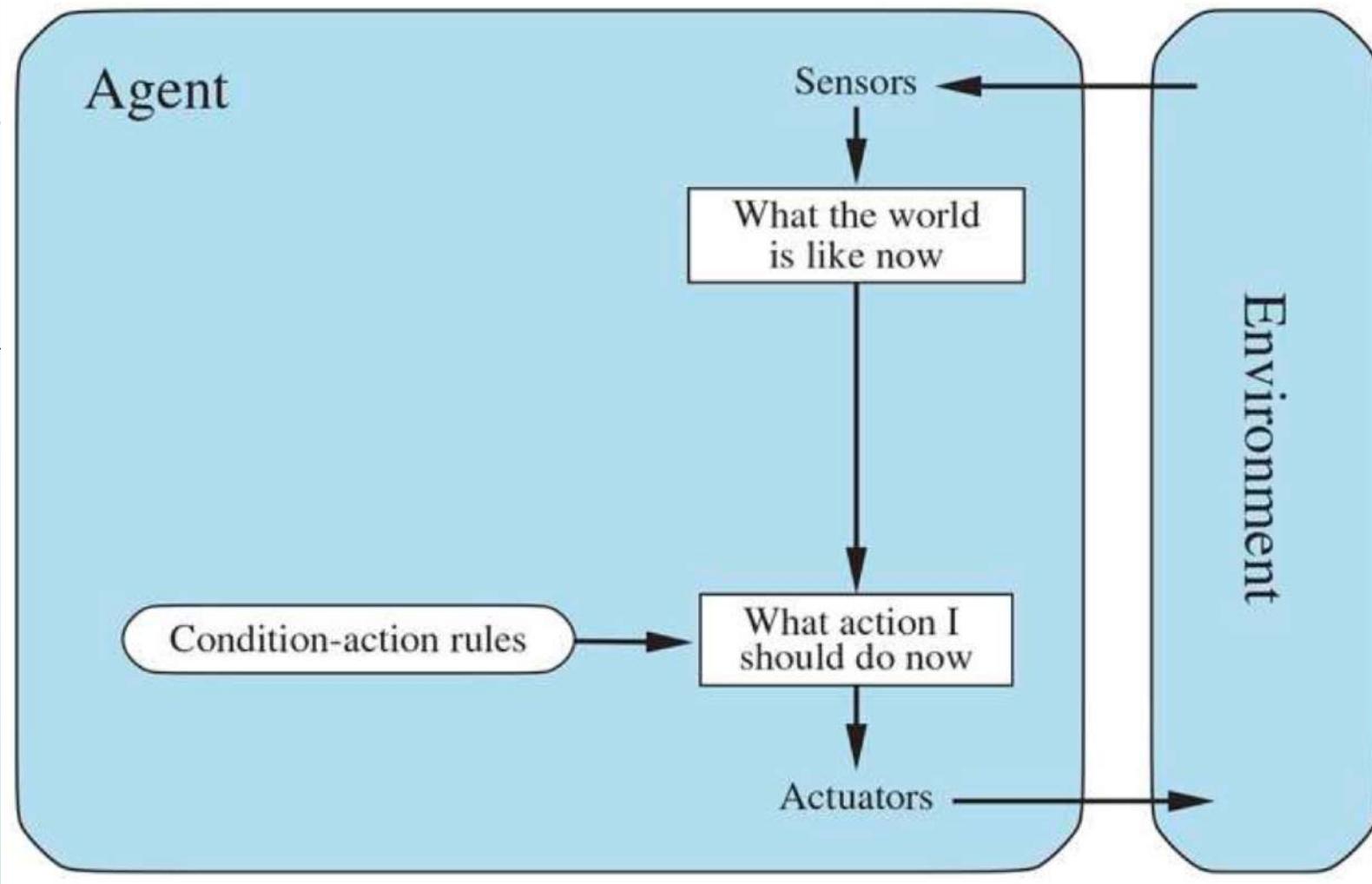
- ▶ We do not just want one environment but an environment class
 - ▶ Run multiple simulations with many different changes to the environment
- ▶ Self-driving Car
 - ▶ Traffic level
 - ▶ Lights
 - ▶ Weather
- ▶ Collectable Card Game Player
 - ▶ Against many other decks
 - ▶ Other player agents



Types of Agents

1. Simple reflex agents
2. Model-based reflex agents
3. Goal-based agents
4. Utility-based agents

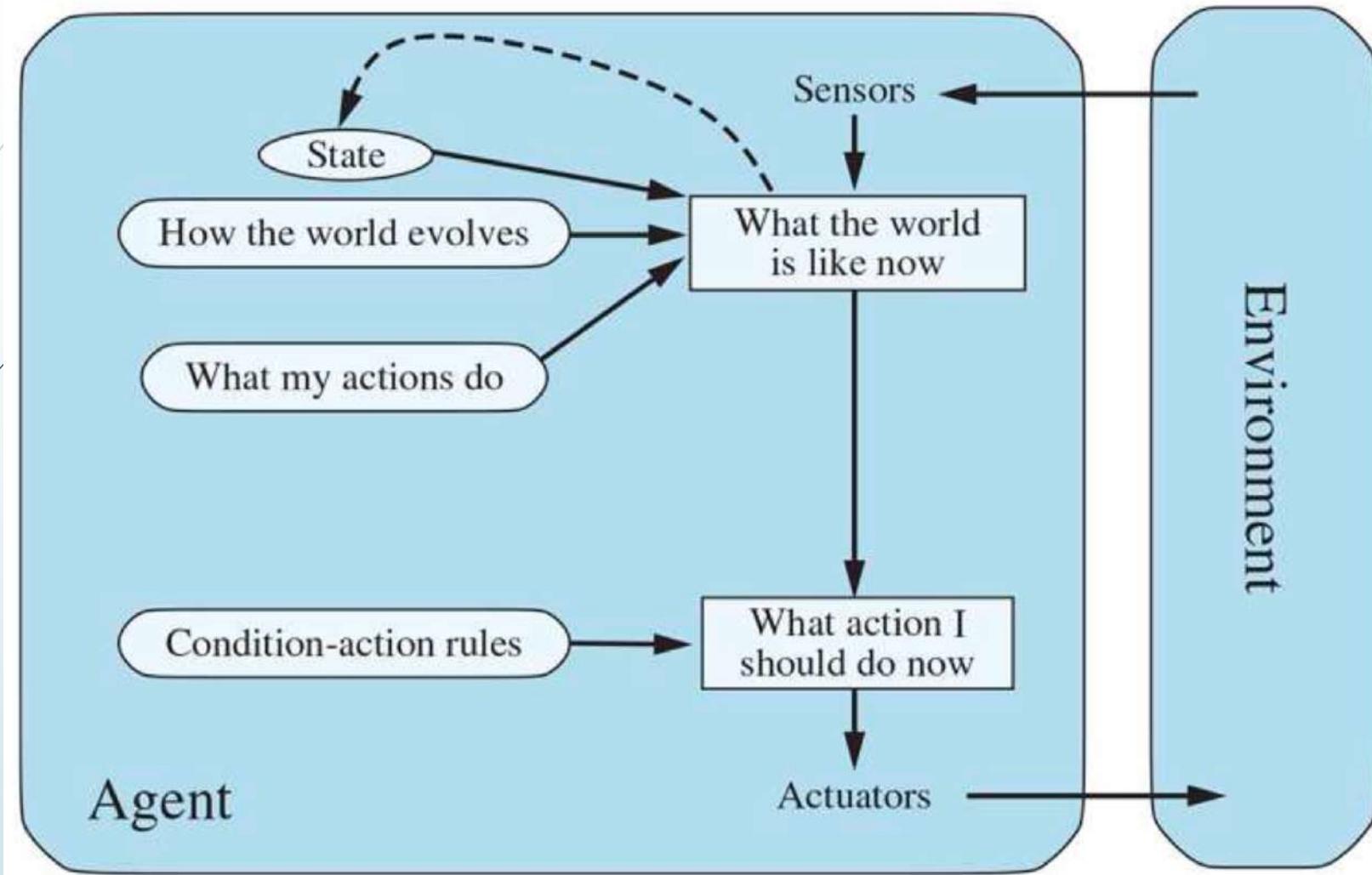
Simple Reflex Agents



Simple Reflex Agents

- ▶ Only current percept matters
- ▶ Work only in fully observable environment
- ▶ Too many condition-action rules may be required

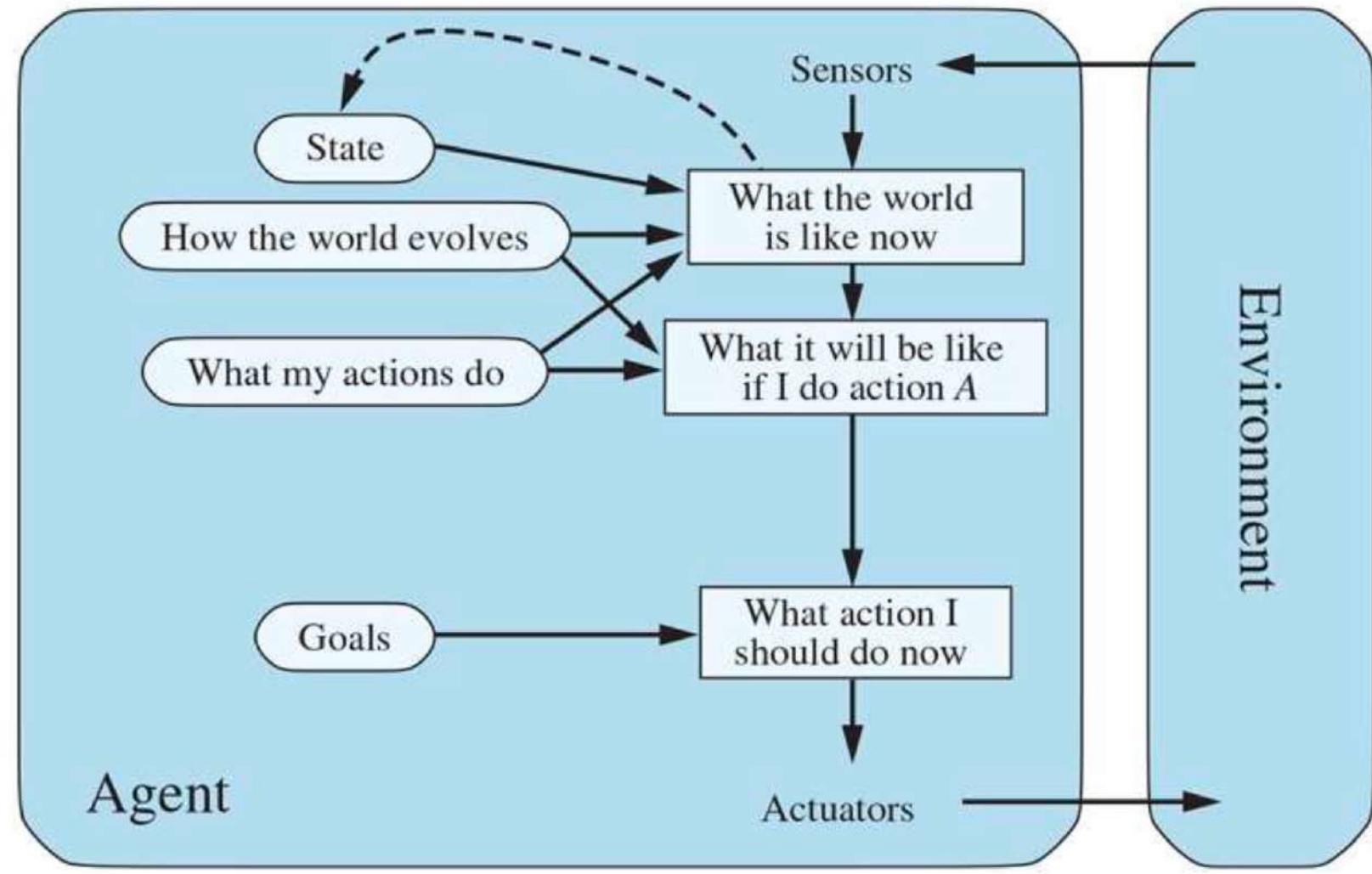
Model-based Reflex Agents



Model-based Reflex Agents

- ▶ Work better in partially observable environment because of keeping track of the world
- ▶ Internal state exists

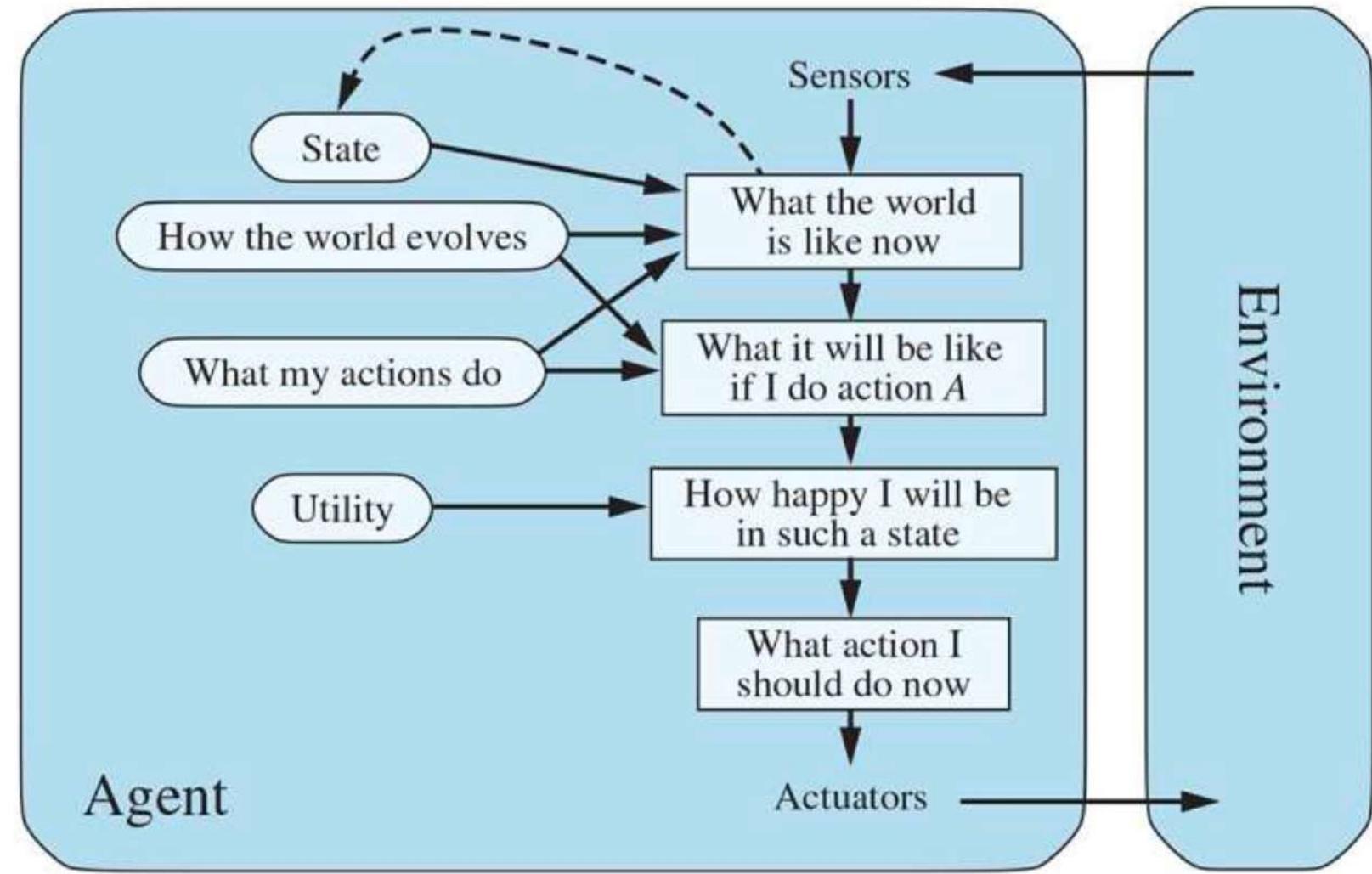
Goal-based Agents



Goal-based Agents

- ▶ More flexible than previous agent types
- ▶ Goal replacement does not entail rewriting of all the rules
- ▶ May be hard to choose between multiple goals

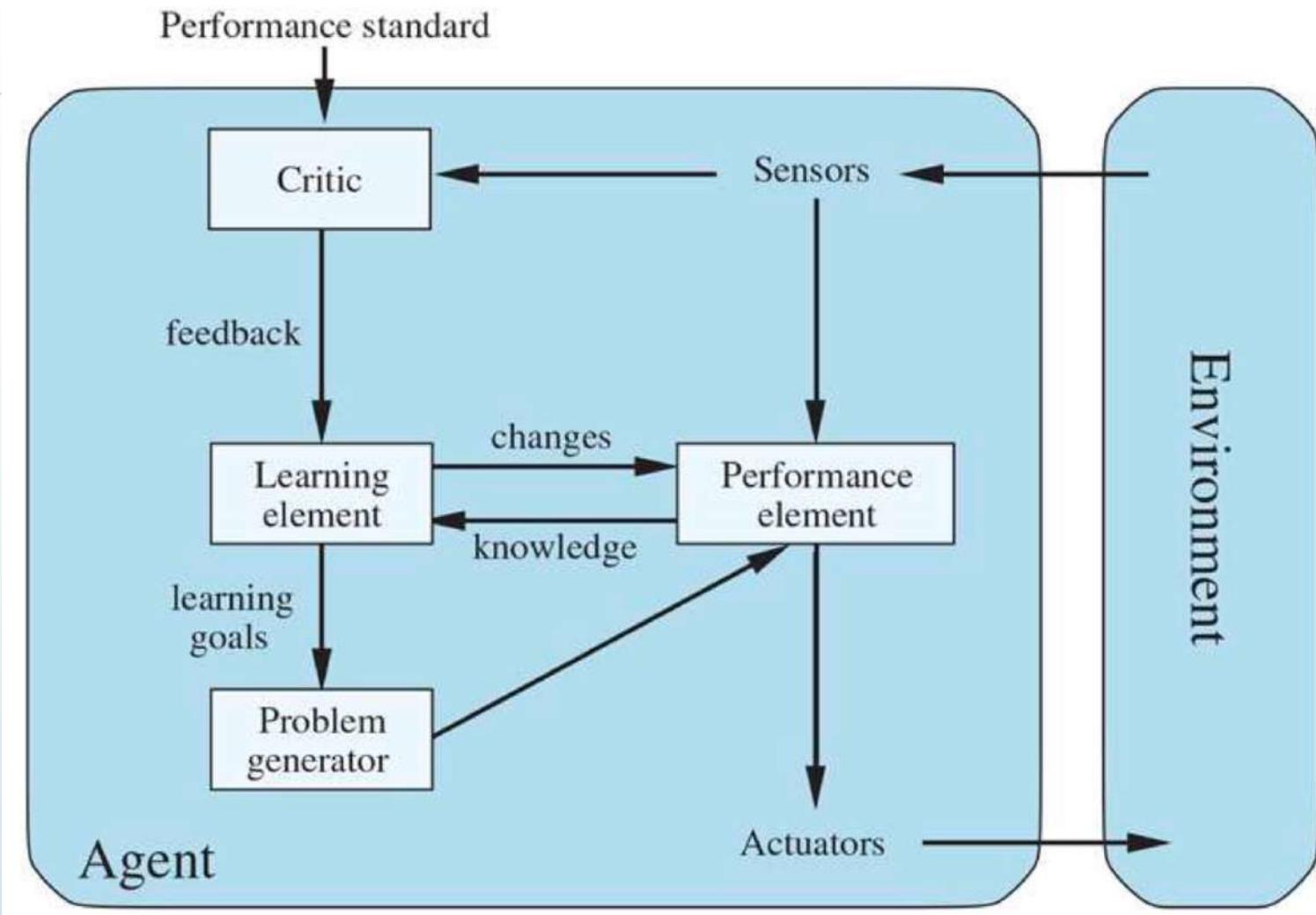
Utility-based Agents



Utility-based Agents

- ▶ Allow to maximize the “happiness” of the agent
- ▶ Allows to find the tradeoff between the conflicting goals

Learning Agents



Reasoning and Knowledge-based agents

Darko Bozhinoski,
Ph.D. in Computer Science

Lecture Objectives

- ❖ Types of Reasoning
- ❖ Agent modeling using Logic

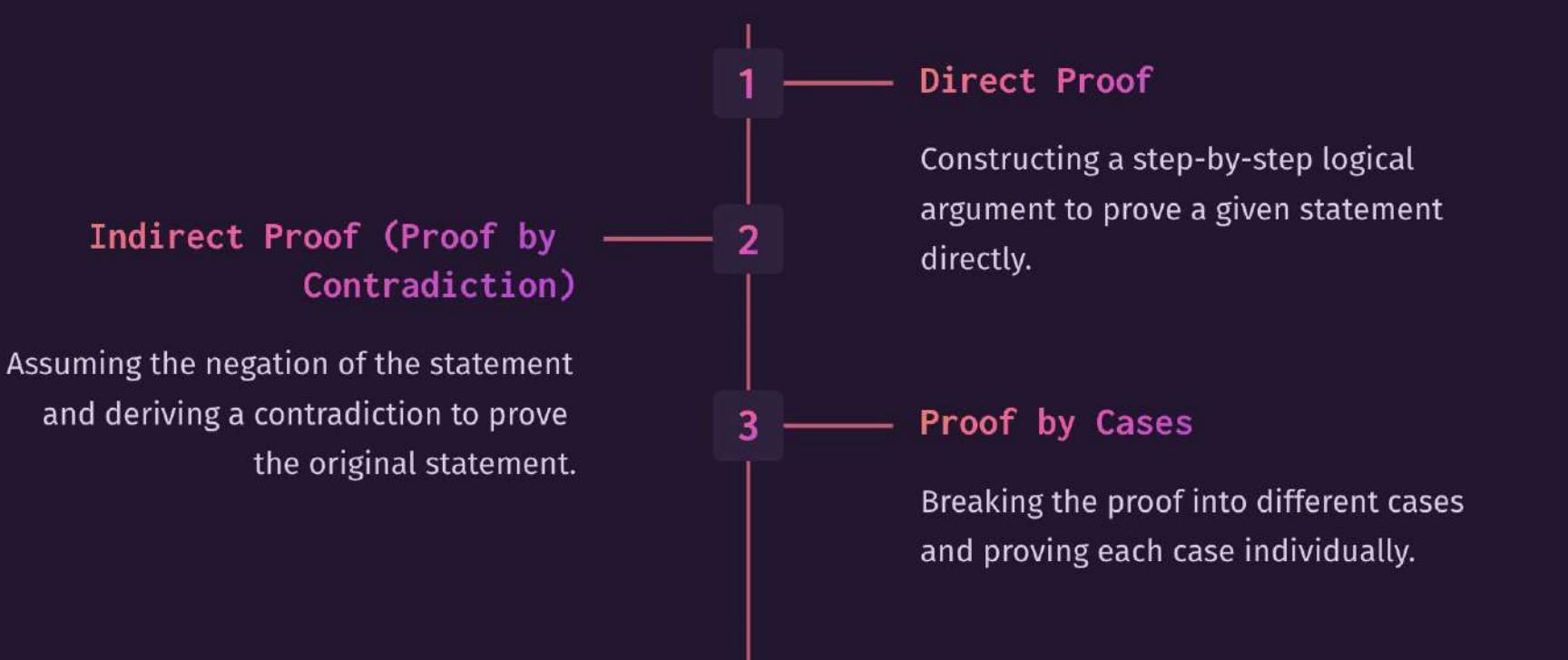
Types of reasoning

Deductive Reasoning: Using general statements to draw specific conclusions with certainty.

Inductive Reasoning: a method of drawing conclusions by going from the specific to the general. (bottom-up approach)

Counterexamples: Providing a specific example that disproves a universal statement.

Logic Proof Techniques



Induction: Example in Mathematics

Principle of Induction. Let P be any property of natural numbers.

Suppose P holds of zero, and whenever

P holds of a natural number n , then it holds of its successor, $n+1$.

Then P holds of every natural number.

Theorem. For every natural number n , $1+2+\dots+2^n=2^{n+1}-1$.

Example in Mathematics (2)

Theorem. For every natural number n , $1+2+\dots+2^n=2^{n+1}-1$.

Proof. We prove this by induction on n . In the base case, when $n = 0$, we have $1 = 2^{0+1} - 1$, as required.

For the induction step, fix n , and assume the *inductive hypothesis*

$$1 + 2 + \dots + 2^n = 2^{n+1} - 1.$$

We need to show that this same claim holds with n replaced by $n + 1$. But this is just a calculation:

$$\begin{aligned} 1 + 2 + \dots + 2^{n+1} &= (1 + 2 + \dots + 2^n) + 2^{n+1} \\ &= 2^{n+1} - 1 + 2^{n+1} \\ &= 2 \cdot 2^{n+1} - 1 \\ &= 2^{n+2} - 1. \end{aligned}$$

Deduction Reasoning

In a simple deductive logic argument, you'll often begin with a premise, and add another premise. Then, you form a conclusion based on these two premises. This format is called “premise-premise-conclusion.”

Here, A **proof** is an argument from **hypotheses** (assumptions) to a **conclusion**. Each step of the argument follows the laws of logic.

Axioms and theorems

- **Axioms** are facts and rules that attempt to capture all of the (important) facts and concepts about a domain; axioms can be used to prove **theorems**
 - Mathematicians don't want any unnecessary (dependent) axioms –ones that can be derived from other axioms
 - Dependent axioms can make reasoning faster, however
 - Choosing a **good set of axioms for a domain is a kind of design problem**

The Axioms of Euclidean Plane Geometry

For well over two thousand years, people had believed that only one geometry was possible, and they had accepted the idea that this geometry described reality. One of the greatest Greek achievements was setting up rules for plane geometry. This system consisted of a collection of undefined terms like point and line, and five axioms from which all other properties could be deduced by a formal process of logic. Four of the axioms were so self-evident that it would be unthinkable to call any system a geometry unless it satisfied them:

1. A straight line may be drawn between any two points.
2. Any terminated straight line may be extended indefinitely.
3. A circle may be drawn with any given point as center and any given radius.
4. All right angles are equal.

But the fifth axiom was a different sort of statement:

5. If two straight lines in a plane are met by another line, and if the sum of the internal angles on one side is less than two right angles, then the straight lines will meet if extended sufficiently on the side on which the sum of the angles is less than two right angles.

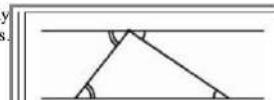
Because this axiom was much more complicated than the previous axioms, it seemed more like a theorem than a self-evident proposition. Since all attempts to deduce it from the first four axioms had failed, Euclid simply included it as an axiom because he knew he needed it. For example, some axiom like this one was necessary for proving one of Euclid's most famous theorems, that the sum of the angles of a triangle is 180 degrees. Mathematicians found alternate forms of the axiom that were easier to state, for example:

- 5'. For any given point not on a given line, there is exactly one line through the point that does not meet the given line.

This form of the fifth axiom became known as the parallel postulate. Although it was simpler to understand than Euclid's original formulation, it was no easier to deduce from the earlier axioms. The attempt to deduce the fifth axiom remained a great challenge right up to the nineteenth century, when it was proved that the fifth axiom did not follow from the first four.

The great advantage of expressing geometry as an axiomatic system was that it no longer was necessary to memorize long lists of independent facts about the nature of the universe—one only had to know a small set of axioms, and by applying them to the rules of inference, one could reconstruct the entire collection of geometric truths.

There was little doubt that the Greeks were attempting to describe a real world when they formulated their geometry, even though it might have been an ideal sort of world, realized only abstractly "in the mind of God." Many mathematicians, now as well as in the distant past, believe that the complete structure of mathematics is something that exists in itself and that it is only gradually discovered by human beings laboring to uncover its mysteries. Even though the framers of the early axiom systems would refer to point and line as undefined terms, they fairly clearly thought of them as real objects, and they thought that the system they were developing was a progressively more and more elaborate and accurate description of the real world. The progress of algebra, on the other hand, was not quite so settled, and people accepted changes in viewpoint there more readily than in the very traditional field of geometry.



Once we can draw a unique line through one vertex of a triangle not meeting the line containing the opposite side, we can use alternate interior angles to see that the sum of the angles of a triangle is the same as a straight angle, 180 degrees.

▶ [Noncommutative Algebra](#)

https://www.youtube.com/watch?v=LPET_HhN0VM

GENERAL CONCEPTS

- **Model:** an interpretation of a set of sentences where every sentence is *True. (a possible world)*.
- E.g. having x men and y women sitting at a table playing bridge, and the sentence $x + y = 4$ is true when there are four people in total. Formally, the possible models are just all possible assignments of real numbers to the variables x and y . Each such assignment fixes the truth of any sentence of arithmetic whose variables are x and y . If a sentence α is true in model m , we say that m **satisfies** α or m **is a model of** α .

GENERAL CONCEPTS (2)

- A sentence is
 - **satisfiable** if it is true under some interpretation
 - **valid** if it is true under all possible interpretations
 - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence:** $S \models X$ if all models of S are also models of X
 - involves the relation of logical **entailment** between sentences—the idea that a sentence *follows logically* from another sentence.

Entailment and derivation

- **Entailment: $\text{KB} \models Q$**

- Q is entailed by KB (a set of premises or assumptions) if and only if there is no logically possible world in which Q is false while all the premises in KB are true.

- **Derivation: $\text{KB} \vdash Q$**

- We can derive Q from KB if there is a proof consisting of a sequence of valid inference steps starting from the premises in KB and resulting in Q

Two important properties for inference

Soundness: If $\text{KB} \vdash Q$ then $\text{KB} \models Q$

- If Q is derived from a set of sentences KB using a given set of rules of inference, then Q is entailed by KB .
- Soundness: only true statements can be proven using the inference rules.

Completeness: If $\text{KB} \models Q$ then $\text{KB} \vdash Q$

- If Q is entailed by a set of sentences KB , then Q can be derived from KB using the rules of inference.
- Completeness guarantees that we can prove everything that is true

Basic rules of inferences

Modus Ponens: If P implies Q, and P is true, then Q is true.

Modus Tollens: If P implies Q, and Q is false, then P is false.

Constructing a Conjunction: If P is true and Q is true, then P and Q is true.

Rule of Simplification: If P and Q is true, then P is true.

Rule of Addition: If P is true, then P or Q is true.

Rule of Disjunctive Syllogism: If P or Q is true, and P is false, then Q is true.

Rule of Hypothetical Syllogism: If P implies Q, and Q implies R, then P implies R.

Some of the rules of inference

- Each can be shown to be sound using a truth table

<u>RULE</u>	<u>PREMISE</u>	<u>CONCLUSION</u>
-------------	----------------	-------------------

Modus Ponens	$A, A \rightarrow B$	B
--------------	----------------------	-----

And Introduction	A, B	$A \wedge B$
------------------	--------	--------------

And Elimination	$A \wedge B$	A
-----------------	--------------	-----

Double Negation	$\neg\neg A$	A
-----------------	--------------	-----

Unit Resolution	$A \vee B, \neg B$	A
-----------------	--------------------	-----

Resolution	$A \vee B, \neg B \vee C$	$A \vee C$
-------------------	---------------------------	------------

Example of Modus Ponens

Premise 1: $(A \vee \neg B) \rightarrow \neg C$

Premise 2: $\neg D$

Premise 3: $(A \vee \neg B)$

Example of Modus Ponens

Premise 1: $(A \vee \neg B) \rightarrow \neg C$

Premise 2: $\neg D$

Premise 3: $(A \vee \neg B)$

Conclusion: $\neg C$ (Modus Ponens (1,3))

Soundness of modus ponens

A	B	$A \rightarrow B$	OK?
True	True	True	✓
True	False	False	✓
False	True	True	✓
False	False	True	✓

Soundness of the resolution inference rule

Exercise I

Premise 1: $\neg A \rightarrow (C \wedge D)$

Premise 2: $A \rightarrow B$.

Premise 3: $\neg B$

Prove: C.

Exercise II

Premise 1: $\neg(A \vee B) \rightarrow C$

Premise 2: $\neg A$

Premise 3: $\neg C$

Prove: B.

Exercise III

Suppose we have the following scenario with a large knowledge base (KB) consisting of the following statements:

1. If it is raining, then the streets are wet.
2. If the streets are wet, then people use umbrellas.
3. It is raining.
4. All dogs are mammals.
5. All mammals have hair.
6. Some animals have fur.
7. All birds have feathers.
8. Penguins are birds.
9. Penguins cannot fly.
10. All fruits have seeds.
11. Apples are fruits.
12. Oranges are fruits.
13. Bananas are fruits.
14. All vegetables are healthy.
15. Spinach is a vegetable.
16. Carrots are vegetables.
17. Broccoli is a vegetable.
18. All cars have wheels.
19. A Tesla is a car.
20. Teslas are electric vehicles.

Exercise III

Suppose we have the following scenario with a large knowledge base (KB) consisting of the following statements:

- | | |
|---|-----------------------------------|
| 1. If it is raining, then the streets are wet. | 11. Apples are fruits. |
| 2. If the streets are wet, then people use umbrellas. | 12. Oranges are fruits. |
| 3. It is raining. | 13. Bananas are fruits. |
| 4. All dogs are mammals. | 14. All vegetables are healthy. |
| 5. All mammals have hair. | 15. Spinach is a vegetable. |
| 6. Some animals have fur. | 16. Carrots are vegetables. |
| 7. All birds have feathers. | 17. Broccoli is a vegetable. |
| 8. Penguins are birds. | 18. All cars have wheels. |
| 9. Penguins cannot fly. | 19. A Tesla is a car. |
| 10. All fruits have seeds. | 20. Teslas are electric vehicles. |

Prove the statements:

- ❖ People are using umbrellas because it is raining.
- ❖ Penguins have feathers.
- ❖ Spinach is healthy.
- ❖ All dogs have hair.
- ❖ Penguins cannot fly.

Exercise IV

If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Given the above, can you prove that the unicorn is mythical? How about magical? Horned?

Solution.

Propositional logic statements of the given facts:

- (1) Mythical → \neg Mortal
- (2) \neg Mythical → Mortal \wedge Mammal
- (3) \neg Mortal \vee Mammal → Horned
- (4) Horned → Magical

We show how inference is done by direct application.

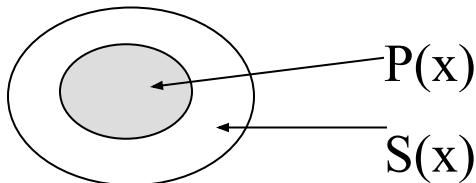
- (5) \neg Mythical \vee \neg Mortal From (1)
- (6a) (Mythical \vee Mortal) From (2)
- (6b) (Mythical \vee Mammal) From (2)
- (7) \neg Mortal \vee Mammal Resolution on (5) and (6b)
- (8) Horned Modus Ponens on (3) and (7)
- (9) Magical Modus Ponens on (4) and (8) Thus the unicorn is both magical and horned.

Necessary and sufficient conditions

- A **necessary** condition must be satisfied for a statement to be true.
- A **sufficient** condition, if satisfied, assures the statement's truth.
- Duality: “P is sufficient for Q” is the same as “Q is necessary for P.”

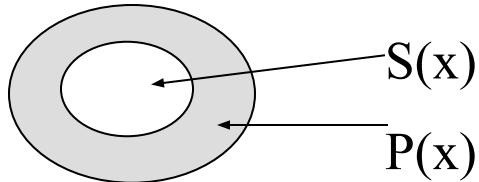
Necessary and sufficient conditions

$S(x)$ is a
necessary
condition of $P(x)$



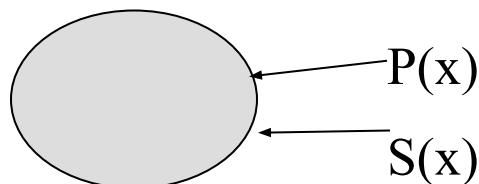
$$(\forall x) P(x) \Rightarrow S(x)$$

$S(x)$ is a
sufficient
condition of $P(x)$



$$(\forall x) P(x) \leq S(x)$$

$S(x)$ is a
necessary and
sufficient
condition of $P(x)$



$$(\forall x) P(x) \Leftrightarrow S(x)$$

Common mistakes to avoid in Rules of Inference and Logic Proofs

1. **Starting with a false premise:** If you start with a false premise, you can prove anything. It is important to ensure that your premises are true before proceeding with the proof.
2. **The fallacy of equivocation:** Errors can result from using undefined symbols, misplacing parentheses, or using symbols incorrectly. Using a term with multiple meanings inconsistently throughout an argument.
3. **The fallacy of affirming the consequent:** Using the wrong rule of inference can lead to an incorrect conclusion. It is important to understand the different rules of inference and when to use them.
4. **Failing to justify each step:** Each step of the argument must follow the laws of logic. It is important to justify each step of the proof to ensure that it is valid.
5. **Making errors in form:** When dealing with arguments in natural language, making errors in form is quite common. It is important to be familiar with the different forms of arguments and to ensure that the form of your argument is valid.

Summary

- The process of deriving new sentences from old one is called **inference**.
 - **Sound** inference processes derives true conclusions given true premises
 - **Complete** inference processes derive all true conclusions from a set of premises
- A **valid sentence** is true in all worlds under all interpretations
- If an implication sentence can be shown to be valid, then—given its premise—its consequent can be derived

Summary

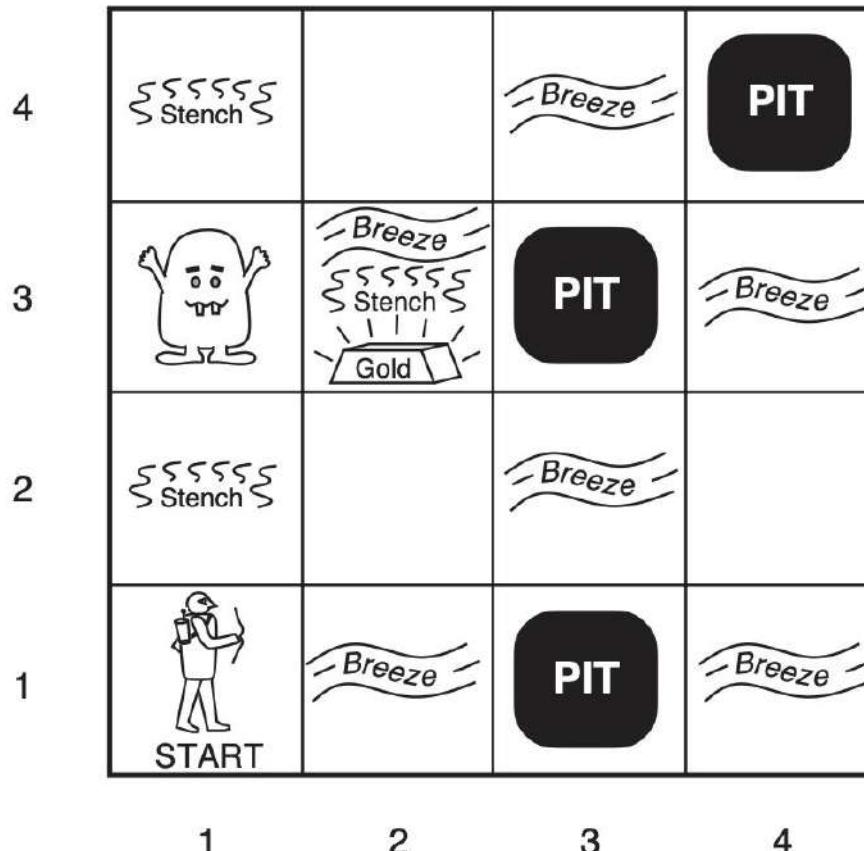
- Different logics make different **commitments** about what the world is made of and what kind of beliefs we can have regarding the facts
- **Propositional logic** commits only to the existence of facts that may or may not be the case in the world being represented
 - It has a simple syntax and simple semantics. It suffices to illustrate the process of inference
 - Propositional logic quickly becomes impractical, even for very small worlds

Logical agents

Three (non-exclusive) agent architectures:

- Reflex agents
 - Have rules that classify situations, specifying how to react to each possible situation
- Model-based agents
 - Construct an internal model of their world
- Goal-based agents
 - Form goals and try to achieve them

Wumpus World



A simple reflex agent

- Rules to map percepts into observations:

$$\forall b,g,u,c,t \text{ Percept}([Stench, b, g, u, c], t) \rightarrow Stench(t)$$
$$\forall s,g,u,c,t \text{ Percept}([s, Breeze, g, u, c], t) \rightarrow Breeze(t)$$
$$\forall s,b,u,c,t \text{ Percept}([s, b, Glitter, u, c], t) \rightarrow AtGold(t)$$

- Rules to select an action given observations:

$$\forall t AtGold(t) \rightarrow Action(Grab, t);$$

- Some difficulties:

- **Consider Climb.** There is no percept that indicates the agent should climb out – position and holding gold are not part of the percept sequence

- **Loops** – the percept will be repeated when you return to a square, which should cause the same response (unless we maintain some internal model of the world)

The “Hunt the Wumpus” agent

- Some atomic propositions:

S_{12} = There is a stench in cell (1,2)

B_{34} = There is a breeze in cell (3,4)

W_{22} = The Wumpus is in cell (2,2)

V_{11} = We have visited cell (1,1)

OK_{11} = Cell (1,1) is safe.

etc

- Some rules:

$$(R1) \neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$$

$$(R2) \neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$$

$$(R3) \neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$$

$$(R4) S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$$

etc

After the third move

- We can prove that the Wumpus is in (1,3) using the four rules given.

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

Proving W13

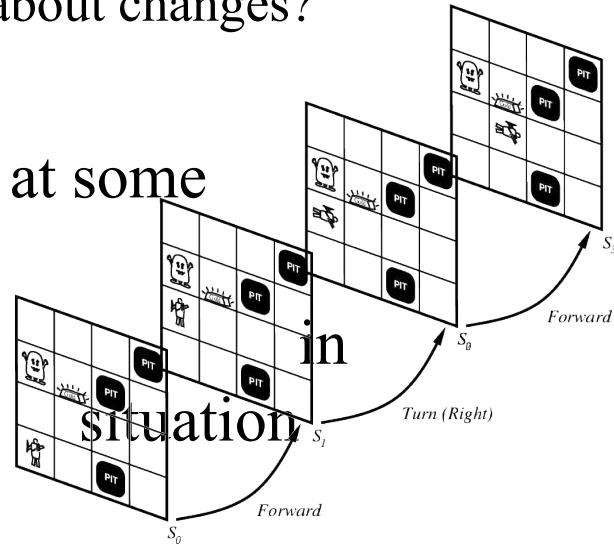
- Apply MP with $\neg S11$ and R1:
 $\neg W11 \wedge \neg W12 \wedge \neg W21$
- Apply And-Elimination to this, yielding 3 sentences:
 $\neg W11, \neg W12, \neg W21$
- Apply MP to $\neg S21$ and R2, then apply And-elimination:
 $\neg W22, \neg W21, \neg W31$
- Apply MP to $S12$ and R4 to obtain:
 $W13 \vee W12 \vee W22 \vee W11$
- Apply Unit resolution on $(W13 \vee W12 \vee W22 \vee W11)$ and $\neg W11$:
 $W13 \vee W12 \vee W22$
- Apply Unit Resolution with $(W13 \vee W12 \vee W22)$ and $\neg W22$:
 $W13 \vee W12$
- Apply UR with $(W13 \vee W12)$ and $\neg W12$:
 $W13$

Problems with the propositional Wumpus hunter

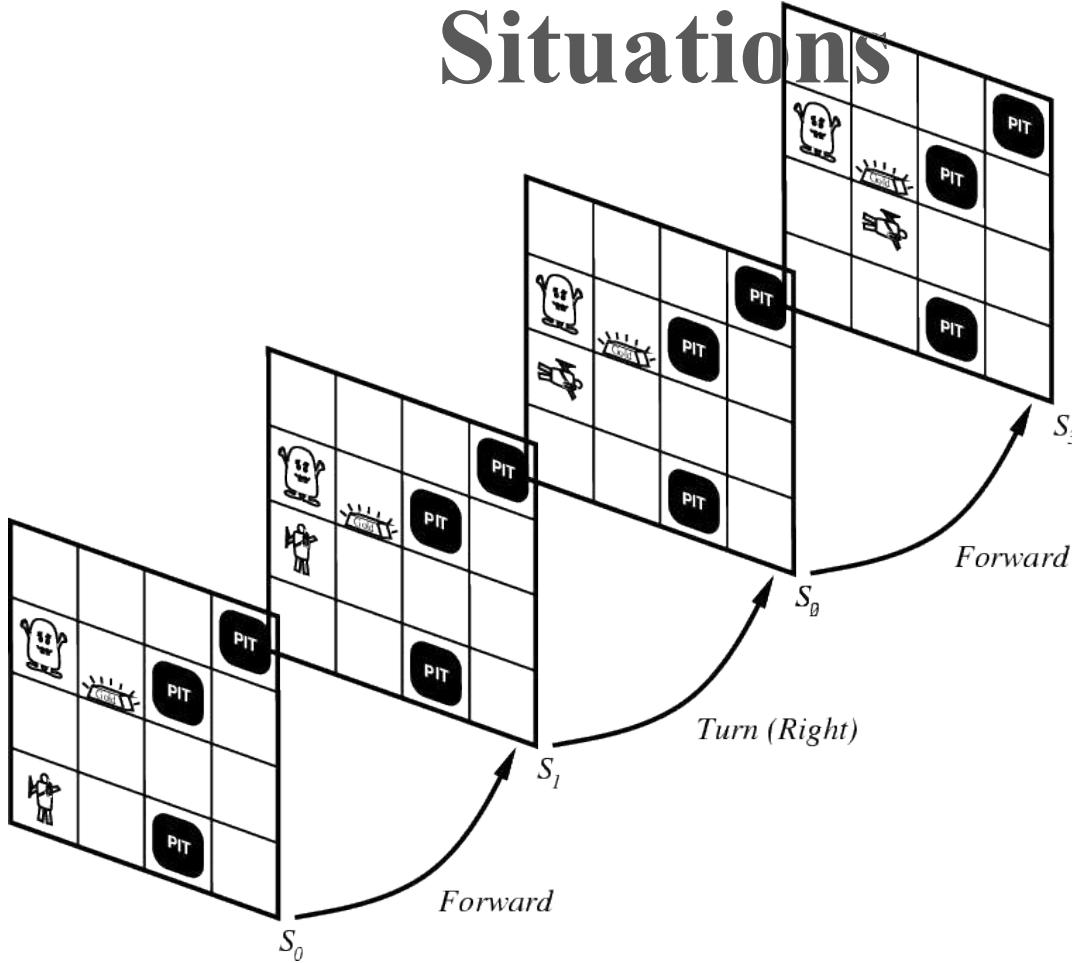
- Lack of variables prevents stating more general rules
 - We need a set of similar rules for each cell
- Change of the KB over time is difficult to represent
 - Standard technique is to index facts with the time when they're true
 - This means we have a separate KB for every time point

Representing change

- Representing change in the world in logic can be tricky.
- One way is just to change the KB
 - Add and delete sentences from the KB to reflect changes
 - How do we remember the past, or reason about changes?
- **Situation calculus** is another way
- A **situation** is a snapshot of the world at some instant in time
- When the agent performs an action A in situation S1, the result is a new S2.



Situations



Situation calculus

- A **situation** is a snapshot of the world at an interval of time during which nothing changes
- Every true or false statement is made with respect to a particular situation.
 - Add **situation variables** to every predicate.
 - $\text{at}(\text{Agent}, 1, 1)$ becomes $\text{at}(\text{Agent}, 1, 1, s_0)$: $\text{at}(\text{Agent}, 1, 1)$ is true in situation (i.e., state) s_0 .
 - Alternatively, add a special 2nd-order predicate, **holds(f,s)**, that means “f is true in situation s.” E.g., $\text{holds}(\text{at}(\text{Agent}, 1, 1), s_0)$
- Add a new function, **result(a,s)**, that maps a situation s into a new situation as a result of performing action a. For example, $\text{result}(\text{forward}, s)$ is a function that returns the successor state (situation) to s
- Example: The action agent-walks-to-location-y

Qualification problem

- Qualification problem:
 - How can you possibly characterize every single effect of an action, or every single exception that might occur?
 - When I put my bread into the toaster, and push the button, it will become toasted after two minutes, unless...
 - The toaster is broken, or...
 - The power is out, or...
 - I blow a fuse, or...
 - A neutron bomb explodes nearby and fries all electrical components, or...
 - A meteor strikes the earth, and the world we know it ceases to exist, or...

Ramification problem

- Similarly, it's just about impossible to characterize every side effect of every action, at every possible level of detail:
 - When I put my bread into the toaster, and push the button, the bread will become toasted after two minutes, and...
 - The crumbs that fall off the bread onto the bottom of the toaster over tray will also become toasted, and...
 - Some of the aforementioned crumbs will become burnt, and...
 - The outside molecules of the bread will become “toasted,” and...
 - The inside molecules of the bread will remain more “breadlike,” and...
 - The toasting process will release a small amount of humidity into the air because of evaporation, and...
 - The heating elements will become a tiny fraction more likely to burn out the next time I use the toaster, and...
 - The electricity meter in the house will move up slightly, and...

Knowledge engineering!

- Modeling the “right” conditions and the “right” effects at the “right” level of abstraction is very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
- Many researchers hope that automated knowledge acquisition and machine learning tools can fill the gap:
 - Our intelligent systems should be able to **learn** about the conditions and effects, just like we do!
 - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!

Preferences among actions

- A problem with Wumpus world knowledge base that we have built so far is that it is difficult to decide which action is best among a number of possibilities.
- For example, to decide between a forward and a grab, axioms describing when it is OK to move to a square would have to mention glitter.
- This is not modular!
- We can solve this problem by **separating facts about actions from facts about goals**. This way our agent can be reprogrammed just by asking it to achieve different goals.

Preferences among actions

- The first step is to describe the desirability of actions independent of each other.

- In doing this we will use a simple scale: actions can be Great, Good, Medium, Risky, or Deadly.

- Obviously, the agent should always do the best action it can find:

$$(\forall a,s) \text{Great}(a,s) \rightarrow \text{Action}(a,s)$$

$$(\forall a,s) \text{Good}(a,s) \wedge \neg(\exists b) \text{Great}(b,s) \rightarrow \text{Action}(a,s)$$

$$(\forall a,s) \text{Medium}(a,s) \wedge (\neg(\exists b) \text{Great}(b,s) \vee \text{Good}(b,s)) \rightarrow \text{Action}(a,s)$$

...

Preferences among actions

- We use this action quality scale in the following way.
- Until it finds the gold, the basic strategy for our agent is:
 - Great actions include picking up the gold when found and climbing out of the cave with the gold.
 - Good actions include moving to a square that's OK and hasn't been visited yet.
 - Medium actions include moving to a square that is OK and has already been visited.
 - Risky actions include moving to a square that is not known to be deadly or OK.
 - Deadly actions are moving into a square that is known to have a pit or a Wumpus.



The image shows three wooden blocks standing upright on a light-colored wooden surface. The blocks are arranged side-by-side to spell out the letters 'Q', '&', and 'A'. The 'Q' and 'A' are on light-colored blocks, while the '&' is on a darker brown block. The background is a soft-focus view of green and yellow foliage, suggesting an outdoor setting.

Q & A



Introduction to Artificial Intelligence

Agents & Environments - Week 3

Agenda

- ▶ Types of Agents
- ▶ Properties of Environments
- ▶ Logics

3

Task

- ▶ Make teams of 4-5 members
- ▶ Select a scenario of an Artificially Intelligent System

PEAS

Performance measure

Environment

Actuators

Sensors

Task

- ▶ Present your selected system on PEAS
- ▶ You have 15 minutes for preparation

Logics - Task

Prove: $\neg S$

Premises:

$$P \wedge Q$$

$$P \rightarrow \neg(Q \wedge R)$$

$$S \rightarrow R$$

Logics - Task

Prove: $\neg S$

Premises:

$$P \wedge Q$$

$$P \rightarrow \neg(Q \wedge R)$$

$$S \rightarrow R$$

Solution:

1. $P \wedge Q$

2. P

3. Q

4. $P \rightarrow \neg(Q \wedge R)$

5. $\neg(Q \wedge R)$

6. $\neg Q \vee \neg R$

7. $\neg R$

8. $S \rightarrow R$

9. $\neg S$

Premise

Decomposing a conjunction (1)

Decomposing a conjunction (1)

Premise

Modus ponens (2, 4)

DeMorgan (5)

Disjunctive syllogism (3, 6)

Premise

Modus tollens (7, 8)



Thank you for
participation



Introduction to Artificial Intelligence

Week 4



2

Expert Systems

ELIZA

- ▶ Program developed at MIT (1964-66) by Joseph Weizenbaum
- ▶ Idea was for it to act as a parity showing the Chinese Room
- ▶ Lead to a number of text adventure computer games
 - ▶ GO LEFT
 - ▶ “you go left and see a gnoll with an axe, what do you do?”
- ▶ Limited success at the Turing test

ELIZA in Action on DOCTOR Script

```
File Edit Options Buffers Tools Help
New Open Save Close Find Replace Undo Redo Cut Copy Paste Delete ? 

I am the psychotherapist. Please, describe your problems.
Each time
you are finished talking, type RET twice.

I have a terrible headache.

Maybe you should consult a doctor of medicine, I am a psychi
atrist.

Okay, I'll do that then.

Can you elaborate on that?

What are you talking about? No I cannot elaborate. See you l
ater.

Why do you say that?

Now you're just talking nonsense!

What makes you believe now I am just talking nonsense?

You are not making any sense.

Perhaps your plans have something to do with this.
-u:*** *doctor*          (Doctor Fill)--L1--Top-----
```

How Does It Work?

- ▶ "I feel like this is an English sentence"
 - ▶ First the string is disassembled based on some rules finding keywords
- ▶ <|> <feel> <this is an English sentence>
 - ▶ Next it is matched to a list of constructions
- ▶ <|> <feel> <this is an English sentence> = <|> <FEEL> <X>
 - ▶ This produced a construction rule
- ▶ <|> <FEEL> <X> ?: - <Why do you feel><X><?>
 - ▶ This rule is applied and response is made
- ▶ Why do you feel like this is an English sentence?
 - ▶ If no rule is found – delay
- ▶ Go on....
Tell me more about <X>

Logic Programming

- ▶ Roots back to 1930s and Alonzo Church with Lambda Calculus
- ▶ Developed in 1950/60s
- ▶ Is a fundamentally differing paradigm than imperative languages (e.g. C, C++, C#, Java, Python, etc.)
 - ▶ Still Turing Complete
 - ▶ Examples include LISP and PROLOG

Statements - Lambda Calculus

Syntax	Name	Description
a	Variable	A character or string representing a parameter or mathematical/logical value
$(\lambda x.M)$	Abstraction	Function definition (M is a lambda term). The variable x becomes bound in the expression
$(M N)$	Application	Applying a function to an argument. M and N are lambda terms

Operations - Lambda Calculus

Operation	Name	Description
$((\lambda x.M) E) \rightarrow (M[x:=E])$	β -reduction	Substituting the bound variable by the argument expression in the body of the abstraction
$(\lambda x.M[x]) \rightarrow (\lambda y.M[y])$	α -conversion	Renaming the bound (formal) variables in the expression. Used to avoid name collisions

Imperative v. Logic Programming

Imperative

- ▶ Statements
- ▶ Code and Data
- ▶ Code by programmer
- ▶ Data as input from programmer/user
- ▶ Actions from User (sometimes)
- ▶ Program build from libraries but many of the units are special purpose

Logic

- ▶ Rules
- ▶ Inference Engine/Knowledge Base
- ▶ Inference Engine built in
- ▶ Knowledge base by programmer/user
- ▶ Queries from User
- ▶ Rapid deployment and prototyping as the language has the inference engine by default

AI Winter

- ▶ Again these systems while being very popular in the 1980s suffered under the AI winter
- ▶ The idea of these systems was that giving a computer all of the knowledge will make it “smart”
- ▶ Legal issues
 - ▶ What if a doctor makes a misdiagnosis

Areas of Application for Expert Systems (Hayes-Roth 1983)

Category	Problem addressed	Examples
Interpretation	Inferring situation descriptions from sensor data	Hearsay (speech recognition), PROSPECTOR
Prediction	Inferring likely consequences of given situations	Preterm Birth Risk Assessment
Diagnosis	Inferring system malfunctions from observables	CADUCEUS, MYCIN , PUFF, Mistral, Eydenet, Kaleidos
Design	Configuring objects under constraints	Dendral , Mortgage Loan Advisor , R1 (DEC VAX Configuration), SID (DEC VAX 9000 CPU)
Planning	Designing actions	Mission Planning for Autonomous Underwater Vehicle
Monitoring	Comparing observations to plan vulnerabilities	REACTOR
Debugging	Providing incremental solutions for complex problems	SAINT, MATHLAB, MACSYMA
Repair	Executing a plan to administer a prescribed remedy	Toxic Spill Crisis Management
Instruction	Diagnosing, assessing, and repairing student behavior	SMH.PAL, Intelligent Clinical Training, STEAMER
Control	Interpreting, predicting, repairing, and monitoring system behaviors	Real Time Process Control, Space Shuttle Mission Control

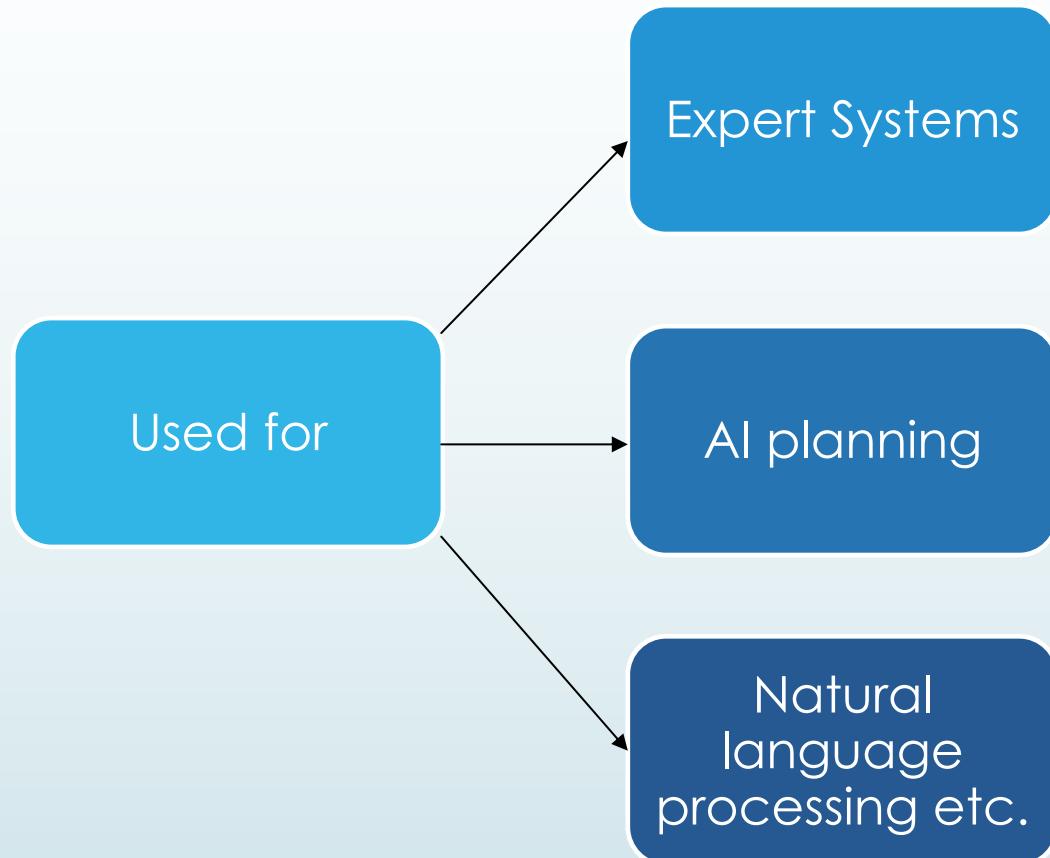
Training an Expert System

- ▶ Interviews with experts
 - ▶ Experts classify knowledge cases
 - ▶ Provide a series of rules which they use to make the classifications
 - ▶ Very costly - knowledge acquisition problem
 - ▶ Experts by definition are rare
 - ▶ Meaning expensive
 - ▶ Why would they want to train the competition?
- ▶ Machine Learning Techniques – especially Decision trees are used to define a pathway to a solution
- ▶ Frontend development and user understanding

Prolog

- ▶ **Prolog** is a general-purpose logic programming language associated with artificial intelligence and computational linguistics
- ▶ Prolog stands for **P**rogramming in **L**ogic
- ▶ Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations
- ▶ The language was first conceived by a group around Alain Colmerauer in Marseille, France, in the early 1970s and the first Prolog system was developed in 1972 by Colmerauer with Philippe Roussel

Prolog Applications



Why to Study?

- ▶ Prolog was one of the first logic programming languages and remains the most popular such language today, with several free and commercial implementations available
- ▶ Currently, not widespread, and not the fastest in terms of execution
- ▶ Very easy to understand and apply
- ▶ Someday its concept can be useful for programming like it happened to functional programming languages

Prolog Semantics

- ▶ Head :- Body.
- ▶ Clauses with empty bodies are called **facts**
cat(tom).
- ▶ Clauses with bodies are called **rules**. Rules describe new rules based on facts and/or other rules
animal(X) :- cat(X).

- ▶ Queries are needed to show outputs. They are also called **goals**
?- cat(X).

Answer: X = tom

Rule Example

- ▶ Facts and Rules together comprises a knowledge base of the program.
Rules consists of facts with added conditions
- ▶ Example: *father(Y,Z) :- man(Y), son(Z,Y).*
 - ▶ If the body of the rule is true then the head is true
 - ▶ “:-” is read as “if”
 - ▶ “,” is logical AND
 - ▶ “;” is logical OR
 - ▶ The statement is read as: Y is a father of Z if Y is a man and Z is a son of Y.
 - ▶ Note that order and semantics of the arguments are specified by programmer
 - ▶ By *father(Y,Z)* we can mean that Y is a father of Z OR Z is father of Y

Data Types

- ▶ Prolog's single data type is the term. There are 4 term types:
 - ▶ Atoms
 - ▶ Numbers (integers and floats)
 - ▶ Variables
 - ▶ Compound terms

Atoms

- ▶ An **atom** is a general-purpose name with no inherent meaning
- ▶ Should begin with a lowercase letter or should be surrounded by single quotes
- ▶ May contain uppercase letters, lowercase letters, digits (0-9), characters (+-*<>=:~)
- ▶ Examples:
 - ▶ cat, 'Dog', 'another cat', my_pet, myPet,

Variables

- ▶ **Variables** consists of letters, digits and underscores. Variables start with uppercase letter or underscore
- ▶ Examples:
 - ▶ X, _Y, Piano, Person_1, DOLL
- ▶ Free variable is the one, for which the value was not assigned, bound is the opposite
 - ▶ ?- 5 = 4 // false
 - ▶ ?- X = 5, X = 4 // false
 - ▶ ?- X = 5 + 4 // it is not a mathematical expression!
 - ▶ ?- X is 5 + 4 // X = 9
 - ▶ ?- X = Y, X = 5, Z is Y + 4 // X = 5, Y = 5, Z = 9
- ▶ Anonymous variable '_' makes the output for that variable invisible
 - ▶ ?- father(X,_). // should return all fathers X without showing names of children

Compound Terms

- ▶ A **compound term** is composed of an atom called a "functor" and a number of "arguments", which are again terms
- ▶ Compound terms are ordinarily written as a functor followed by a comma-separated list of argument terms, which is contained in parentheses
- ▶ The number of arguments is called the term's arity. An atom can be regarded as a compound term with arity zero
- ▶ Examples:
 - ▶ friend(max,peter).
 - ▶ person_friends(zelda,[tom,jim]).

List

- ▶ A **list** is an ordered collection of terms. It is denoted by square brackets with the terms separated by commas
- ▶ It is a compound term type
- ▶ Examples:
 - ▶ [1,2,3]
 - ▶ [1]
 - ▶ []
 - ▶ [red,green,blue].

Strings

- ▶ **String** is a sequence of characters surrounded by double quotes
- ▶ It is equivalent to either a list of (numeric) character codes, a list of characters (atoms of length 1), or an atom depending on the value of the Prolog flag double quotes
- ▶ It is a compound term type
- ▶ Examples:
 - ▶ "to be, or not to be"

Useful Operators

- ▶ Negation: `not(term)` or `\+ term`
 - ▶ `female(X) :- not(male(X)).`
 - ▶ `legal(X) :- \+ illegal(X).`
- ▶ Not equal: `\=`
 - ▶ `X\=Y`
- ▶ Cut: `!`
 - ▶ Takes only the first result
 - ▶ `?- parent(Y,X), !.` // Even if there is more than 1 child with parent, will return only // the first found pair, e.g. X=tom, Y=max

References

1. [Wikipedia](#)
2. [A Taste of Prolog by Aja Hammerly](#)

Introduction to Prolog

**Darko Bozhinoski,
Ph.D. in Computer Science**

What is Prolog?

Theoretical Foundations

Prolog was developed based on formal logic and mathematical theories of computation.

Declarative Language

Prolog is a declarative language, which means you describe the problem and let the computer figure out how to solve it.

Pattern Matching

Prolog uses pattern matching to compare the input with the knowledge base and find solutions to problems.

Basic Concepts of Prolog

Quick Tutorial

```
?- likes(john, jane). ← dot necessary  
true. ← answer from prolog interpreter  
  
sign on prolog query prompt  
variables  
? - friends(X, Y).  
X = john,  
Y = jane ; ← type ; to get next solution  
X = jane,  
Y = john.
```

What is Logic Programming?

- "The use of mathematical logic for computer programming." (Wikipedia)
- "A declarative, relational style of programming based on first-order logic." (Dictionary.com)

What is Prolog?

- Prolog is a logical and a declarative programming language.
- The name itself, Prolog, is short for PROGramming in LOGic.
- Prolog is a Programming Language for symbolic, non-numeric computation.
- Prolog is the major example of a fourth generation programming language.

Syntax and Structure

Prolog programs are composed of facts and rules that follow a specific syntax and structure.

Logic Programming

Logic programming is based on rules, which consist of a head and a body that define relationships between objects.

Interpreters

Prolog programs are run by an interpreter that matches queries to the knowledge base and produces results.

Three Constructs in Prolog

- Facts
- Rules
- Queries

Working with Facts and Rules in Prolog

Facts

Facts are statements that are always true in the knowledge base, which can be used to build further relationships between objects.

Rules

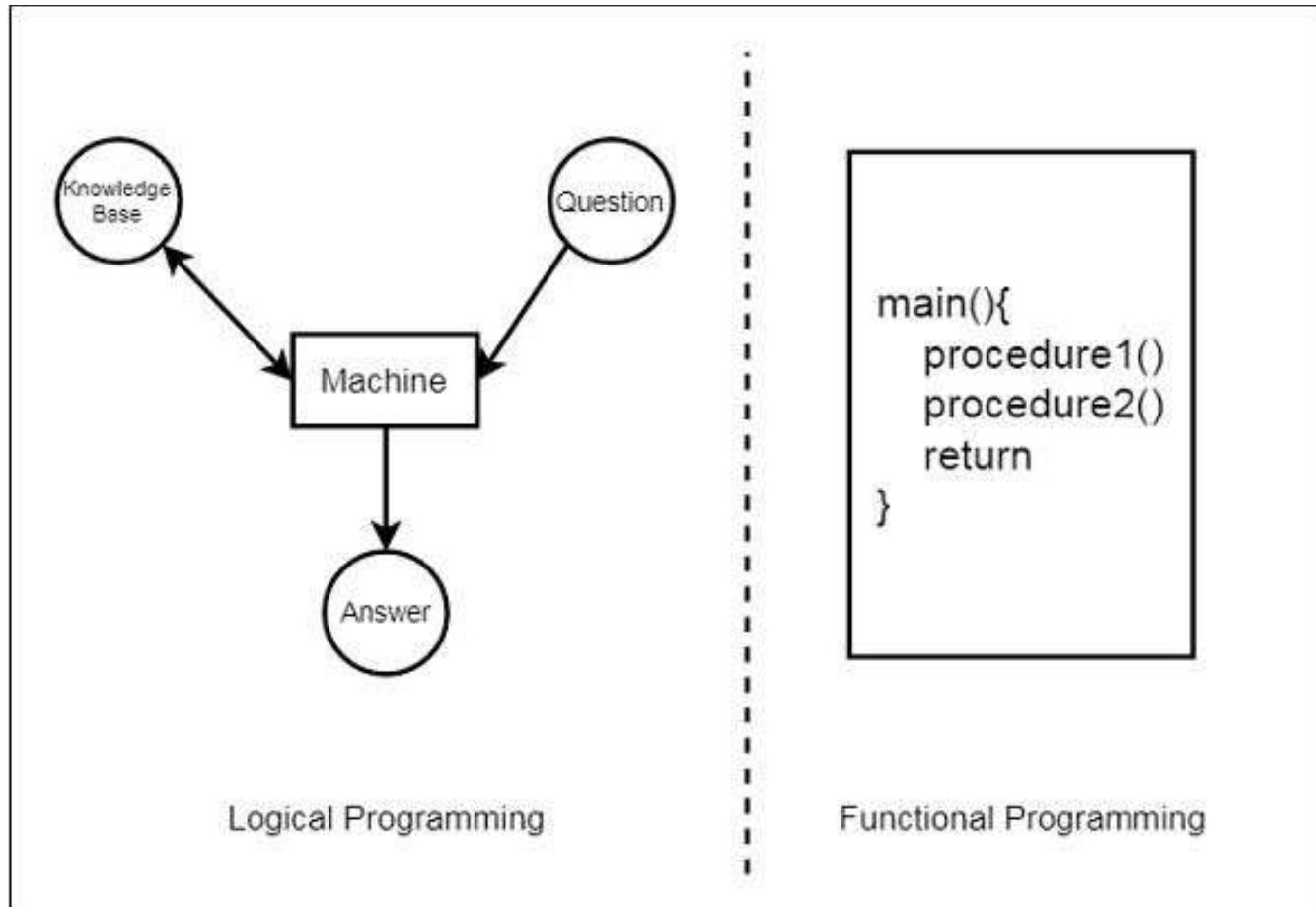
Rules define relationships between objects and can be used to solve problems by querying them with specific input.

KNOWLEDGE BASE

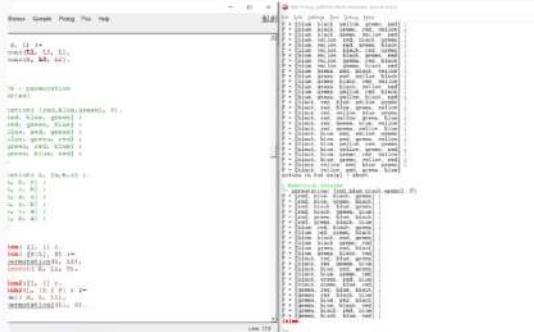
**Facts and Rules
together
comprises a
knowledge base
of the program.**

Queries are used on the knowledge base.

Syntax: Queries end with a “full stop”.



Using Prolog for Logic Programming

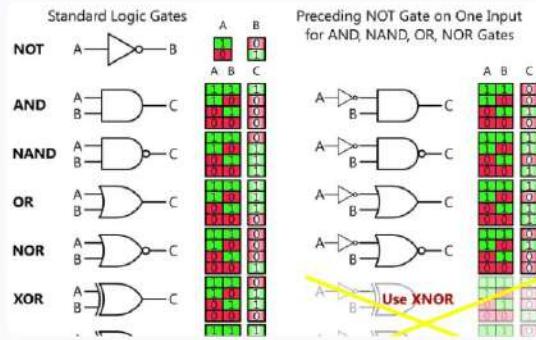


A screenshot of a Prolog development environment. On the left, there is a code editor with Prolog source code. On the right, there is a graphical interface showing a logic truth table for a circuit. The table has columns for inputs A and B, and rows for outputs C and D. The table shows the results of various logic operations.

```
:- module(logic, [and/2, or/2, not/2, nand/2, nor/2, xor/2]).  
  
and(A, B) :- !, !, A = 1, B = 1, C = 1.  
and(A, B) :- !, !, A = 1, B = 0, C = 0.  
and(A, B) :- !, !, A = 0, B = 1, C = 0.  
and(A, B) :- !, !, A = 0, B = 0, C = 0.  
  
or(A, B) :- !, !, A = 1, B = 1, C = 1.  
or(A, B) :- !, !, A = 1, B = 0, C = 1.  
or(A, B) :- !, !, A = 0, B = 1, C = 1.  
or(A, B) :- !, !, A = 0, B = 0, C = 0.  
  
not(A, B) :- !, !, A = 1, B = 0.  
not(A, B) :- !, !, A = 0, B = 1.  
  
nand(A, B) :- !, !, A = 1, B = 1, C = 0.  
nand(A, B) :- !, !, A = 1, B = 0, C = 1.  
nand(A, B) :- !, !, A = 0, B = 1, C = 1.  
nand(A, B) :- !, !, A = 0, B = 0, C = 1.  
  
nor(A, B) :- !, !, A = 1, B = 1, C = 0.  
nor(A, B) :- !, !, A = 1, B = 0, C = 0.  
nor(A, B) :- !, !, A = 0, B = 1, C = 0.  
nor(A, B) :- !, !, A = 0, B = 0, C = 1.  
  
xor(A, B) :- !, !, A = 1, B = 1, C = 0.  
xor(A, B) :- !, !, A = 1, B = 0, C = 1.  
xor(A, B) :- !, !, A = 0, B = 1, C = 1.  
xor(A, B) :- !, !, A = 0, B = 0, C = 0.
```

Rule-Based Systems

Prolog is great for creating rule-based systems that can be used for problem-solving in various fields.



Logic Circuits

Prolog can simulate the behavior of digital logic gates and circuits, which can be used for testing logic designs.



Expert Systems

Prolog is widely used for creating expert systems that can provide automated advice and decision-making.

Disadvantages of Prolog

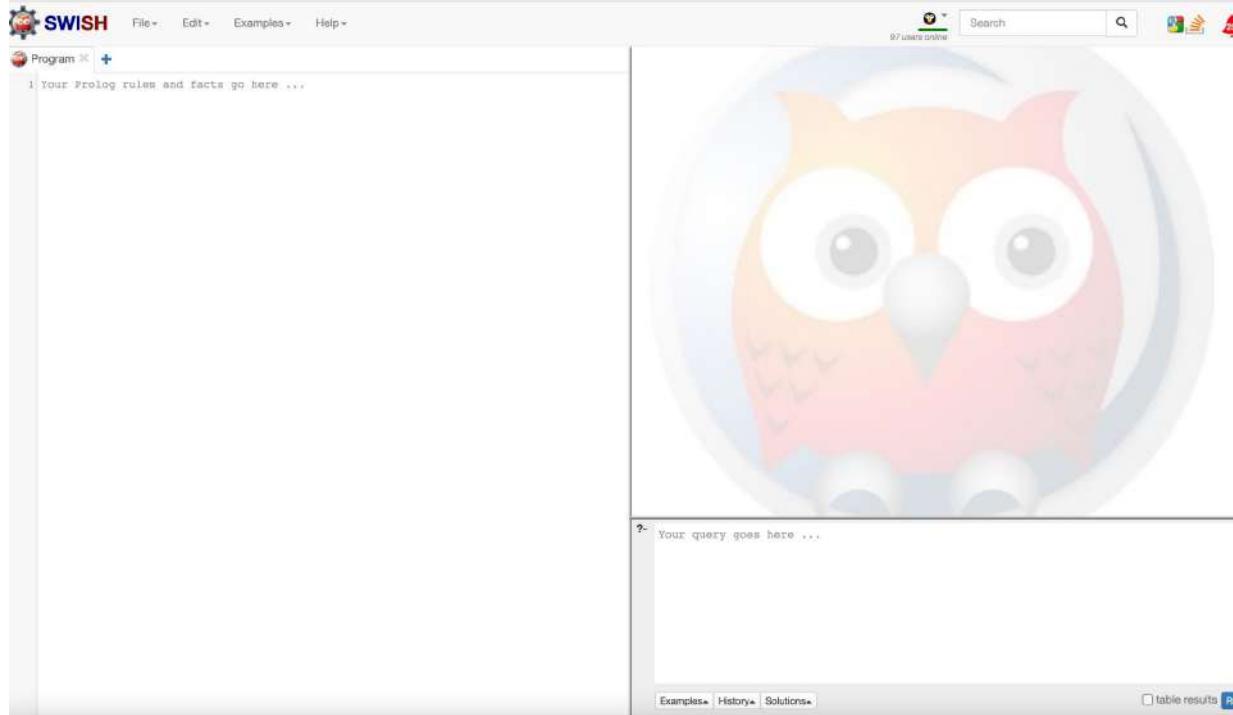
- Prolog does not have good Input/Output features compared to other languages like LISP.
- Prolog cannot handle statements where one of several things is true, but you don't know which.
- Prolog can be less performant than other languages, particularly for tasks that require a lot of computation.

SWI-Prolog

- SWI-Prolog runs on Windows, Macintosh and Linux platforms.
- Can be used online:
<https://swish.swi-prolog.org/>

DEMO

- <https://swish.swi-prolog.org/>



Example

- Lili is happy if she dances.
- Tom is hungry if he is searching for food.
- Andrew and Ilnar are friends if both of them love to play football.
- Ryan will go to play if school is closed, and he is free.

Example

- Lili is happy if she dances.
- Tom is hungry if he is searching for food.
- Andrew and Ilnar are friends if both of them love to play football.
- Ryan will go to play if school is closed, and he is free.

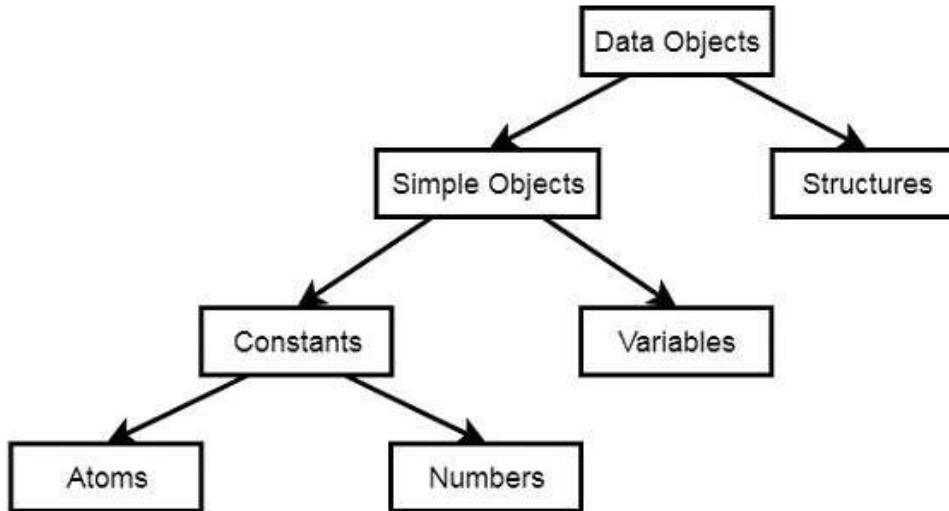
Solution:happy(lili) :- dances(lili).

hungry(tom) :- search_for_food(tom).

friends(Andrew, Ilnar) :- lovesFootball(jack),
lovesFootball(bili).

goToPlay(ryan) :- isClosed(school), free(ryan).

Data Objects in Prolog



Data objects in Prolog

Arithmetic Expressions

The ***is built-in predicate*** is used in Prolog to force the **evaluation** of arithmetic expressions. If you just write something like $X = 2 + 4$, the result is to bind X to the unevaluated term $2 + 4$, not to 6. Example:

```
?- X = 2 + 4.
```

```
X = 2+4
```

If instead you write $X \text{ is } 2 + 4$, Prolog arranges for the second argument, the arithmetic expression $2 + 4$, to be evaluated (giving the result 6) before binding the result to X .

```
?- X is 2 + 4.
```

```
X = 6
```

Comparison Definition	Evaluates?
<code>x = y</code>	succeeds if X and Y <u>unify</u> (match) in the Prolog sense
<code>x \= y</code>	succeeds if X and Y do not unify; i.e. if not <code>(x = y)</code>
<code>T1 == T2</code>	succeeds if terms T1 and T2 are identical; e.g. names of variables have to be the same
<code>T1 \== T2</code>	succeeds if terms T1 and T2 are not identical
<code>E1 =:= E2</code>	succeeds if values of expressions E1 and E2 are equal
<code>E1 =\= E2</code>	succeeds if values of expressions E1 and E2 are not equal
<code>E1 < E2</code>	succeeds if numeric value of expression E1 is < numeric value of E2
<code>E1 =< E2</code>	succeeds if numeric value of expression E1 is \leq numeric value of E2
<code>E1 > E2</code>	succeeds if numeric value of expression E1 is > numeric value of E2
<code>E1 => E2</code>	succeeds if numeric value of expression E1 is \geq numeric value of E2
<code>T1 @< T2</code>	succeeds if T1 is alphabetically < T2
<code>T1 @=< T2</code>	succeeds if T1 is alphabetically \leq T2
<code>T1 @> T2</code>	succeeds if T1 is alphabetically > T2
<code>T1 @>= T2</code>	succeeds if T1 is alphabetically \geq T2

LISTS

- ❖ Lists:
 - []
 - [1, 2, 3]
 - [4, chan]
 - [pen, apple]
 - [[pen, pineapple], [apple, pen]]
- ❖ HEADS | TAILS
 - [1, 2, 3] % 1 - head, [2, 3] - tail
 - [H | T] % "read: H bar T" %
 - [H | T] matches with [1, 2, 3] as [1 | [2, 3]]
- ❖ [1, _, 3]
 - % could be:
 - % [1, 2, 3]
 - % [1, x, 3]
 - % [1, [.....], 3]

Inbuilt predicates in Prolog

- **Identify if an element is part of a list:**
 - `member(a, [a, b, c, d]).`
- **Test for integers in a range:**
 - `maplist(between(1, 3), [1, 2, 3]).`

...

http://www.gprolog.org/manual/html_node/gprolog024.html

Exercise I

Write a program that compares two numbers.

Exercise I - Solution

Write a program that compares two numbers.

Solution 1.

```
find_max(X, Y, X) :- X >= Y.  
find_max(X, Y, Y) :- X < Y.
```

Solution 2.

```
gte(X,Y) :- X > Y, write('X is greater').  
gte(X,Y) :- X =:= Y, write('X and Y are same').  
gte(X,Y) :- X < Y, write('X is smaller').
```

Exercise I - Extension

Extend the program to a program that recursively compares each element in the list to find the maximum value.

Exercise I - Extension Solution

Extend the program to a program that recursively compares each element in the list to find the maximum value.

```
max_list([X], X).
```

```
max_list([H|T], Max) :-  
    max_list(T, RestMax),  
    Max is max(H, RestMax).
```

Exercise II

Write a program that checks if two lists are equal.

Exercise II - Solution

Write a program that checks if two lists are equal.

Solution:

is_equal([],[]).

is_equal([H1|T1],[H2|T2]):- H1=:H2,is_equal(T1,T2).

Inbuilt predicate: You can use compare(=,List1, List2)

Exercise III: Factorial Checker

- Write the program to check if the factorial of the first value is equal to the second value
- Expected outputs:
 - `?- factorial(3,6). // true`
 - `?- factorial(3,5). // false`

Exercise III: Factorial Checker Solution

Solution.

factorial(0, 1).

*factorial(N,F) :-
 N > 0,
 N1 is N - 1,
 factorial(N1,F1),
 F is N * F1.*

Exercise IV: Sum of elements

Define a program `sum_list` that takes a list of integers as its first argument and returns the sum of all the integers in the list as its second argument.

Exercise IV: Sum of elements (Solution)

Define a program sum_list that takes a list of integers as its first argument and returns the sum of all the integers in the list as its second argument.

```
sum_list([], 0).  
sum_list([H|T], Sum) :-  
    sum_list(T, Rest),  
    Sum is H + Rest.
```

Exercise V: Palindrome

Define a program `palindrome` that takes a list as its argument and returns true if the list is a palindrome (i.e., reads the same forwards and backwards).

Solution V: Palindrome

Define a program `palindrome` that takes a list as its argument and returns true if the list is a palindrome (i.e., reads the same forwards and backwards).

Solution: We define `reverse_list` that takes a list as its first argument and returns the reversed list as its second argument.

```
reverse_list([], []).  
reverse_list([H|T], Reversed) :-  
    reverse_list(T, Rest),  
    append(Rest, [H], Reversed).  
  
palindrome(List) :-  
    reverse_list(List, List).
```

Exercise VI: Sorting algorithm

While sorting algorithms can be implemented in Prolog, they are generally more complex and less efficient than their functional programming counterparts. Merge sort is a popular sorting algorithm that divides the list into smaller sublists, sorts them recursively, and then merges them back together.

```
merge_sort([], []).  
merge_sort([X], [X]).  
merge_sort(List, Sorted) :-  
    List = [_,_|_],  
    split(List, Left, Right),  
    merge_sort(Left, SortedLeft),  
    merge_sort(Right, SortedRight),  
    merge(SortedLeft, SortedRight, Sorted).  
  
split([], [], []).  
split([X], [X], []).  
split([X,Y|Rest], [X|Left], [Y|Right]) :-  
    split(Rest, Left, Right).  
  
merge([], Right, Right).  
merge(Left, [], Left).  
merge([X|Left], [Y|Right], [X|Sorted]) :-  
    X <= Y,  
    merge(Left, [Y|Right], Sorted).  
merge([X|Left], [Y|Right], [Y|Sorted]) :-  
    X > Y,  
    merge([X|Left], Right, Sorted).
```

Merge-sort implementation in Prolog

```
def mergeSort(arr):  
    if len(arr) > 1:  
        mid = len(arr) // 2  
        leftArr = arr[:mid]  
        rightArr = arr[mid:]  
  
        mergeSort(leftArr)  
        mergeSort(rightArr)  
  
        i = j = k = 0  
  
        while i < len(leftArr) and j < len(rightArr):  
            if leftArr[i] < rightArr[j]:  
                arr[k] = leftArr[i]  
                i += 1  
            else:  
                arr[k] = rightArr[j]  
                j += 1  
            k += 1  
  
        while i < len(leftArr):  
            arr[k] = leftArr[i]  
            i += 1  
            k += 1  
  
        while j < len(rightArr):  
            arr[k] = rightArr[j]  
            j += 1  
            k += 1
```

Merge-sort implementation in Python

Logic Puzzle VII: The Problem

1. Adam doesn't live on the top floor
2. Bill doesn't live on the bottom floor
3. Claire doesn't live on either the top or the bottom floor
4. David lives on a higher floor than Bill's
5. Eric doesn't live on a floor adjacent to Claire's
6. Claire doesn't live on a floor adjacent to Bill's

Logic Puzzle VII: The Solution

```
1 higher(X, Y, [X | T]) :- member(Y, T).
2 higher(X, Y, [_ | T]) :- higher(X, Y, T).
3 not_adjacent(X, Y, [X, Z | T]) :- 
4     Z \== Y,
5     member(Y, T).
6 not_adjacent(X, Y, [Y, Z | T]) :- 
7     Z \== X,
8     member(X, T).
9 not_adjacent(X, Y, [_ | T]) :- 
10    not_adjacent(X, Y, T).
11 puzzle(L) :- 
12    % Unpack L into local Variables Top, F4, ...
13    L = [Top, _, _, _, Bottom],
14    % `there exists permutaion of L and [...]` 
15    permutation(L, [adam, bill, claire, david, eric]),
16    adam \== Top,
17    bill \== Bottom,
18    claire \== Top, claire \== Bottom,
19    higher(david, bill, L),
20    not_adjacent(eric, claire, L),
21    not_adjacent(claire, bill, L).
22
23
```



A screenshot of a logic programming interface showing the solution to the puzzle. At the top, there is a small icon of a person's head and shoulders. Below it, the command `puzzle([A, B, C, D, E]).` is entered. To the right, the variables are defined: **A** = david, **B** = claire, **C** = adam, **D** = bill, and **E** = eric. At the bottom, there is a button bar with "Next", "10", "100", "1,000", and "Stop". In the bottom right corner, the query `?- puzzle([A, B, C, D, E]).` is displayed.



The image shows three wooden blocks standing upright on a light-colored wooden surface. The blocks are arranged side-by-side to spell out the letters 'Q', '&', and 'A'. The 'Q' and 'A' are on light-colored blocks, while the '&' is on a darker brown block. The background is a soft-focus view of green and yellow foliage, suggesting an outdoor setting.

Q & A



Introduction to Artificial Intelligence

Week 4

A decorative graphic on the left side of the slide features several thin, curved lines in varying shades of gray and dark blue. A prominent dark blue arrow points from the text '2' towards the main title.

2

Expert Systems – Prolog

Agenda

- Prolog syntax and semantics
- Prolog exercises

Prolog

- **Prolog** is a general-purpose logic programming language associated with artificial intelligence and computational linguistics
- Prolog stands for **P**rogramming in **L**ogic
- Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations
- The language was first conceived by a group around Alain Colmerauer in Marseille, France, in the early 1970s and the first Prolog system was developed in 1972 by Colmerauer with Philippe Roussel

Prolog Syntax & Semantics (1)

- Head :- Body.
- Clauses with empty bodies are called facts
***cat(tom).* [Tom is cat]**
- Clauses with bodies are called rules. Rules describe new rules based on facts and/or other rules
***animal(X) :- cat(X).* [X is animal if X is cat]**
- Queries are needed to show outputs. It is also called a goal
?- cat(X).

Answer: X = tom

Prolog Syntax & Semantics (2)

- Rules can be more complex and contain **AND (",")** or **OR (";"**) operators.

animal(X) :- cat(X); dog(X). [X is animal if X is cat OR X is dog]

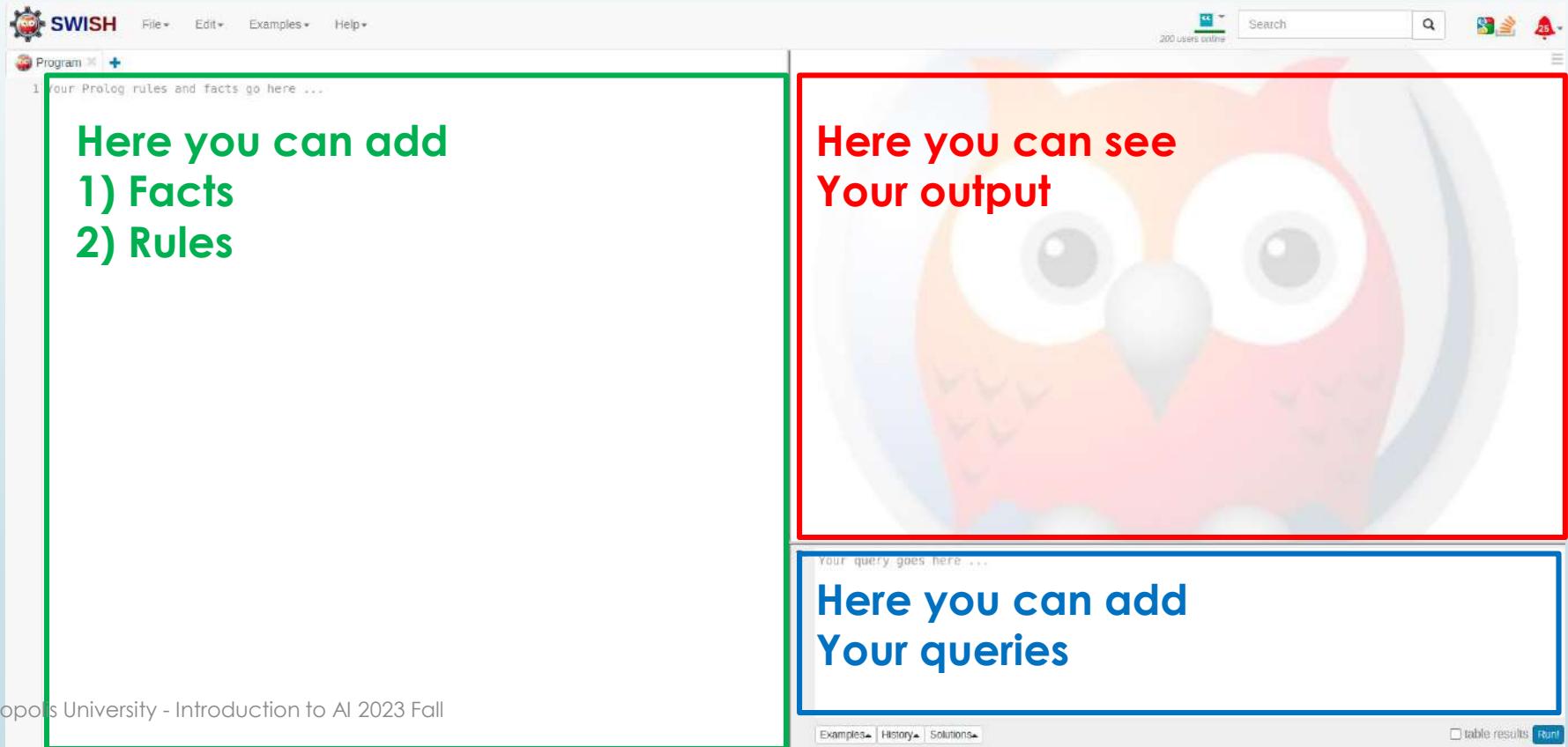
animal(X) :- cat(X), alive(X). [X is animal if X is cat AND X is alive]

- Rules can also contain **NOT ("not")** operator.

pet(X) :- cat(X), not(wild(X)). [X is pet if X is cat AND X is not wild]

SWI-Prolog

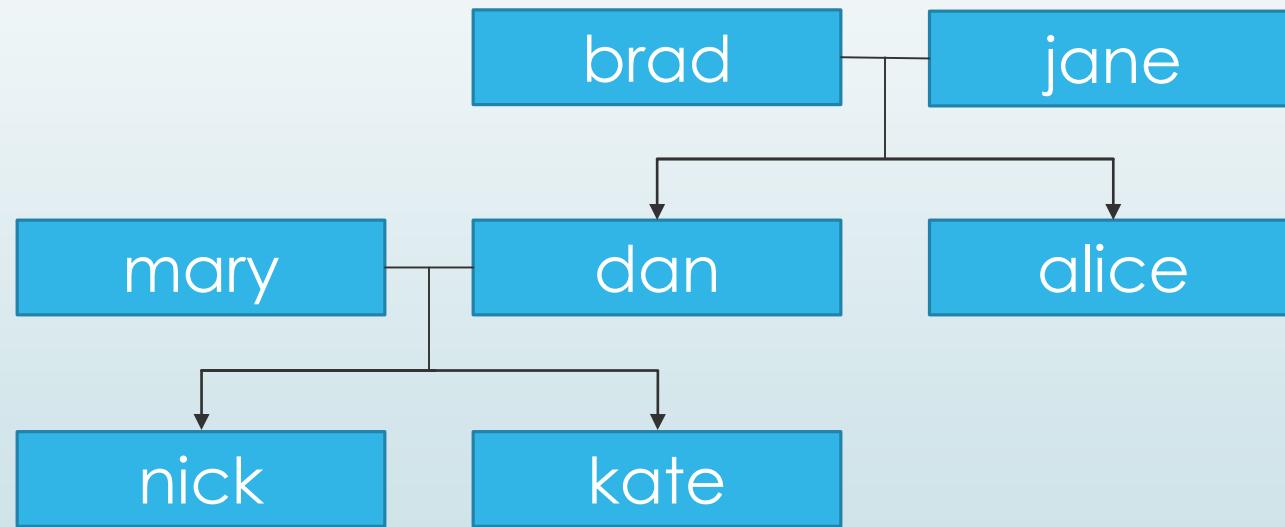
- Go to https://swish.swi-prolog.org/example/prolog_tutorials.swinb
- It will look as following



Exercise 1: Family Tree

1. Based on lecture example draw your own family tree.
2. Write the program with facts and rules for sisters, brothers, mother, father, aunts, uncles, grandfathers and grandmothers. It should be possible to check all these above mentioned relations.
 - Create **facts** with
 $\text{male}(\text{nick})$.
 - Create **rules** for these people like
 $\text{mother}(X,Y) :- \text{parent}(X,Y), \text{female}(X).$
where X is mother, Y is a child.
 - Check **queries** with
 $?- \text{mother}(\text{jane},\text{nick}).$
 - If it is correct, then the output should be **true**. Otherwise, **false**.

Exercise 1: Family Tree



Exercise 1: Family Tree Solution (1/3)

Facts:

male(nick).	parent(dan,nick).
male(dan).	parent(dan,kate).
male(brad).	parent(jane,dan).
female(jane).	parent(brad,dan).
female(alice).	parent(jane,alice).
female(kate).	parent(brad,alice).
female(mary).	parent(mary,nick).
	parent(mary,kate).

Exercise 1: Family Tree Solution (2/3)

Rules:

`father(X,Y) :- parent(X,Y), male(X).`

`mother(X,Y) :- parent(X,Y), female(X).`

`grandfather(X,Y) :- father(X,Z), parent(Z,Y).`

`grandmother(X,Y) :- mother(X,Z), parent(Z,Y).`

`sibling(X,Y) :- father(F,X), father(F,Y), mother(M,X), mother(M,Y), X\=Y.`

`brother(X,Y) :- sibling(X,Y), male(X), X\=Y.`

`sister(X,Y) :- sibling(X,Y), female(X), X\=Y.`

`aunt(X,Y) :- parent(Z,Y), sister(X,Z).`

`uncle(X,Y) :- parent(Z,Y), brother(X,Z).`

Exercise 1: Family Tree Solution (3/3)

- **Goal:** ?- *mother(jane,nick).*
Answer: false
- **Goal:** ?- *father(dan,nick).*
Answer: true
- **Goal:** ?- *parent(jane,X).*
Answer: X = dan
X = alice
- **Goal:** ?- *parent(jane,X), !.*
Answer: X = dan
- **Goal:** ?- *aunt(X,Y).*
Answer: X = alice, Y = nick ;
X = alice, Y = kate
- **Goal:** ?- *aunt(X,_).*
Answer: X = alice

Exercise 2: Ages

Modify previous exercise adding surname, age, height and weight to each person.

- Show all people with your surname
- Show all people with your age
- Show all people with your height
- Show all people with your weight

Exercise 3: Homework

- As previous exercise, we want to get all people with same surname **without adding rules to your code, only facts are used.**

Your name is **nick** if you are a boy.

Your name is **jane** if you are a girl.

- **Hint:**
you can make a good query that your rules will be embedded in.

Bonus Exercise

- Write the program to count from 1 to 10 using recursive call of rule

References

- <https://www.youtube.com/watch?v=0rKD13BIHNQ&t=795s>
- <http://confreaks.tv/videos/cascadiaruby2012-a-taste-of-prolog>



Introduction to Artificial Intelligence

Week 4

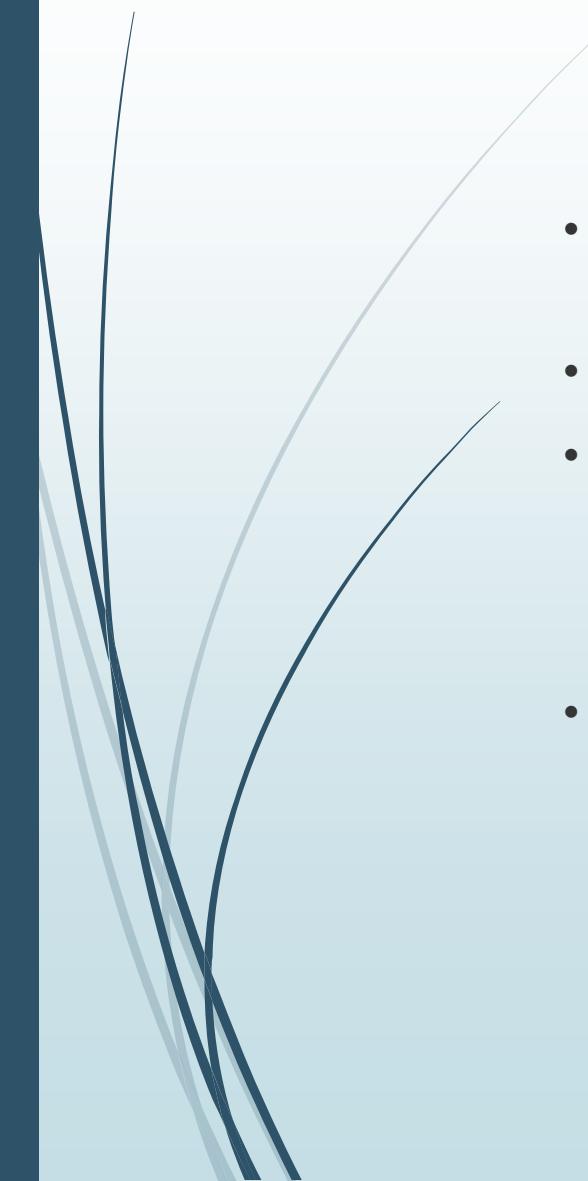
Feedback

SCAN ME





Expert Systems



Prolog

- **Prolog** is a general-purpose logic programming language associated with artificial intelligence and computational linguistics
- Prolog stands for **P**rogramming in **L**ogic
- Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations
- The language was first conceived by a group around Alain Colmerauer in Marseille, France, in the early 1970s and the first Prolog system was developed in 1972 by Colmerauer with Philippe Roussel

Prolog Semantics

- Head :- Body.
- Clauses with empty bodies are called facts
 $\text{cat}(tom).$
- Clauses with bodies are called rules. Rules describe new rules based on facts and/or other rules
 $\text{animal}(X) :- \text{cat}(X).$

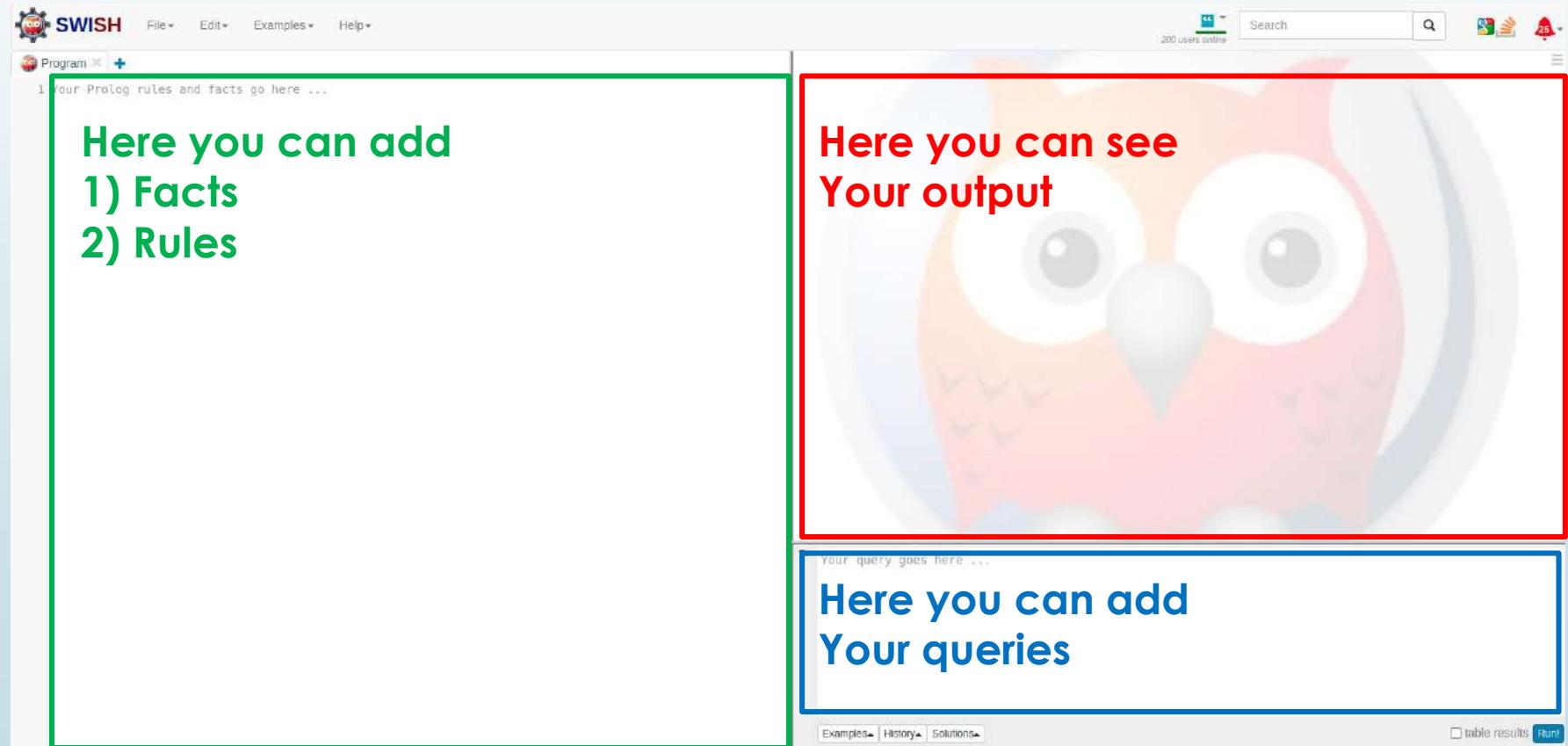
- Queries are needed to show outputs. It is also called a goal

$?- \text{cat}(X).$

Answer: $X = tom$

SWI-Prolog

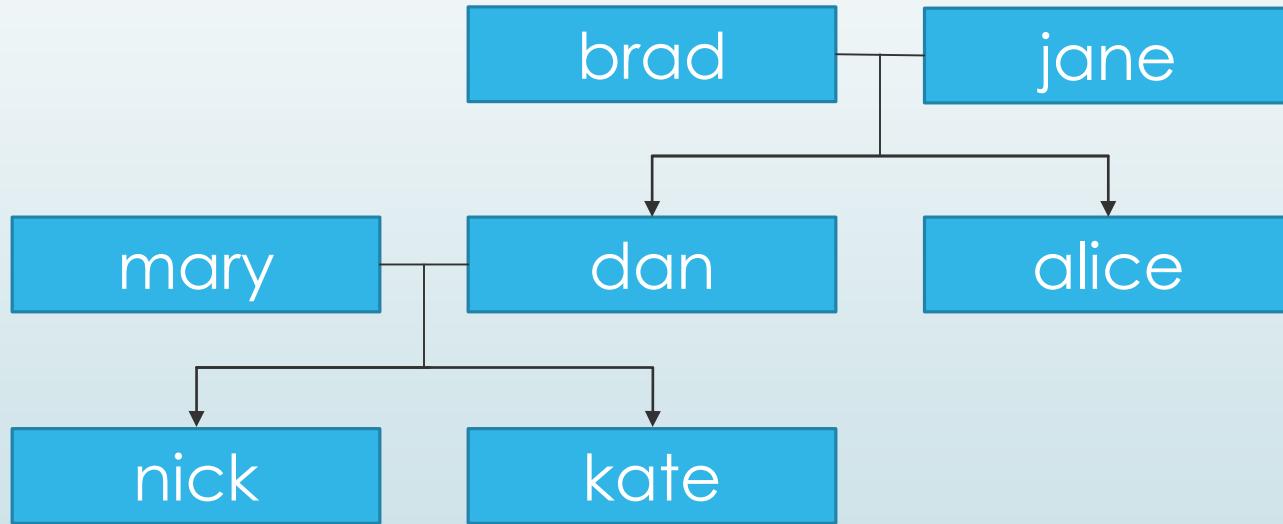
- Go to https://swish.swi-prolog.org/example/prolog_tutorials.swinb
- It will look as following



Exercise 1: Family Tree

1. Based on lecture example draw your own family tree
2. Write the program with facts and rules for sisters, brothers, mother, father, aunts, uncles, grandfathers and grandmothers. It should be possible to check all these abovementioned relations
 - Create **facts** with
 $male(nick)$.
 - Create **rules** for these people like
 $mother(X,Y) :- parent(X,Y), female(X).$
where X is mother, Y is a child.
 - Check **queries** with
 $?- mother(jane,nick).$
 - If it is correct, then the output should be **true**. Otherwise, **false**

Exercise 1: Family Tree



Exercise 1: Family Tree Solution (1/3)

Facts:

male(nick).	parent(dan,nick).
male(dan).	parent(dan,kate).
male(brad).	parent(jane,dan).
female(jane).	parent(brad,dan).
female(alice).	parent(jane,alice).
female(kate).	parent(brad,alice).
female(mary).	parent(mary,nick).
	parent(mary,kate).

Exercise 1: Family Tree Solution (2/3)

Rules:

`father(X,Y) :- parent(X,Y), male(X).`

`mother(X,Y) :- parent(X,Y), female(X).`

`grandfather(X,Y) :- father(X,Z), parent(Z,Y).`

`grandmother(X,Y) :- mother(X,Z), parent(Z,Y).`

`sibling(X,Y) :- father(F,X), father(F,Y), mother(M,X), mother(M,Y), X\=Y.`

`brother(X,Y) :- sibling(X,Y), male(X), X\=Y.`

`sister(X,Y) :- sibling(X,Y), female(X), X\=Y.`

`aunt(X,Y) :- parent(Z,Y), sister(X,Z).`

`uncle(X,Y) :- parent(Z,Y), brother(X,Z).`

Exercise 1: Family Tree Solution (3/3)

- **Goal:** ?- *mother(jane,nick).*
Answer: false
- **Goal:** ?- *father(dan,nick).*
Answer: true
- **Goal:** ?- *parent(jane,X).*
Answer: X = dan
X = alice
- **Goal:** ?- *parent(jane,X), !.*
Answer: X = dan
- **Goal:** ?- *aunt(X,Y).*
Answer: X = alice, Y = nick ;
X = alice, Y = kate
- **Goal:** ?- *aunt(X,_).*
Answer: X = alice

Exercise 2: Ages

Modify previous exercise adding surname, age, height and weight to each person.

- Show all people with your surname
- Show all people with your age
- Show all people with your height
- Show all people with your weight

Exercise 2: Bonus Question

- As previous exercise, we want to get all people with same surname **without adding rules to your code, only facts are used.**

Your name is **nick** if you are a boy.

Your name is **jane** if you are a girl.

- **Hint:**
you can make a good query that your rules will be embedded in.



Exercise 3: Counter

- Write the program to count from 1 to 10 using recursive call of rule



References

- <https://www.youtube.com/watch?v=0rKD13BIHNQ&t=795s>
- <http://confreaks.tv/videos/cascadiaruby2012-a-taste-of-prolog>



Introduction to Artificial Intelligence

Week 5



2

Learning by Searching

Part I

Your Car is Lost in a Carpark

- ▶ How would you find it?
- ▶ Walk at random
 - ▶ No guarantee you will find it
 - ▶ Could be lost forever
- ▶ Walk down each row
 - ▶ Guarantee you will find it
 - ▶ Might take a long time
- ▶ Walk to the place you think it was last, then expand your search
 - ▶ Guarantee you will find it
 - ▶ As long as your memory is good – on average it will be less than every row, worst case just as good
- ▶ Walk to the place you think it was last, hit the alarm button, then expand search
 - ▶ Guarantee you will find it
 - ▶ Alarm can be heard from a long way away, and if you press it to no effect the car is not local to you (radius of activation)



"Here ya go. I'm always losing mine
in big parking lots, too."

Search Space

- ▶ Many problems define a set of actions on an environment which can be seen as a search space of states
 - ▶ We are at one point in the environment with a destination, there is a list of action states which will bring us to the destination
- ▶ Each search type is a method of navigation of the space
- ▶ We will rarely want to go to the same point in the search space
 - ▶ Represents a cycle in the search, which are not useful in terms of state spaces
- ▶ Solution – most searches will avoid this by using tree structures
 - ▶ Each node in the tree is a state within the search
 - ▶ Search Trees, Game Trees

Types of Searches

Uninformed Search

- ▶ Have no sense of the problem domain
- ▶ Generally applicable in all cases

Informed Search

- ▶ Use a heuristic function developed for the domain
- ▶ Applicable in their own domain

Goal-based Agent

- ▶ Four Steps in the Agent:
 1. Define Goals
 - ▶ What are the states which are considered to be satisfying the goal
 2. Problem Formulation
 - ▶ What actions are available to move towards the goal?
 - ▶ What states are available to move towards the goal?
 3. Search
 - ▶ Determine the pathway of states in order to meet with the goal
 4. Execute
 - ▶ Move through the series of states



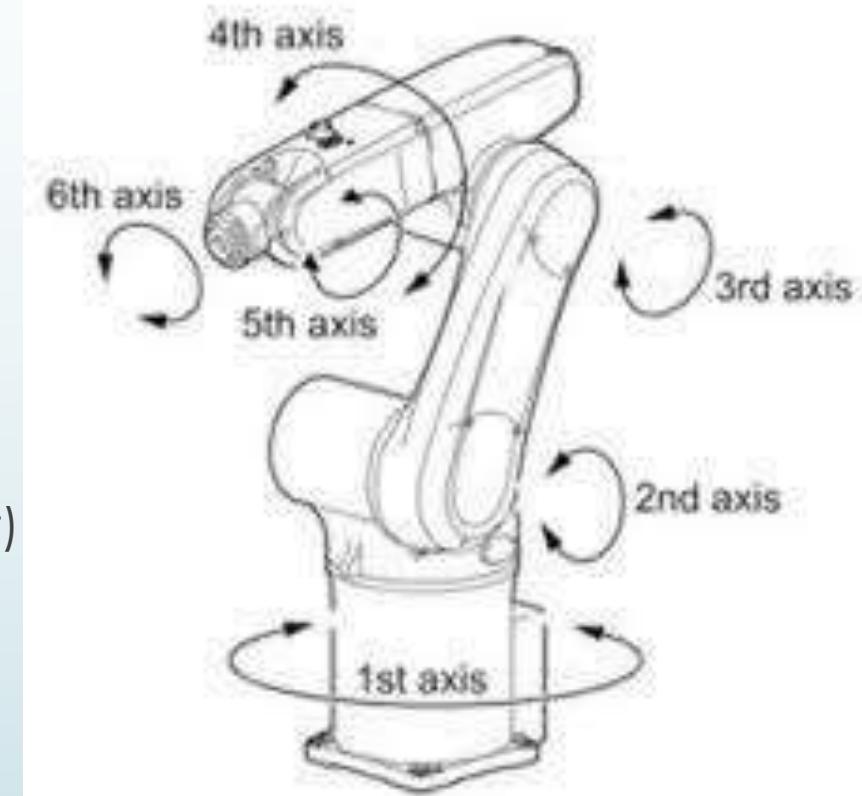
Example: Painting Robot (simplified)

1. Goal
 - ▶ All show surfaces of the part are covered in 1mm of paint
2. Problem Formulation
 - ▶ A sequence of points about the part for the robot arm to move into in order to ensure a dispersal about the entire part
3. Search
 - ▶ Examine all pathways about the part, taking into account the movements on each part of the arm
 - ▶ A pathway is good as long as it covers the part with 1mm of paint
4. Execute
 - ▶ The arm moves through the sequence of positions



States Space of the Robot

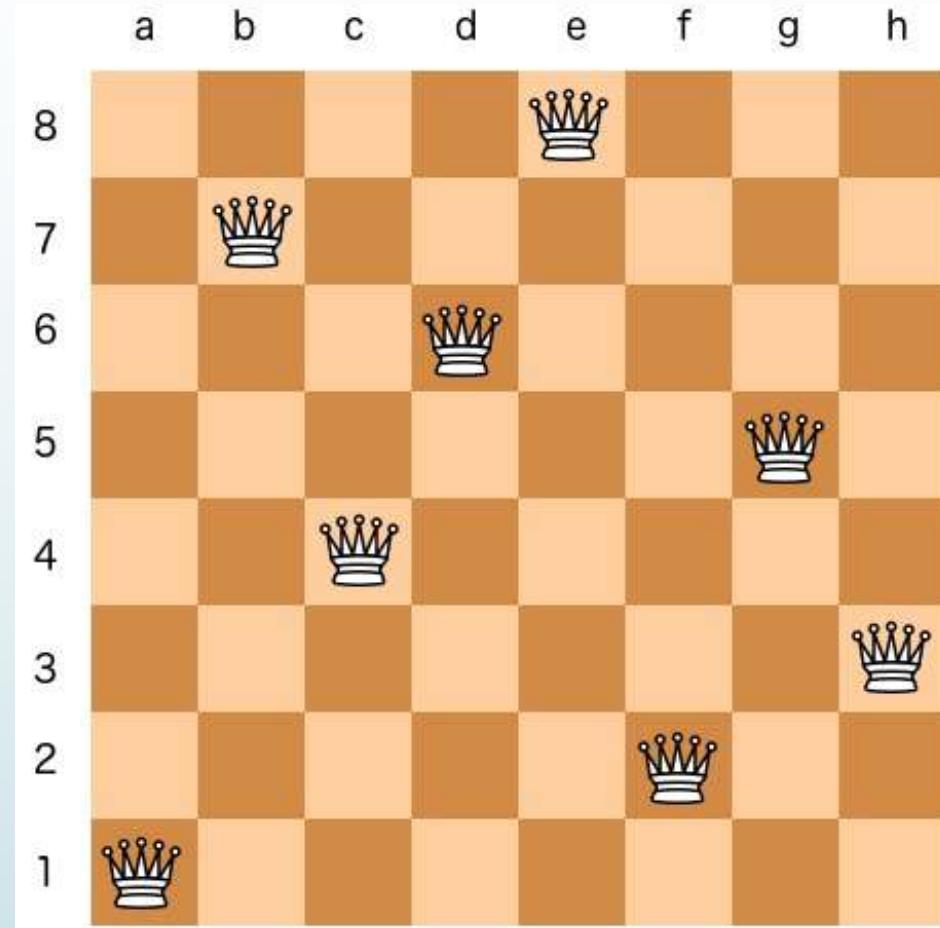
- ▶ State space is defined as the location of all moving elements on the robot
 - ▶ Relative positioning on all joints from a home position
 - ▶ Might also include the rate of spray on the nozzle – from 0 to full
- ▶ Transitions of states could be broken down into a sequence of movements in the joints on each rotational axis (hence a 6-axis robot)
 - ▶ Move(Elbow, X degrees)



N Queens and Incremental Goals

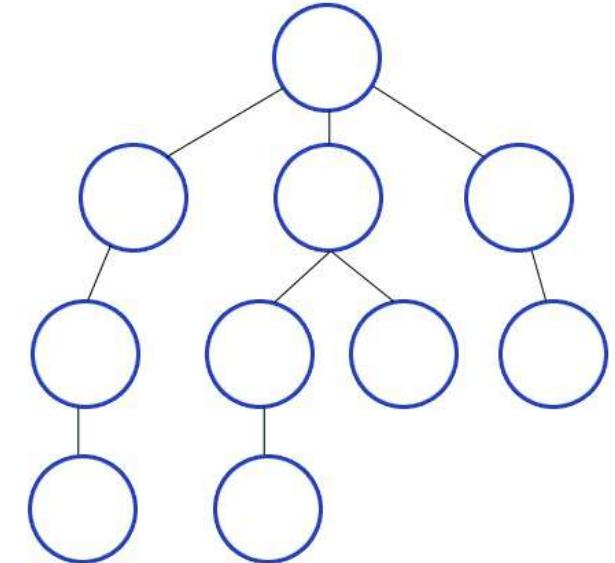


- ▶ Place N queens onto an NxN chess board such that no queen attacks any other queen; a more complicated problem is a Costas Array
- ▶ States
 - ▶ Place a queen down
- ▶ Goal
 - ▶ No queen attacks any other queen
 - ▶ Hard sometimes to see a good pathway to the goal
- ▶ Need a measure in order to see a good pathway
 - ▶ Minimize number of queens attacking



Tree Search

- ▶ Recursive definition of a tree search as DFS
- ▶ `SearchTree(State, Move, visited list)`
 - ▶ New State <- Apply Move
 - ▶ If (New State == goal) – return TRUE
 - ▶ Else
 - ▶ If neighbouring states are empty - return FALSE
 - ▶ For each of the MOVE on neighbouring states not in visited list
 - ▶ If `SearchTree(Neighbouring states, MOVE, visited list + State)` return TRUE;



Backtracking Searches

- ▶ Move towards the goal, if you reach a position which is going to be less successful – stop and move back up the tree (backtrack, retrace your steps)
- ▶ Recursive definition of a depth first search of a backtrack:
- ▶ `SearchTreeBacktrack(State, Move, visited list)`
 - ▶ New State <- Apply Move
 - ▶ If (New State == goal) – return TRUE
 - ▶ If (New State causes an invalid path to goal/or costs too much) – return FALSE
 - ▶ Else
 - ▶ If neighbouring states is empty - return FALSE
 - ▶ For each of the MOVE on neighbouring states not in visited list
 - ▶ If `SearchTreeBacktrack(Neighbouring states, MOVE, visited list + State)` return TRUE;

Sudoku

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

Backtrack on Sudoku

- ▶ IsValid
 - ▶ Check if we have not violated the rules
 - ▶ 1-9 in each row
 - ▶ 1-9 in each column
 - ▶ 1-9 in each box
 - ▶ Choice at each empty box 1-9
 - ▶ Multiple Solutions?

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

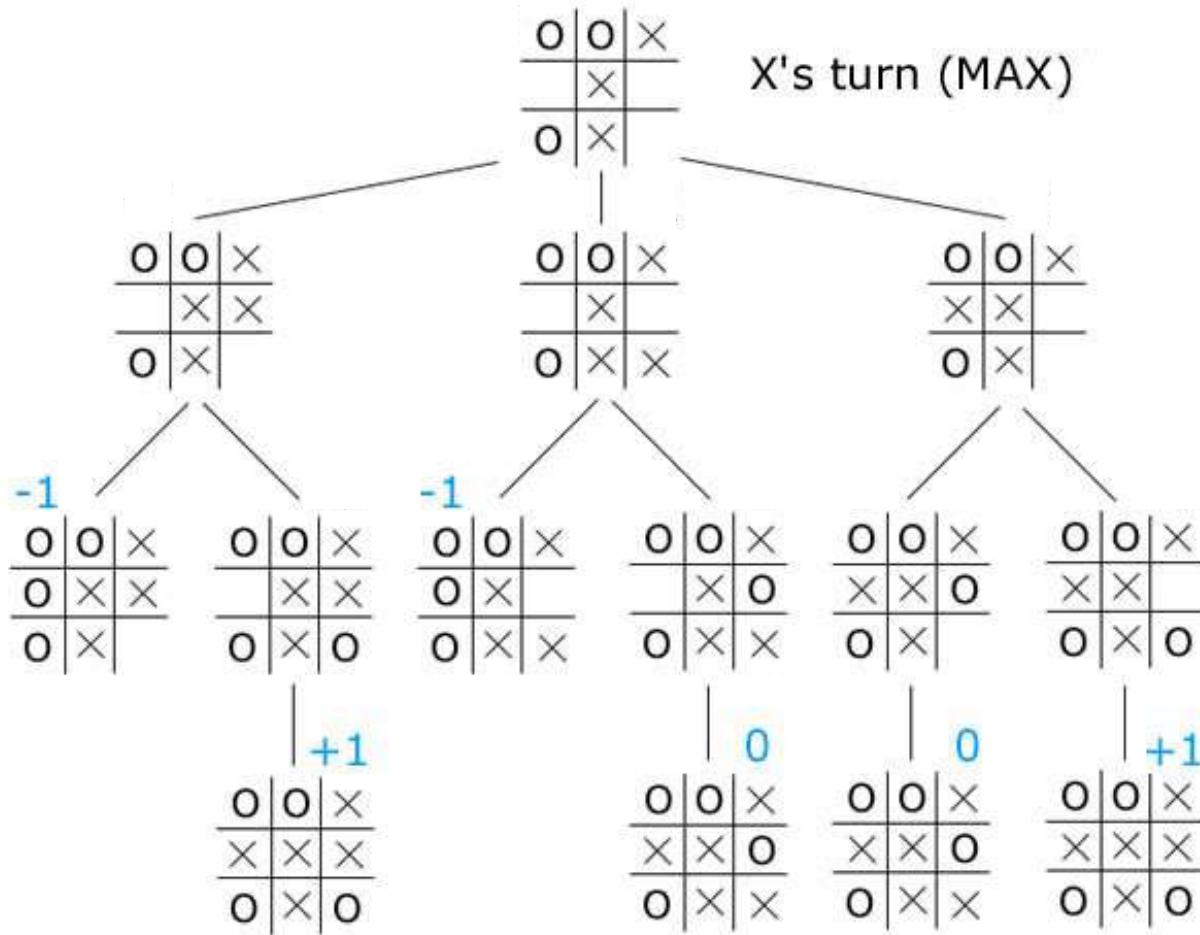
Heuristic Search

- ▶ Word comes from the Greek for discovery
- ▶ Heuristics use an educated guess on where to move the search next – rules-based movement about a search space
- ▶ Not necessarily deterministic – may have stochastic generation methods
- ▶ Trial and Error
 - ▶ Experimental and Experiential
 - ▶ Does not guarantee the most optimal solution
- ▶ Examples
 - ▶ Greedy and Hill Climbers are heuristics

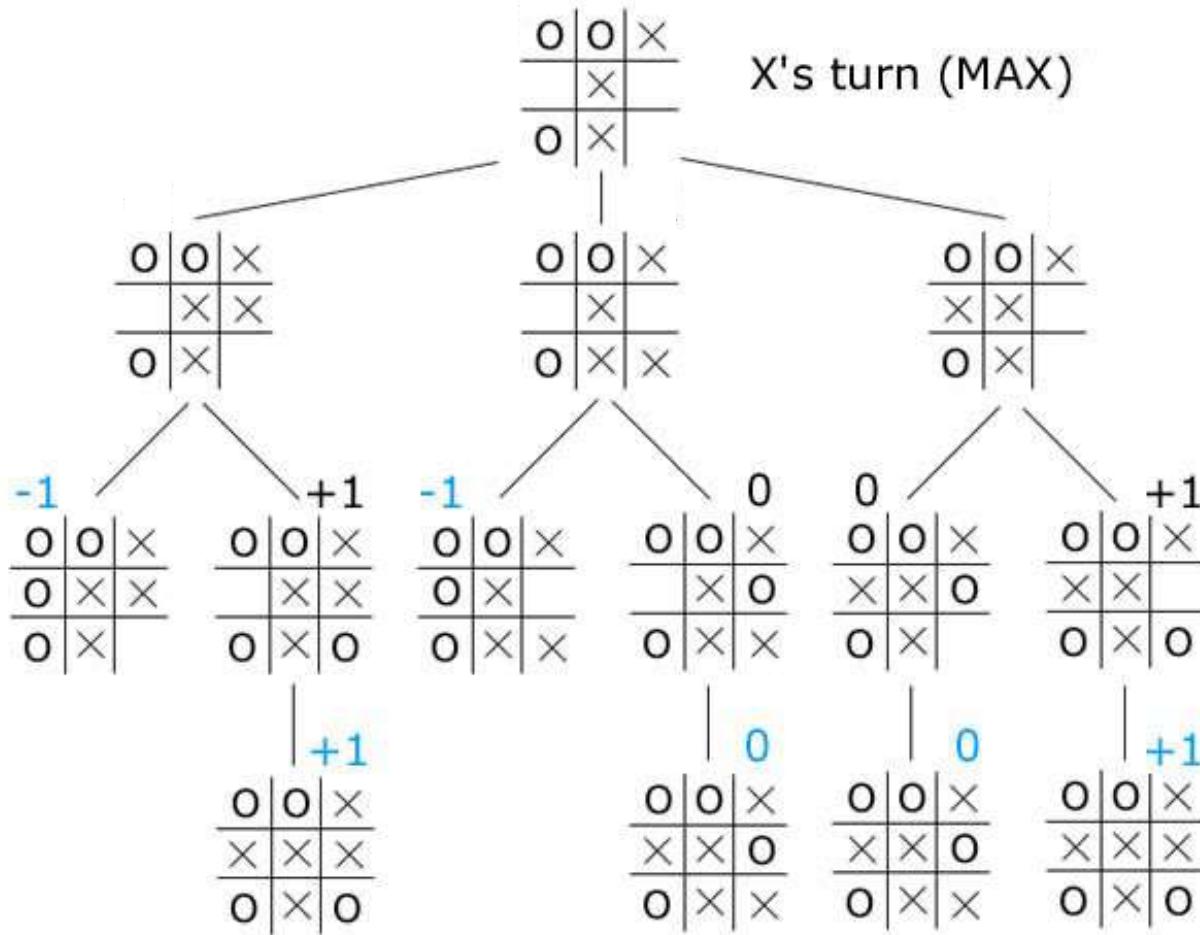
Minimax

- ▶ Opponent goal is to reduce winnings (MINI-Move) and player goal is to increase winnings (MAX-Move)
- ▶ Game with a scoring system or win/lose condition which can be evaluated
 - ▶ GO
 - ▶ Chess
- ▶ Game Tree
 - ▶ Selection of plays in a turn-based game can be reduced to looking at a pathway on a tree
 - ▶ Used as part of the theoretical foundations of GAME THEORY

Minimax Example



Minimax Example

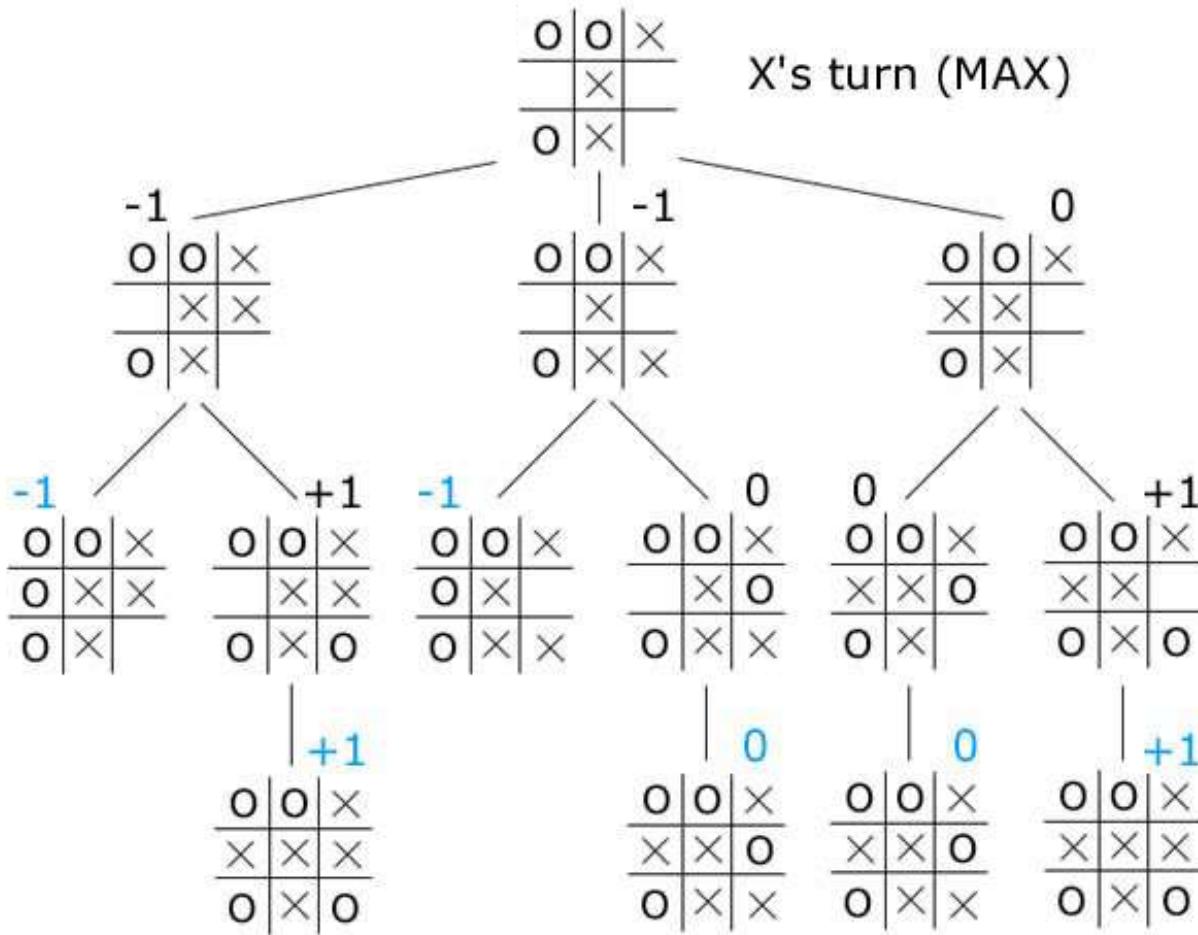


X's turn (MAX)

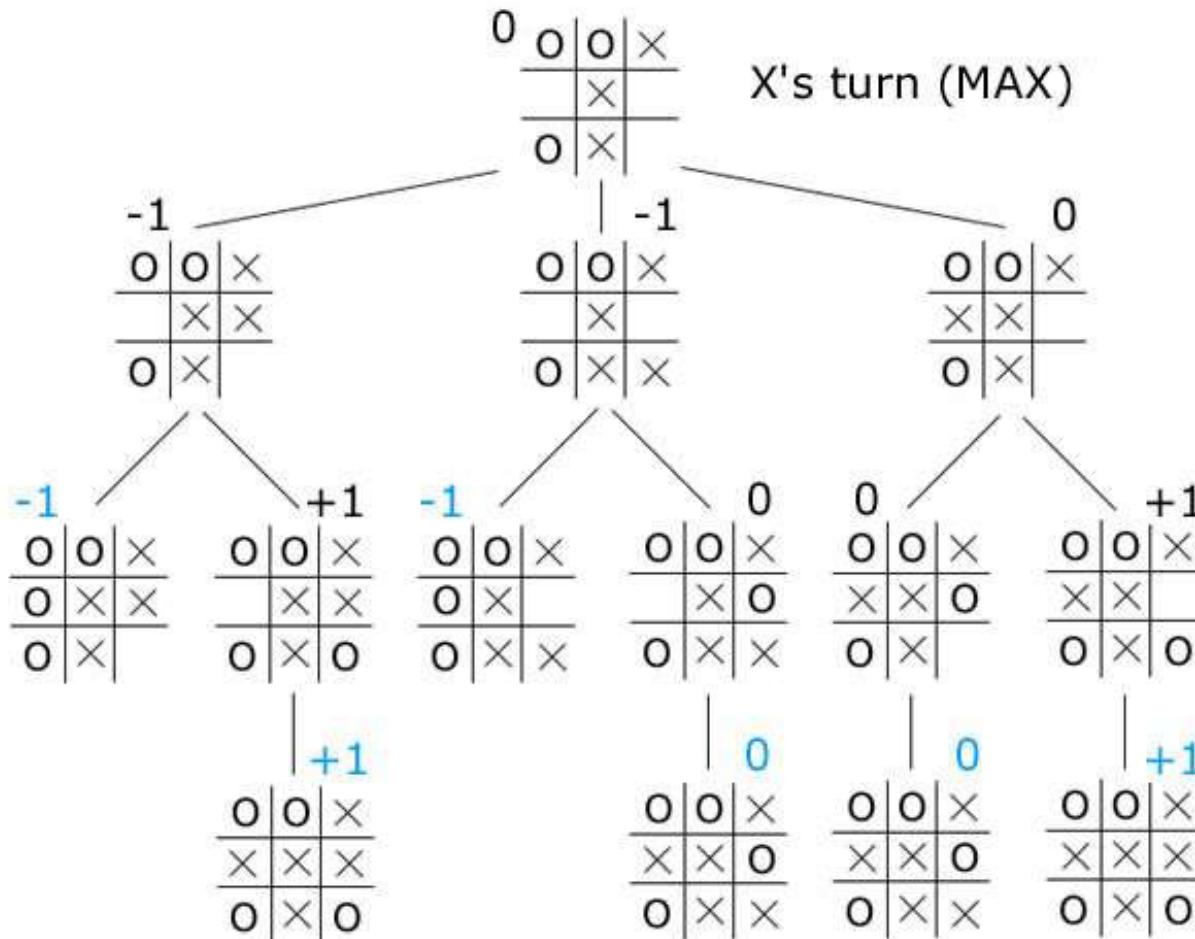
O's turn (MIN)

X's turn (MAX)

Minimax Example



Minimax Example



Backtracking

- ▶ Simulate the entire game
- ▶ Assume rational/perfect play from the opponent
- ▶ Theoretically there is a perfect game play method if we search the entire game
- ▶ Solved Games
 - ▶ Tic Tac Toe
 - ▶ Checkers
 - ▶ Two player heads-up limit poker – Recent discovery by researchers in University of Alberta
- ▶ Unsolved
 - ▶ Chess
 - ▶ Go

Backtracking with Alpha Beta Cuts

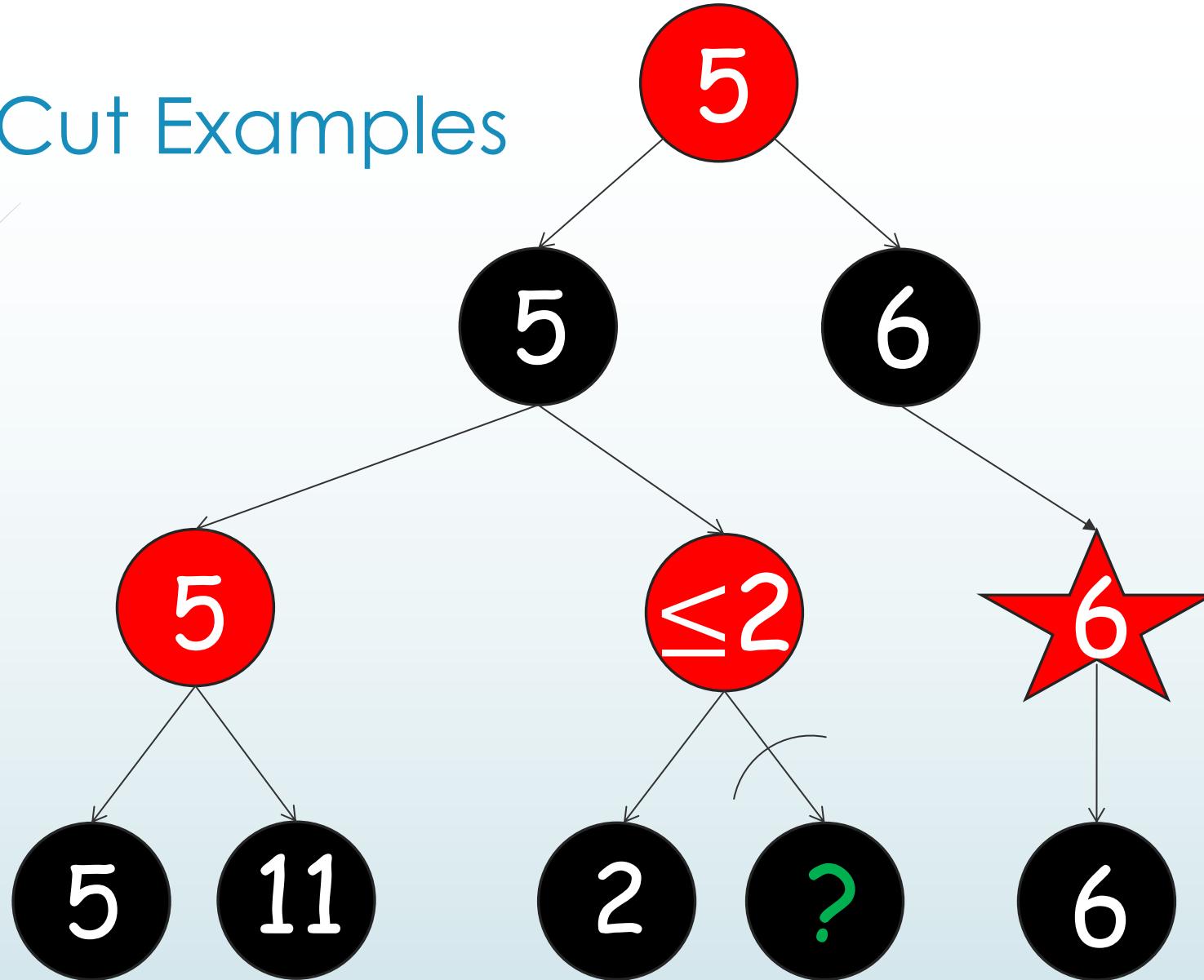
- ▶ Heuristic search based on backtracking
- ▶ Used in situations where there is a value to the solution at each point to allow for an evaluation
- ▶ Applied for Game Trees
- ▶ Alpha Cut
 - ▶ Maximum lower bound (Max Plays - Black)
- ▶ Beta Cut
 - ▶ Minimum upper bound (Min Play - Red)

Cut Examples

MIN

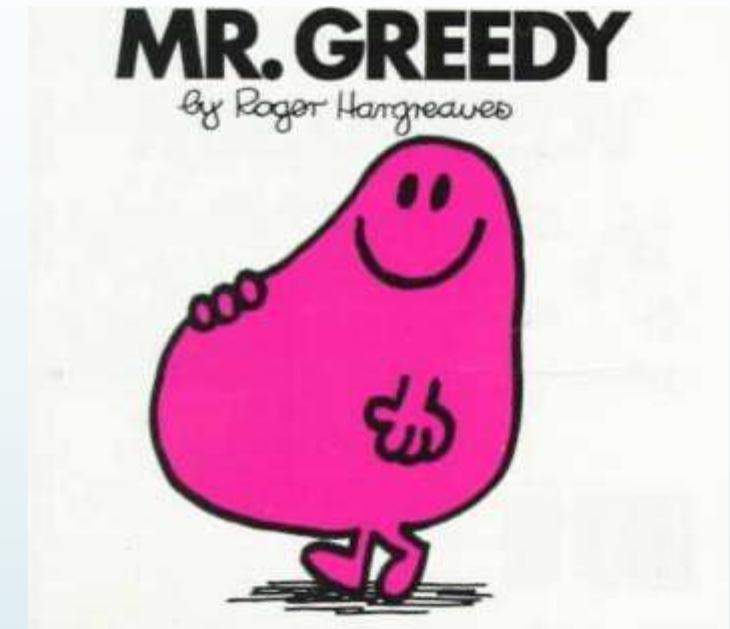
MAX

MIN



Greedy Approaches

- ▶ Generate and Test Methods primarily
- ▶ Current best solutions are held
- ▶ When I find something better – I get rid of what I had
- ▶ Always wanting more
- ▶ Always takes the Locally Optimal choice

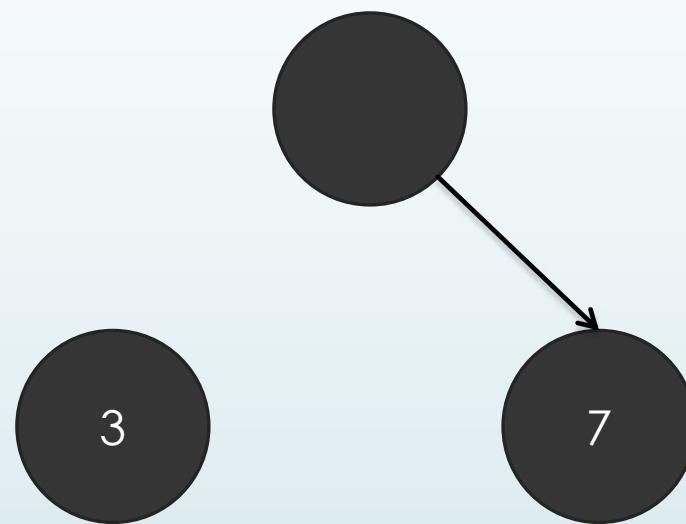


Greedy Algorithm Basics

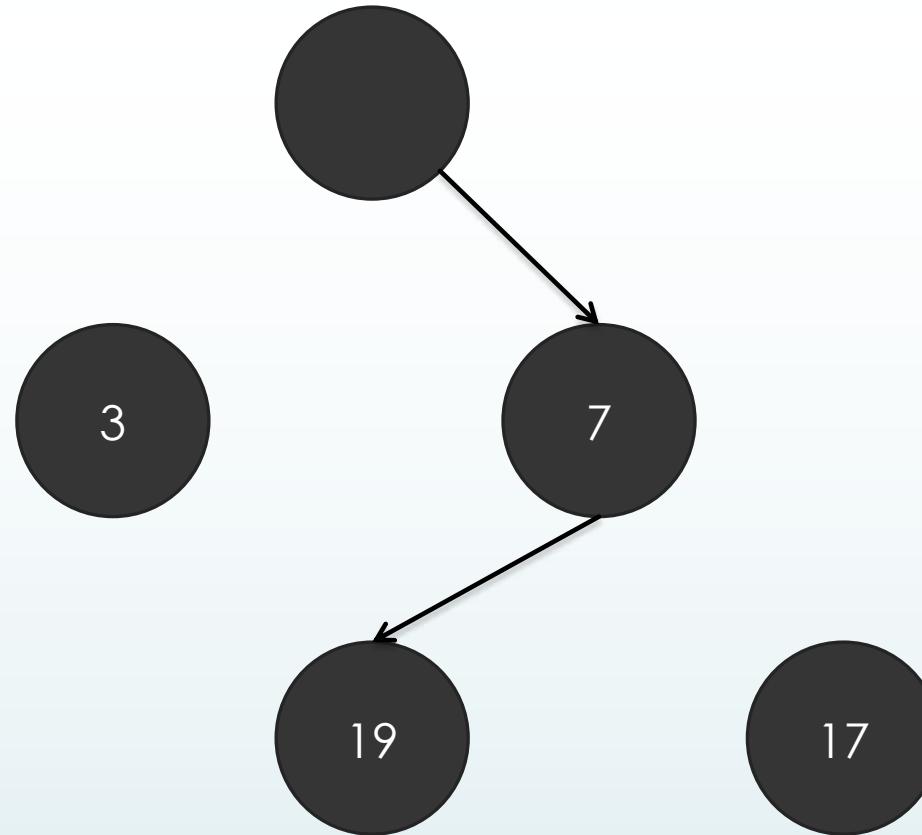
- ▶ Start with a search space from which a candidate solution will be taken from
- ▶ There is a number of decisions to be made in sequence to find a solution, at each step make a choice which maximizes the outcome
- ▶ Done

Example – Game Tree

- Two players are making moves in a game, each move gives a certain number of points to the player – goal is to make the most points



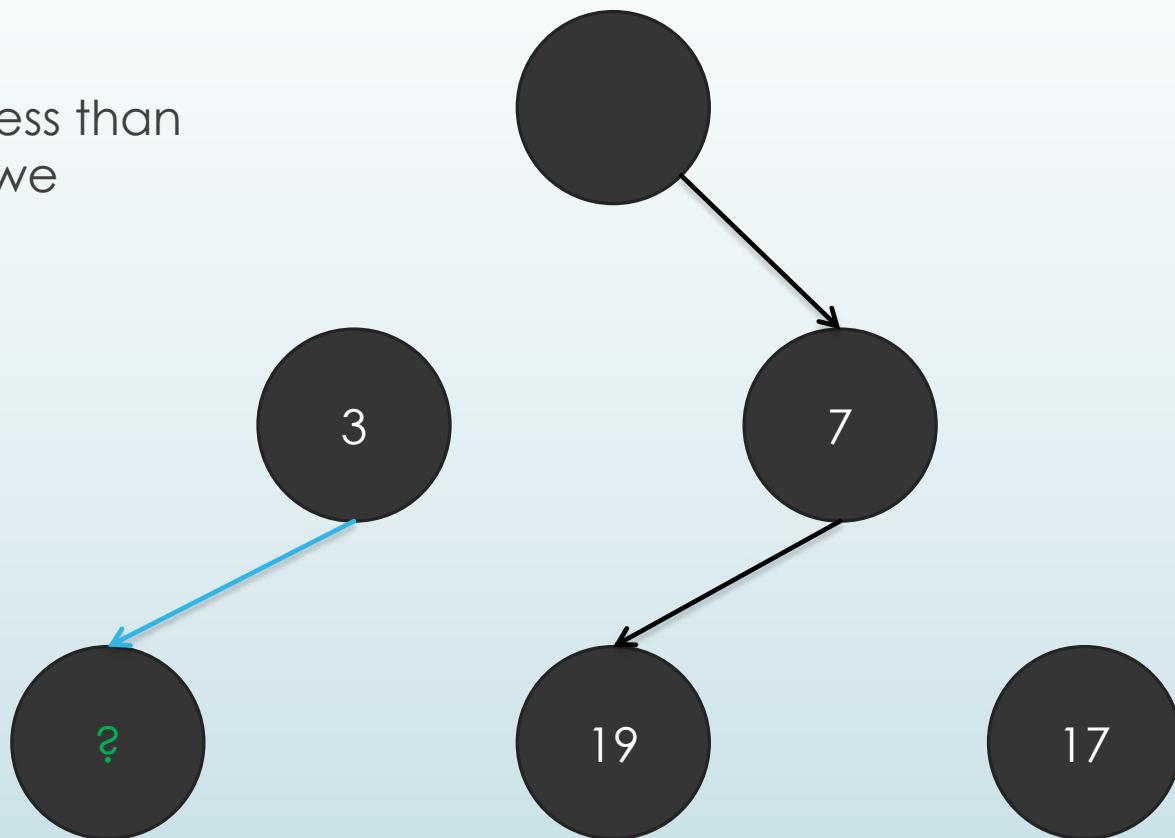
- In the first move our greedy player will take 7 points



- In the second move move our greedy player will take 19 points, for a total of 26 points

Was Greedy Good?

- ▶ Maybe – Depends on the Game being played
- ▶ If the ? Position was less than or equal to 23, then we have maximized the score
- ▶ If ? was greater, then we have failed



What Ensures That Greed Will Be Rewarded?

- ▶ Sub-problem optimization
 - ▶ Also called optimal substructures
 - ▶ The optimal solution to the problem contains optimal sub-problems
- ▶ Greedy Choice Property
 - ▶ We never need to consider the play until now and can move from this point only making the current best decision. We never need to review the past moves in order to make a better decision
 - ▶ In our example, if we have played the optimal game until now, then this starting point is also part of the optimal value on this move

Generate and Test

- ▶ Greedy is a constructive solution method – we construct the solution one part at a time
- ▶ Method to produce a solution may also generate the entire solution in one step; non-constructive
- ▶ Generate the problem solution
- ▶ Test the solution against the problem, or some representative amount of the problem

Hill Climbers

- ▶ Single Searching point
- ▶ Search locally for changes to the actions
- ▶ Have a representation of a solution string
- ▶ Have a test of the representation about the space
- ▶ Each solution string maps to an outcome
 - ▶ An objective function
 - ▶ A fitness on the problem

Climbing Hill in Our Example

- ▶ Representation
 - ▶ String of actions which we can take
- ▶ Objective
 - ▶ Maximization of profit value based on the actions
 - ▶ No string of actions is allowed to have an overlap
- ▶ Local Change
 - ▶ Add and/or remove some number of items from the string
- ▶ Climbing the Hill
 - ▶ Replace my position if the change is better

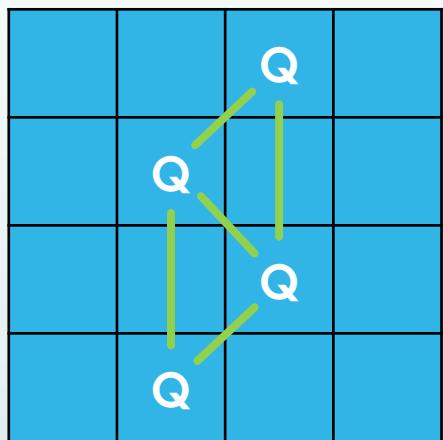
Hill Climbing Algorithm

```
function HILL-CLIMBING(problem) returns a solution state
  inputs: problem, a problem
  static: current, a node
          next, a node

  current  $\leftarrow$  MAKE-NODE(INITIAL-STATE[problem])
  loop do
    next  $\leftarrow$  a highest-valued successor of current
    if VALUE[next] < VALUE[current] then return current
    current  $\leftarrow$  next
  end
```

Hill Climber for N Queens Problem

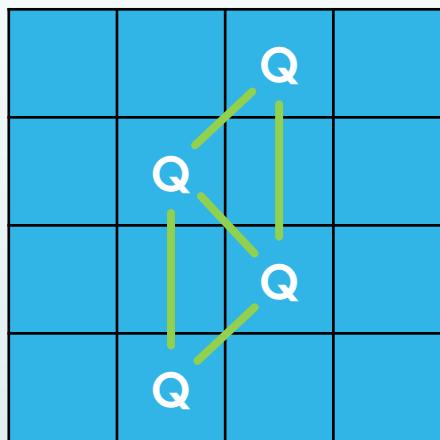
- h – number of queen pairs attacking each other
- $h \rightarrow \min$



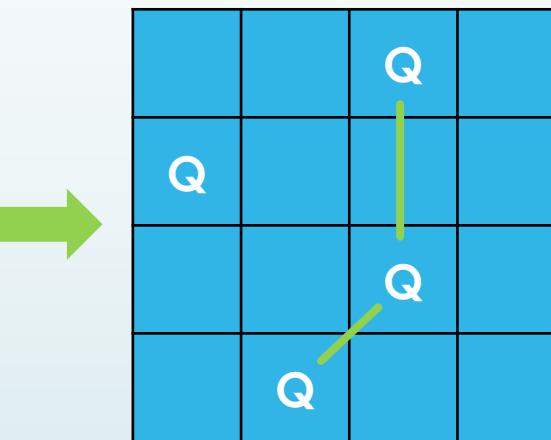
$$h = 5$$

Hill Climber for N Queens Problem

- h – number of queen pairs attacking each other
- $h \rightarrow \min$



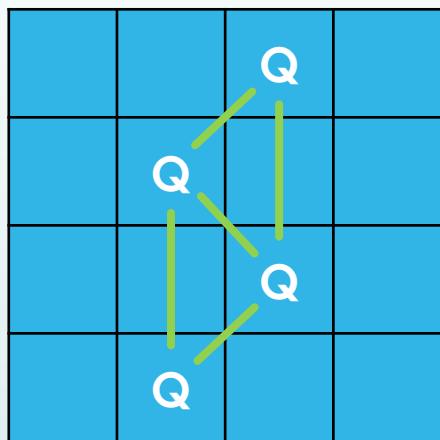
$h = 5$



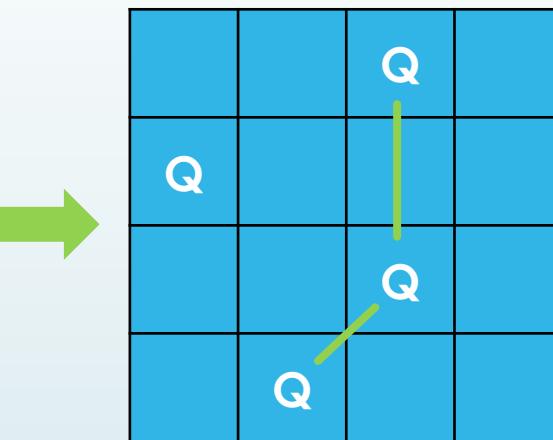
$h = 2$

Hill Climber for N Queens Problem

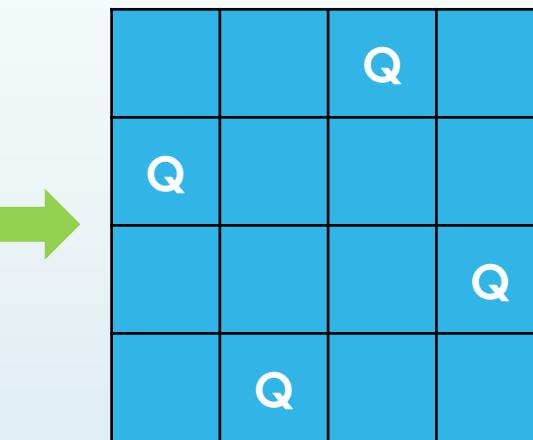
- h – number of queen pairs attacking each other
- $h \rightarrow \min$



$h = 5$



$h = 2$



$h = 0$

8-puzzle Problem

- ▶ The 8-puzzle problem is a puzzle invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square
- ▶ Goal is to rearrange the blocks so that they are in order. It is permitted to slide blocks horizontally or vertically into the blank square

3	1	2
4	5	8
6		7

Initial state



	1	2
3	4	5
6	7	8

Goal state

Hill Climbers on 8-puzzle Problem

1	2	
3	4	5
6	7	8

Goal state

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7

$$h = 5$$

Hill Climbers on 8-puzzle Problem

1	2	
3	4	5
6	7	8

Goal state

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

$$h = 5$$

3	1	2
4	5	8
6		7

$$h = 5$$

3	1	2
4		8
6	5	7



$$h = 6$$

3	1	2
4	5	8
	6	7

$$h = 4$$

3	1	2
4	5	8
6	7	

Hill Climbers on 8-puzzle Problem

1	2	
3	4	5
6	7	8

Goal state

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7



3	1	2
4	5	8
6	7	

$h = 5$

$h = 4$

Hill Climbers on 8-puzzle Problem

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7

$h = 5$



3	1	2
4	5	8
6	7	

$h = 4$



3	1	2
4	5	8
6	7	

$h = 5$

3	1	2
4	5	8
6	7	

$h = 3$

	1	2
3	4	5
6	7	8

Goal state

Hill Climbers on 8-puzzle Problem

1	2	
3	4	5
6	7	8

Goal state

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7



3	1	2
4	5	8
6	7	



3	1	2
4	5	
6	7	8

$$h = 5$$

$$h = 4$$

$$h = 3$$

Hill Climbers on 8-puzzle Problem

1	2	
3	4	5
6	7	8

Goal state

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7



3	1	2
4	5	8
6	7	



3	1	2
4	5	
6	7	8



3	1	2
4		5
6	7	8

$h = 5$

$h = 4$

$h = 3$

$h = 2$

Hill Climbers on 8-puzzle Problem

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7

$h = 5$



3	1	2
4	5	8
6	7	

$h = 4$



3	1	2
4	5	
6	7	8

$h = 3$



3	1	2
6	4	5
7	8	

$h = 1$

3	1	2
4		5
6	7	8

$h = 2$

1	2
3	4
6	7

Goal state

Hill Climbers on 8-puzzle Problem

- h – number of misplaced numbered tiles
- $h \rightarrow \min$

3	1	2
4	5	8
6		7

$h = 5$



3	1	2
4	5	8
6	7	

$h = 4$



3	1	2
4	5	
6	7	8

$h = 3$



3	1	2
4		5
6	7	8

$h = 2$

3	1	2
6	7	8

1	2
3	4
6	7

Goal state
 $h = 0$



Issues and Work About

- ▶ Local Maximums
 - ▶ Stuck in suboptimal places if our value is on a localized hill
 - ▶ Change the mutation
 - ▶ Change the representation
 - ▶ Random restart
- ▶ Multiple Optimums
 - ▶ Multiple Good Solutions
 - ▶ Random Restart
 - ▶ Large areas of equal solutions
 - ▶ Change the mutation
 - ▶ Random Restart
- ▶ Jumping over areas of the space
 - ▶ Selected Mutation hops over the space

Summary

- ▶ Uniformed Search
- ▶ Informed Search
- ▶ Exploitation and Exploration
- ▶ Backtracking
- ▶ Hill Climbers

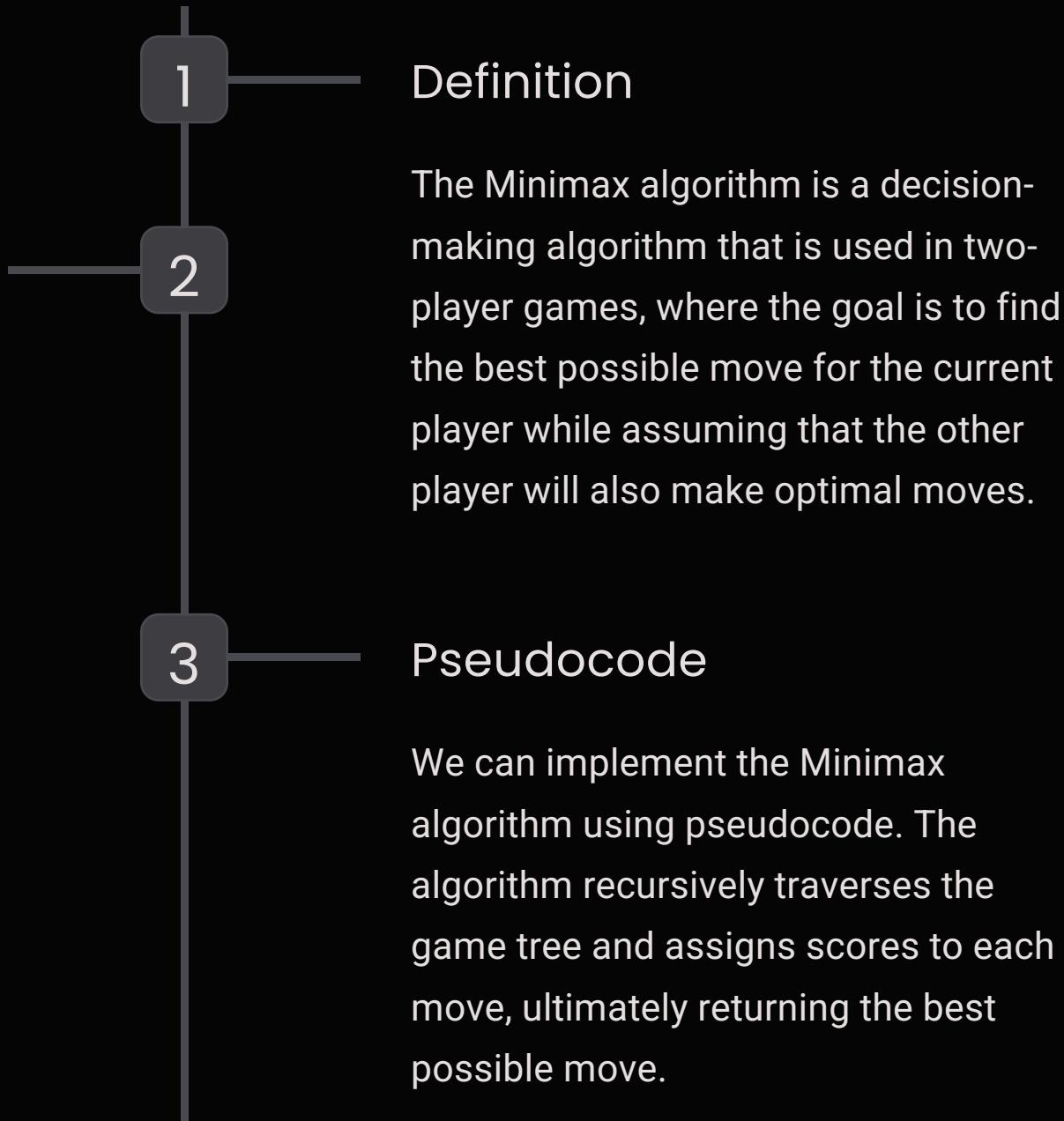
References

- ▶ [Minimax with alpha-beta pruning](#)
- ▶ [One more minimax with alpha-beta pruning](#)
- ▶ [Hill Climbers](#)

Basics of Minimax Algorithm

Working Principle

The algorithm works by creating a game tree that represents all possible moves by both players. It then analyzes the tree, minimizing the maximum possible loss for the player, and maximizing the minimum possible gain.



Understanding Alpha-Beta Pruning



The Concept

Alpha-Beta pruning is a technique used to improve the efficiency of the Minimax algorithm. It works by pruning branches of the game tree that don't need to be explored further, thereby reducing the total number of nodes that need to be evaluated.

How it Works

The algorithm maintains two values, alpha and beta, that are used to keep track of the best known score of the maximizing and minimizing player, respectively. It then prunes branches that can't affect the final score and limits the search space accordingly.

Performance Boost

Alpha-Beta pruning can significantly reduce the number of nodes evaluated, making the algorithm much faster than the regular Minimax algorithm for large game trees.

Benefits and Limitations

Advantages

- Can be applied to any two-player game
- Efficient in searching game trees
- Maximizes the player's benefit while minimizing the opponent's gain

Disadvantages

- Not suitable for games with imperfect information or unknown state space
- Requires a lot of memory to store and traverse the game tree
- Can be computationally expensive in certain cases

Real-World Applications

Gaming

It is widely used in games, from Chess and Go to computer games like Starcraft and World of Warcraft.

Decision Making

These algorithms can be used in decision-making scenarios such as resource allocation and project planning.

Optimization

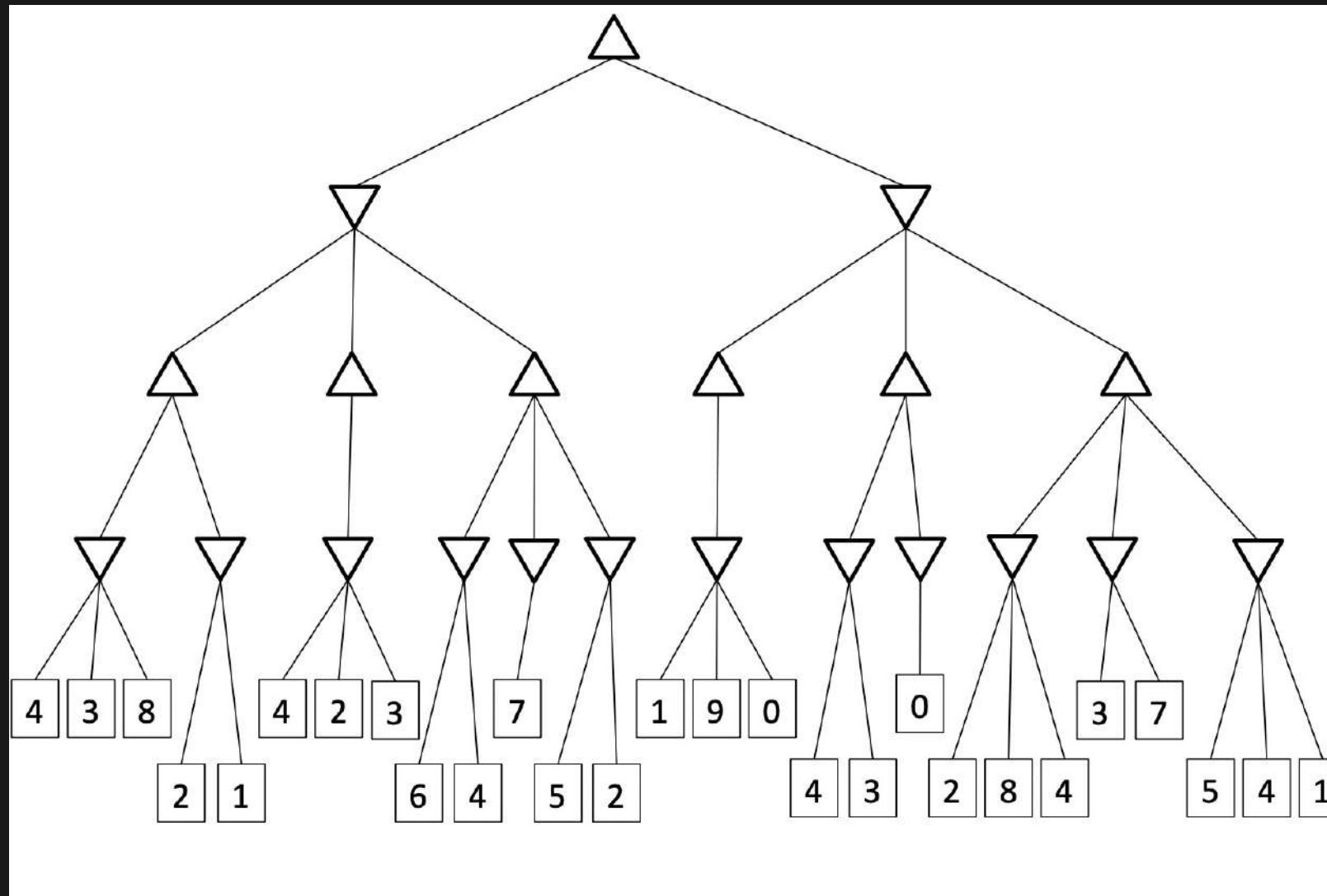
They can be used to optimize complex systems such as traffic flow, energy management, and financial portfolios.

Materials

<https://www.youtube.com/watch?v=l-hh51ncgDI>

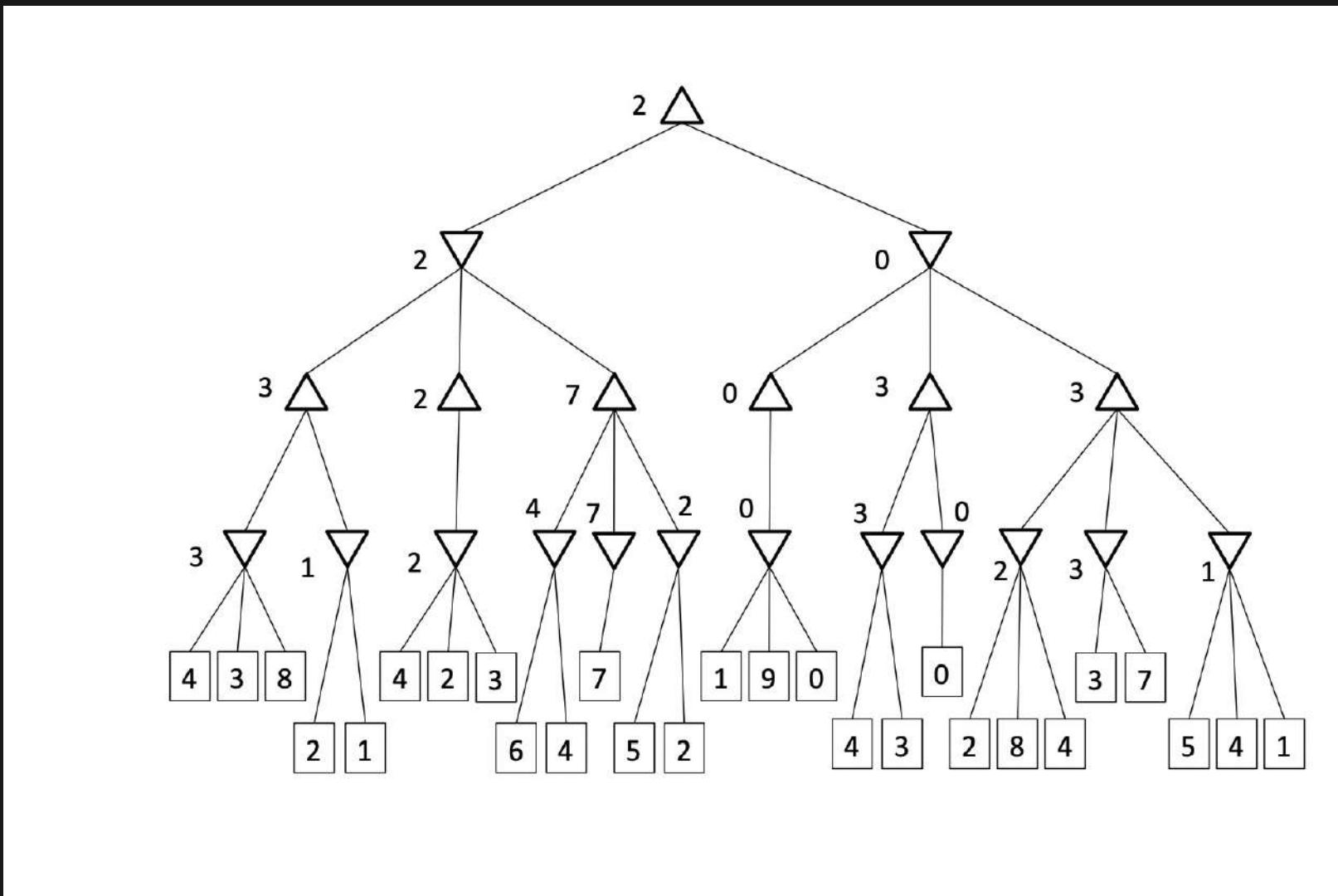
Exercise

Give the values calculated by minimax for all states in the tree.



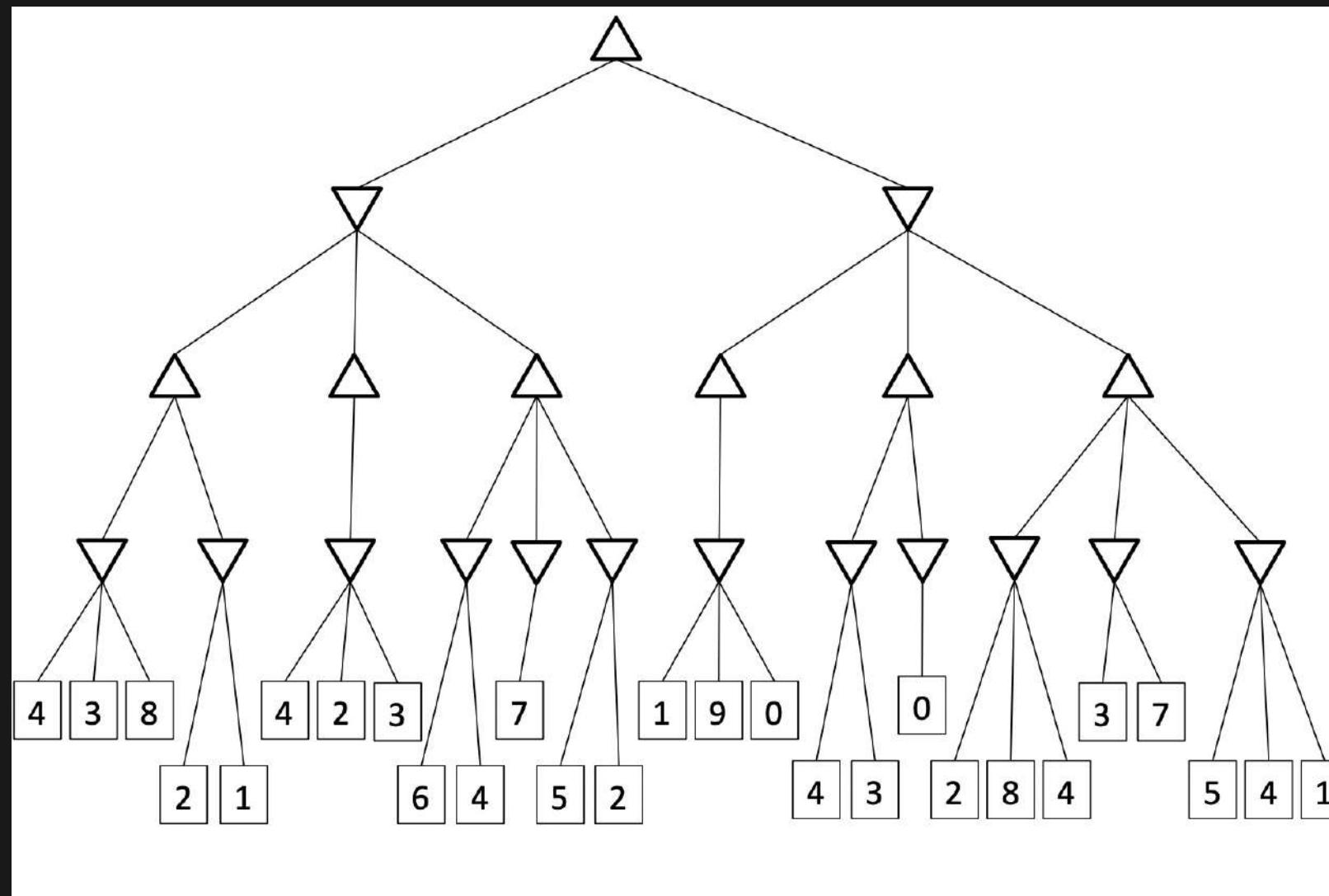
Solution

Give the values calculated by minimax for all states in the tree.



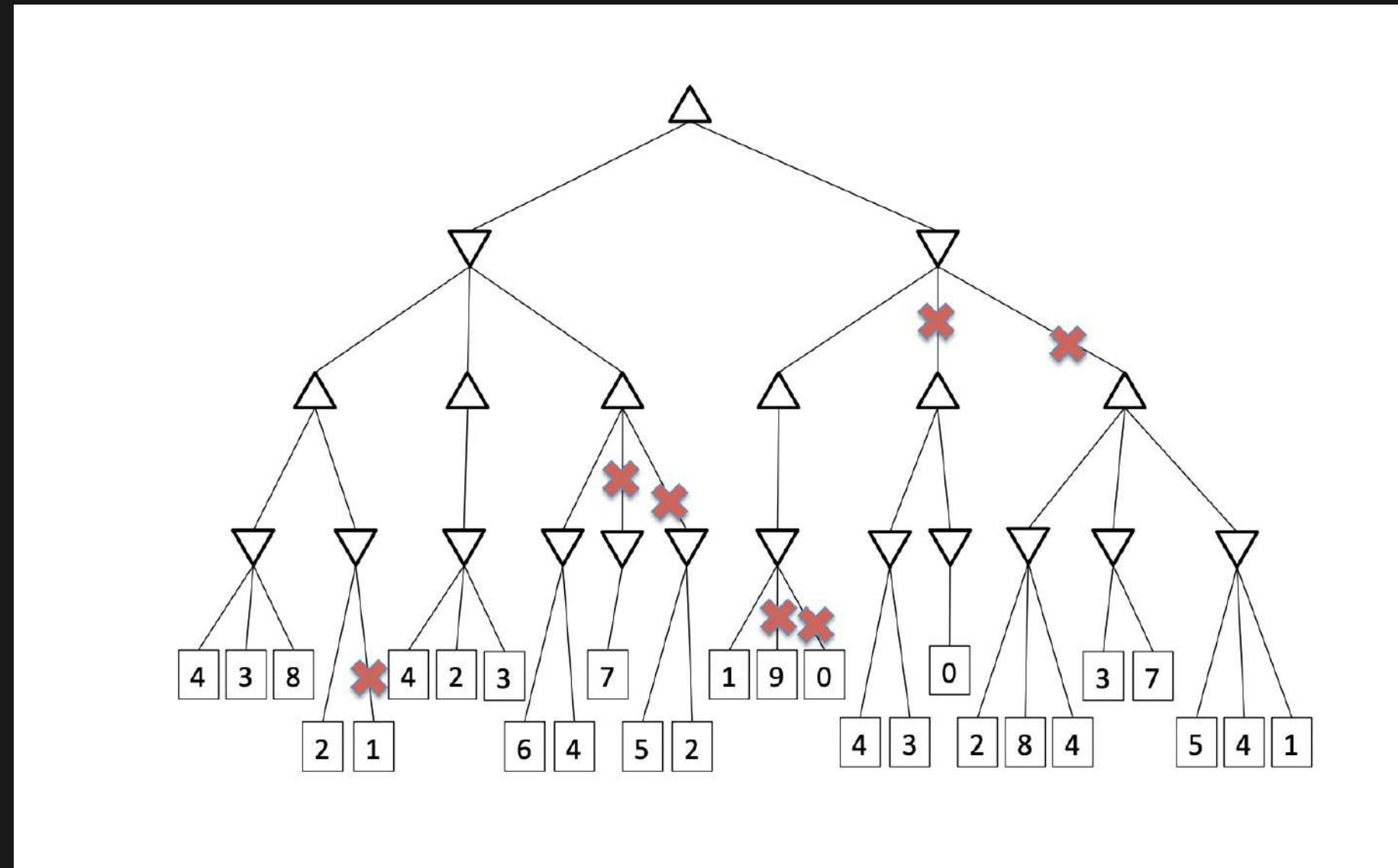
Extension

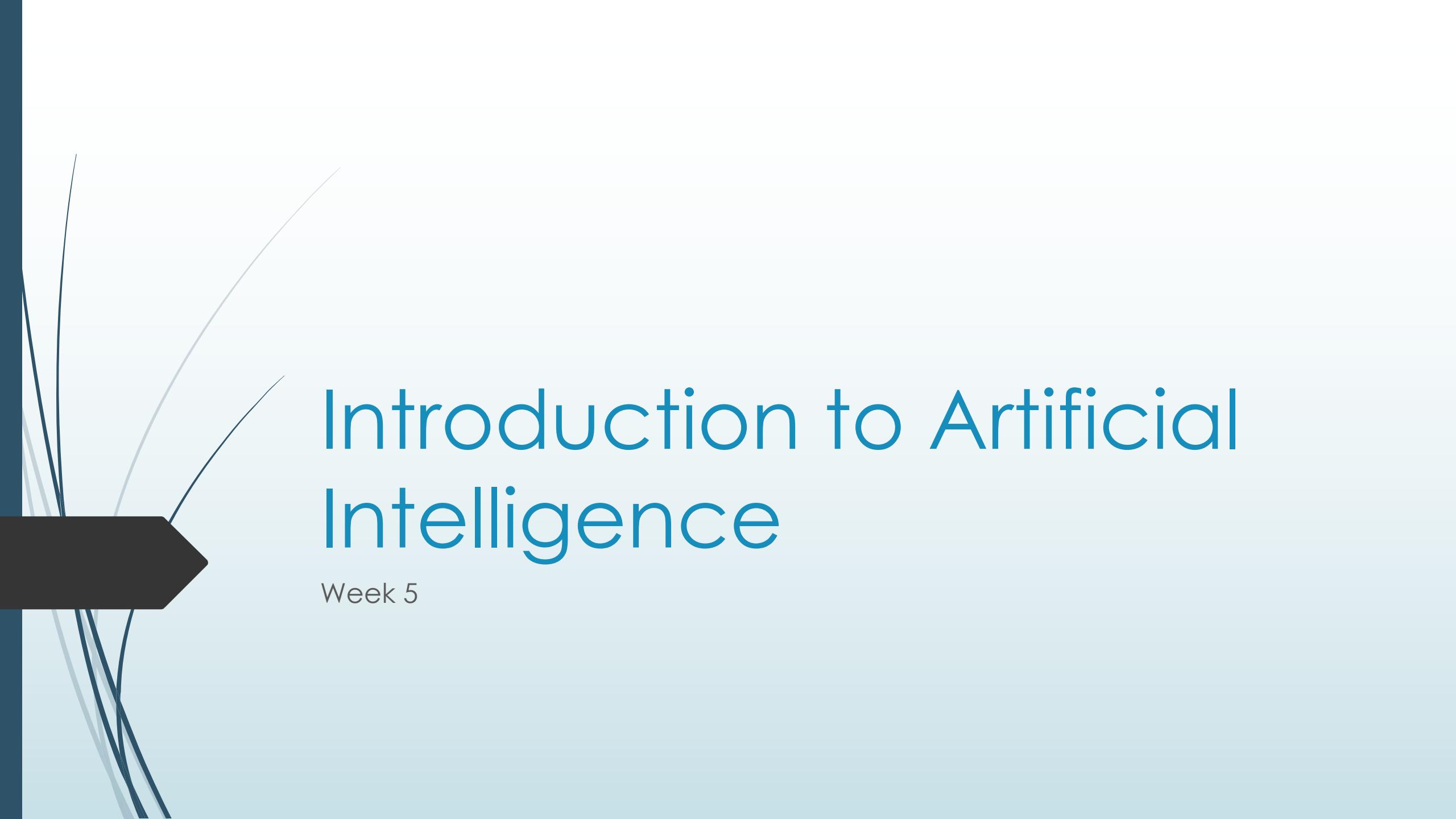
Indicate which branches of the tree will be pruned by alpha-beta pruning.



Solution

Indicate which branches of the tree will be pruned by alpha-beta pruning.





Introduction to Artificial Intelligence

Week 5



2

Learning by Searching

Part 1

Agenda

- ▶ Backtracking examples
- ▶ Game on backtracking

Backtracking

- ▶ Simulate the entire game
- ▶ Assume rational/perfect play from the opponent
- ▶ Theoretically there is a perfect game play method if we search the entire game
- ▶ Solved Games
 - ▶ Tic Tac Toe
 - ▶ Checkers
 - ▶ Two player heads-up limit poker – Recent discovery by researchers in University of Alberta
- ▶ Unsolved
 - ▶ Chess
 - ▶ Go

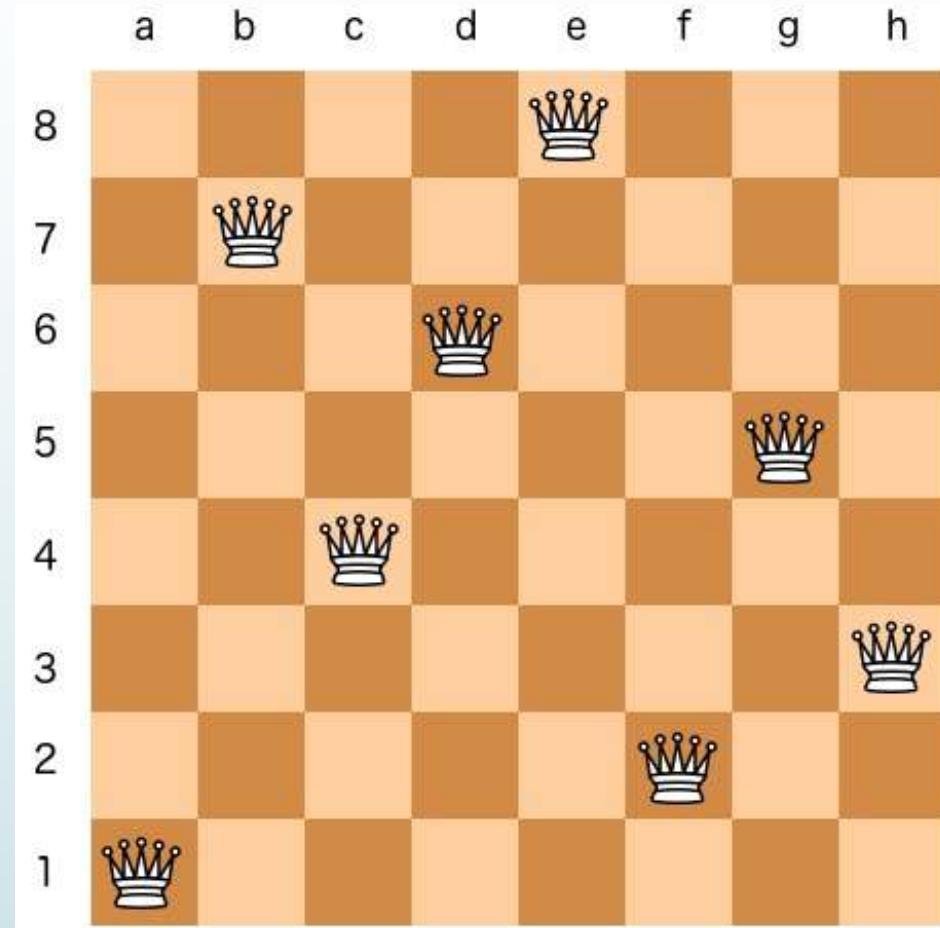
Backtracking with Alpha Beta Cuts

- ▶ Heuristic search based on backtracking
- ▶ Used in situations where there is a value to the solution at each point to allow for an evaluation
- ▶ Applied for Game Trees
- ▶ Alpha Cut
 - ▶ Maximum lower bound (Max Plays - Black)
- ▶ Beta Cut
 - ▶ Minimum upper bound (Min Play - Red)

N Queens and Incremental Goals



- ▶ Place N queens onto an NxN chess board such that no queen attacks any other queen; a more complicated problem is a Costas Array
- ▶ States
 - ▶ Place a queen down
- ▶ Goal
 - ▶ No queen attacks any other queen
 - ▶ Hard sometimes to see a good pathway to the goal
- ▶ Need a measure in order to see a good pathway
 - ▶ Minimize number of queens attacking



Solutions for N Queens Problem

N	Fundamental	All
1	1	1
2	0	0
3	0	0
4	1	2
5	2	10
6	1	4
7	6	40
8	12	92
...

- If solutions that differ only by the symmetry operations of rotation and reflection of the board are counted as one and called *fundamental* solutions
- Solutions exist for all natural numbers N with the exception of N = 2 and N = 3

N Queens Problem

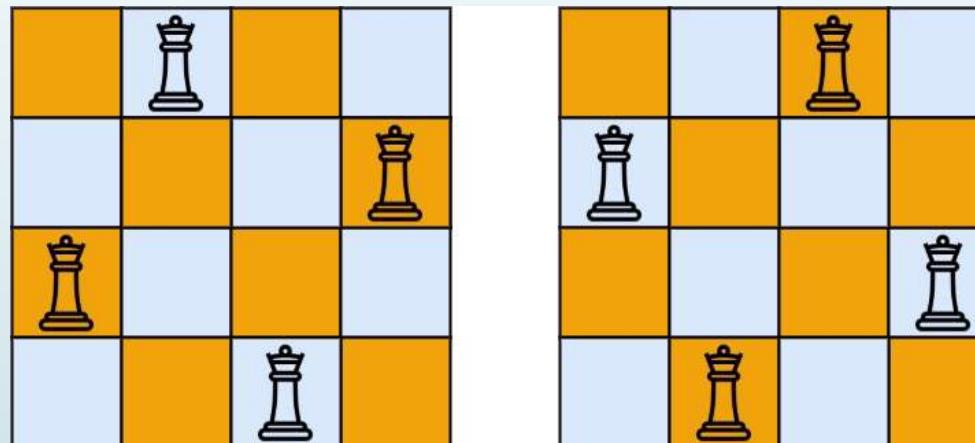
- ▶ By using backtracking algorithm preferably with heuristics solve the N queens problem with N=4
- ▶ Which heuristics to apply?

N Queens Problem

- ▶ By using backtracking algorithm preferably with heuristics solve the N queens problem with N=4
- ▶ Which heuristics to apply?
 - ▶ No more than 1 queen on rows, columns and on the diagonals
 - ▶ If while consideration the column/row was left empty, backtrack

N Queens Problem

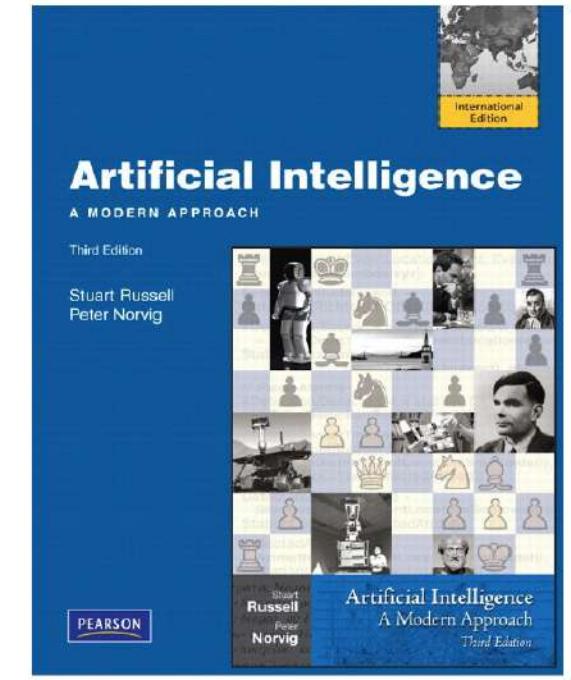
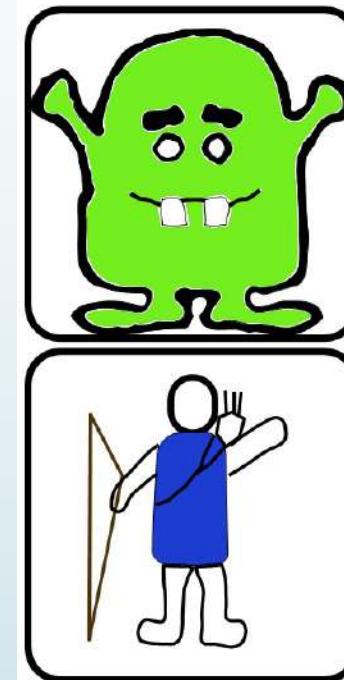
- ▶ By using backtracking algorithm preferably with heuristics solve the N queens problem with N=4
- ▶ Which heuristics to apply?
 - ▶ No more than 1 queen on rows, columns and on the diagonals
 - ▶ If while consideration the column/row was left empty, backtrack



The Wumpus World

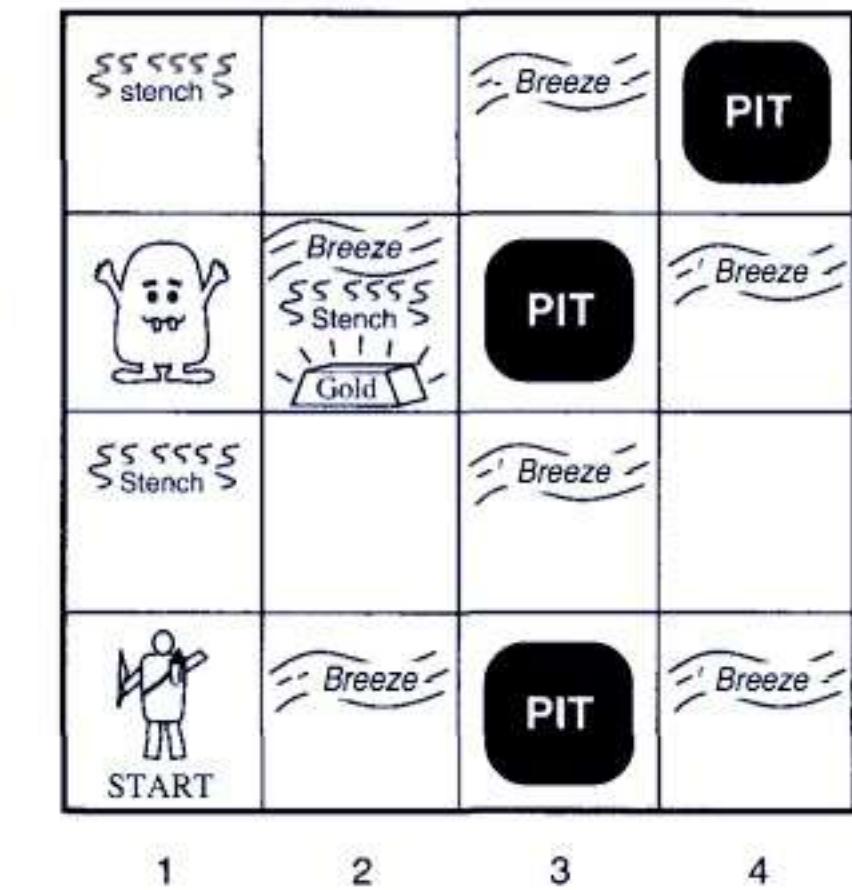
- Wumpus was an early computer game, based on an agent who explores a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the wumpus, a beast that eats anyone who enters its room. To make matters worse, some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, who is too big to fall in). The only mitigating feature of living in this environment is the occasional heap of gold

Wumpus World



The Wumpus World Rules (1)

- Like the vacuum world, the wumpus world is a grid of squares surrounded by walls, where each square can contain agents and objects. The agent always starts in the lower left corner, a square that we will label [1,1]. The agent's task is to find the gold if it is possible, return to [1,1] and climb out of the cave



The Wumpus World Rules (2)

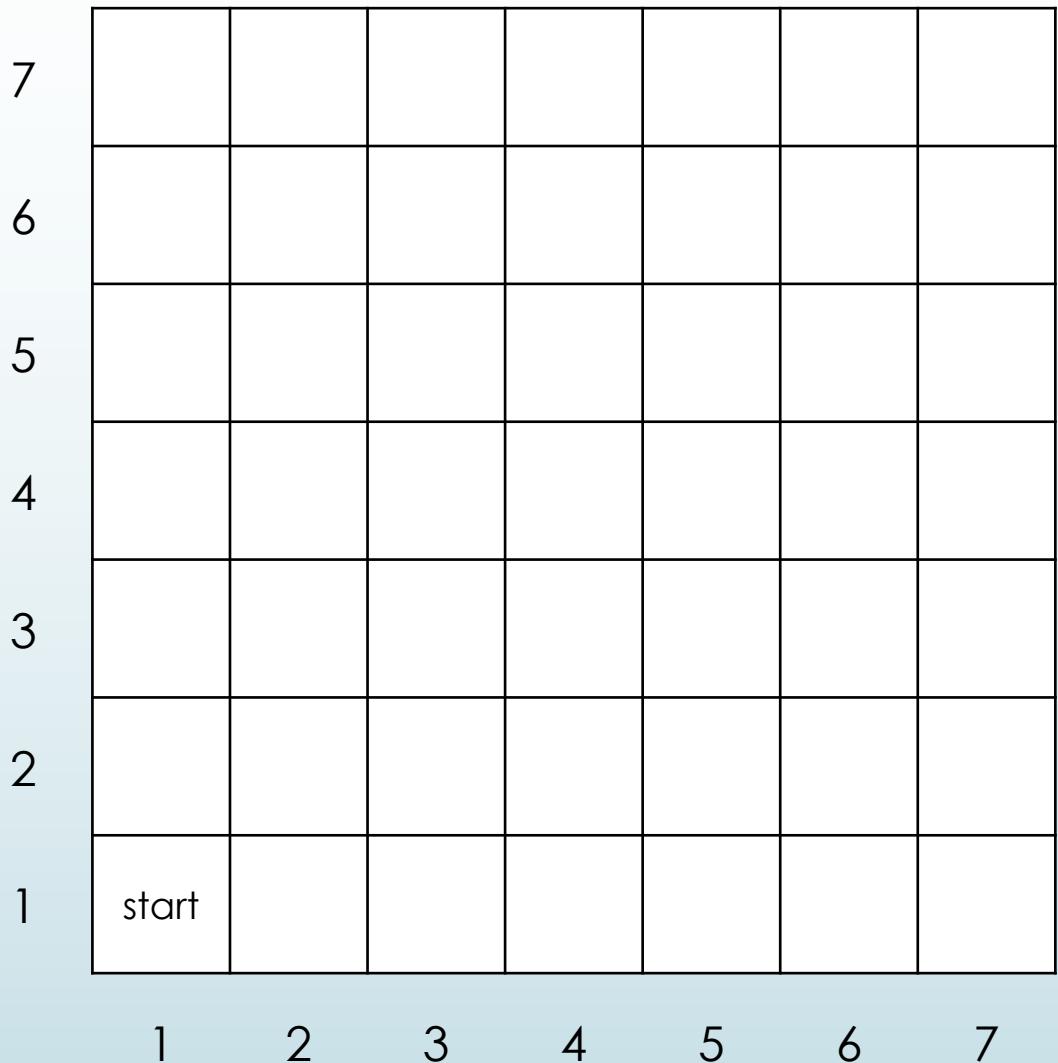
- ▶ In the square containing the wumpus and in the directly (not diagonally) adjacent squares the agent will perceive a stench
- ▶ In the squares directly adjacent to a pit, the agent will perceive a breeze
- ▶ In the square where the gold is, the agent will perceive a glitter
- ▶ When an agent walks into a wall, it will perceive a bump
- ▶ When the wumpus is killed, it gives out a woeful scream that can be perceived anywhere in the cave
- ▶ The percepts will be given to the agent in the form of a list of five symbols; for example, if there is a stench, a breeze, and a glitter but no bump and no scream, the agent will receive the percept [Stench, Breeze, Glitter, None, None]. The agent cannot perceive its own location

The Wumpus World Rules (3)

- ▶ Just as in the vacuum world, there are actions to go forward, turn right by 90°, and turn left by 90°. In addition, the action *Grab* can be used to pick up an object that is in the same square as the agent. The action *Shoot* can be used to fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits and kills the wumpus or hits the wall. The agent only has one arrow, so only the first *Shoot* action has any effect. Finally, the action *Climb* is used to leave the cave; it is effective only when the agent is in the start square
- ▶ The agent dies a miserable death if it enters a square containing a pit or a live wumpus. It is safe (but smelly) to enter a square with a dead wumpus
- ▶ The agent's goal is to find the gold and bring it back to the start as quickly as possible, without getting killed. To be precise, 1000 points are awarded for climbing out of the cave while carrying the gold, but there is a 1-point penalty for each action taken, and a 10,000-point penalty for getting killed

Let's Play

- We will try to simulate backtracking search with some modifications
 - Divide into teams of 4-6 people and choose a captain and score counter
 - Talk for 5 min inside the team about the applied heuristics



Scheme

- Not each map can be solved
- Some maps can be solved only by risking, which does not guarantee the victory

		B					B	PIT
7								
6	B	PIT	B			B		B
5		B			B	PIT	B	PIT
4			B	PIT	B	S	B	
3			B	B	S B	W S	S	
2		B	PIT	B	PIT	S B G	B	
1	start		B	PIT	B	B	B	PIT
	1	2	3	4	5	6	7	

References

- [Minimax and alpha-beta pruning](#)



Introduction to Artificial Intelligence

Week 6



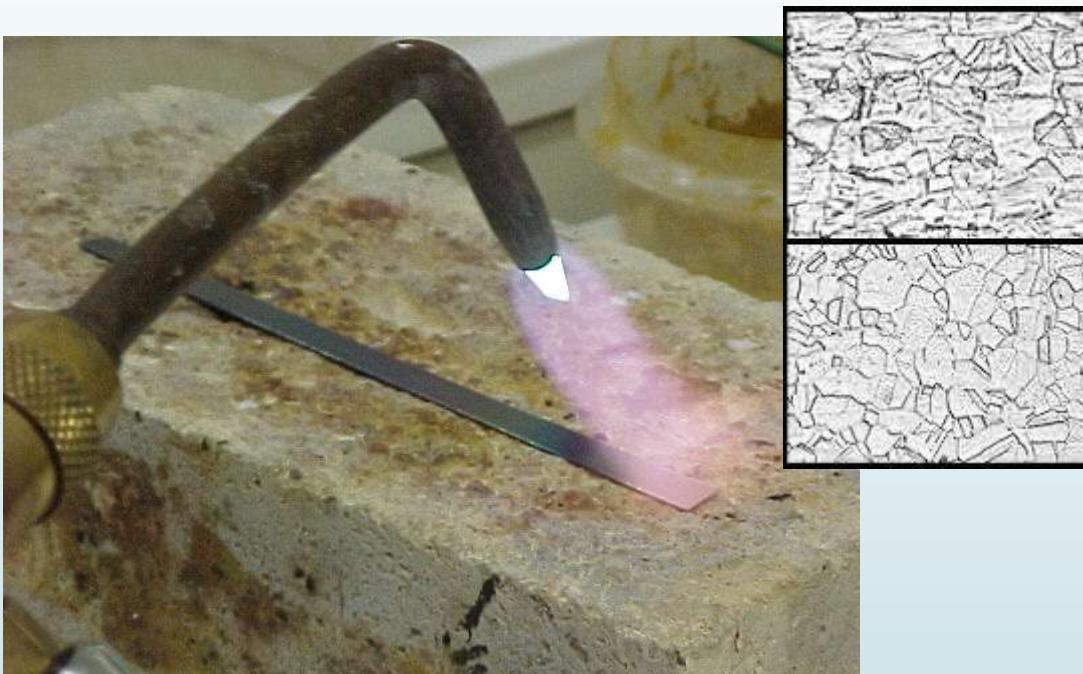
2

Learning by Searching

Part II

Simulated Annealing

- ▶ Based on the properties of Metal working
- ▶ Metal is heated and cooled in order for it to be made workable



Simulation of This Process

- ▶ Set down a number of search points
- ▶ Points move dependent upon the temperature applied
- ▶ They move about the space based on a temperature which is slowly reducing over time
 - ▶ This allows for large movement at the beginning of the algorithm and reduces into smaller and smaller movements as time goes on
- ▶ Two Types
 - ▶ Temperature controls size of step
 - ▶ Temperature controls the acceptance of a step

Simulated Annealing – Step Size

- ▶ Start at an initial temperature value (usually something hot) and create a random initial solution
- ▶ Until temperature is cool
 - ▶ Make a change to the current solution (neighbour) based on the temperature
 - ▶ Current location + Rand_Distribution(temperature)
 - ▶ Decrease the temperature

Temperature Controls the Step Size

Temperature: 25.0



Simulated Annealing

- ▶ Start at an initial temperature value (usually something hot) and create a random initial solution
- ▶ Until temperature is cool
 - ▶ Make a small change to the current solution (neighbour)
 - ▶ Decide if we move to this neighbour solution
 - ▶ Neighbour solution is better OR
 - ▶ $\text{Rand_Distribution}() < \text{temp}$
 - ▶ Decrease the temperature

Exploring and Exploiting

- ▶ Exploration – the want for a search method to find new areas of the search space which have not been visited yet
- ▶ Exploitation – the want for a search method to locally search about the best known areas in the space in order to refine these current solutions
- ▶ All search algorithms will have some mix of both properties
- ▶ Note that if we only have a limited number of objective tests available, then we have a zero-sum game – we can explore or we can search with a test

Exploration or Exploitation?

- We create a simulated annealing which has a reheat if we are not seeing a 10% improvement in the solution

Exploration

- ▶ The reheat increases the step size
- ▶ The reheat increases the probability we accept worst solutions

Exploration or Exploitation?

- After the application of a search we look at the least significant digit and test from 0-9 taking the best final solution

Exploitation

- ▶ Looking at the least significant digit we are looking in a close locality to the previous, making the assumption that close parameter settings generate closer outputs

Parallel

- ▶ All of these methods are trivially parallel
- ▶ We can place one of these searching units onto a different ‘core’ processing unit without issue
- ▶ Hill Climbers and Simulated Annealing use the idea of a solution neighbourhood, so we can divide the space

A* Search

- ▶ Fast well-established algorithm for game AI
 - ▶ Pathways
 - ▶ Movement of NPC
 - ▶ RTS movement
 - ▶ Usually one member of a flock
- ▶ Works in areas where we do not know the connectivity
 - ▶ Makes it very suitable for PCG – because we do not need nodes defined by the generator – but can still understand the measurements
- ▶ Can take into account multiple different connection topologies and styles of movement
- ▶ Large number of iterative improvements and problem domain specific implementations with heuristics

A* Formula

- $f(n) = g(n) + h(n)$
 - $g(n)$ is the path from the initial position to the current node
 - $h(n)$ is the path from the current node to the goal position

A* Algorithm

- ▶ A node contains a location, a cost, and a parent
- ▶ Initialize a set CLOSED to be null
- ▶ Initialize a priority queue based on cost of a node called OPEN to be the starting location, set cost = 0
- ▶ While OPEN is non-empty do
 - ▶ Current <- Pop OPEN.top
 - ▶ For all neighbouring locations X
 - ▶ If not in (BLOCKED or CLOSED)
 - ▶ If not in OPEN
 - ▶ X.cost = current.cost+1
 - ▶ X.parent = current
 - ▶ Push X to OPEN

A* Algorithm

- ▶ Else if X is in OPEN
 - ▶ If current.cost+1 < OPEN.X.cost // we found a better path
 - ▶ OPEN.X.cost = current.cost+1
 - ▶ OPEN.X.parent = current
- ▶ END FOR ALL X
- ▶ Push Current to CLOSED

A* for PATH FINDING

- ▶ REPORT_PATH (START, FINISH)
 - ▶ STACK PATH initialized to NULL
 - ▶ RUN A* (START)
 - ▶ Find FINISH in CLOSED
 - ▶ Current <- FINISH
 - ▶ While Current not START
 - ▶ PUSH Current to PATH
 - ▶ Current <- Current.parent
 - ▶ PRINT START
 - ▶ While STACK not NULL
 - ▶ PRINT POP PATH

Rover World

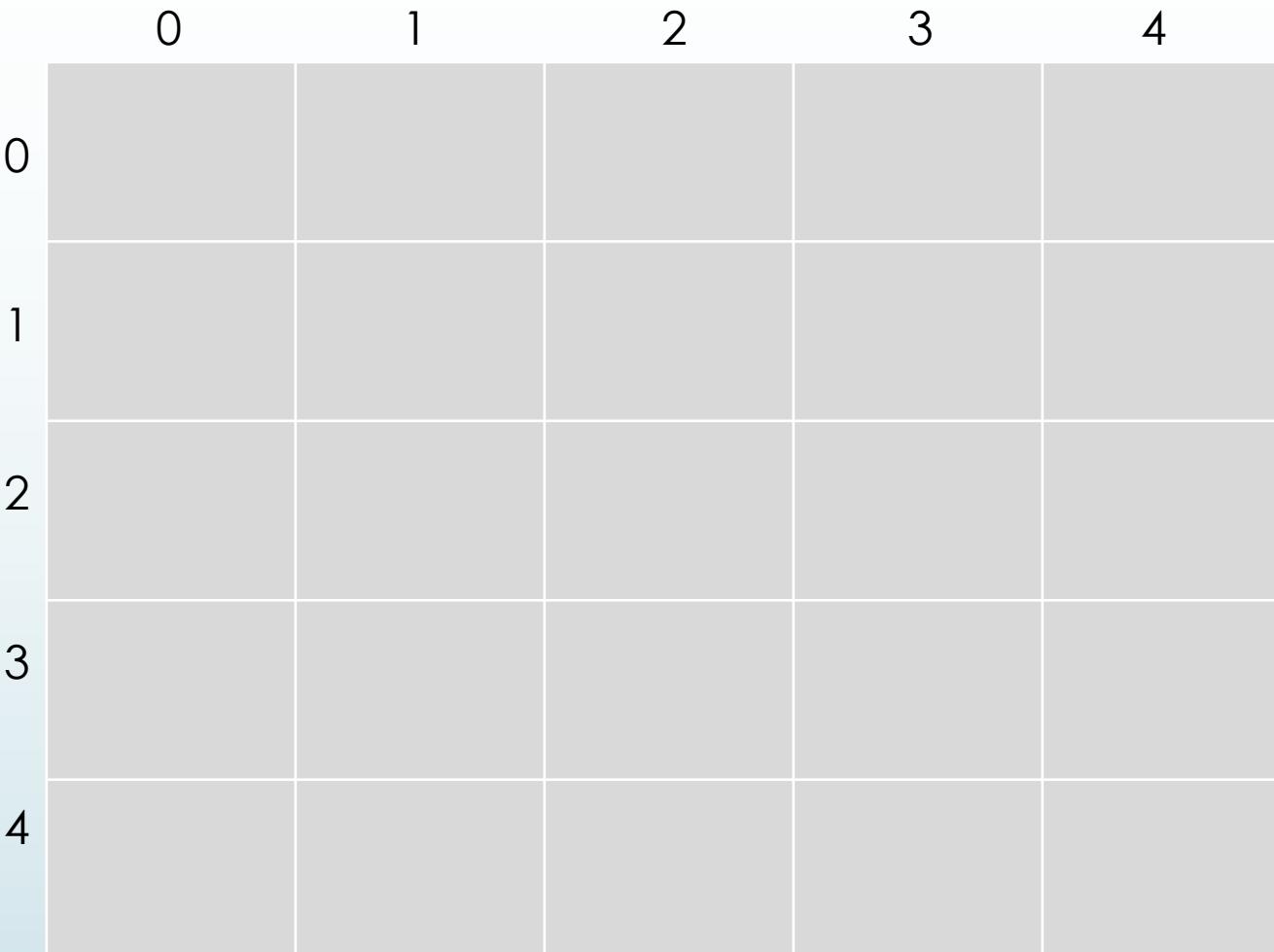
Rover



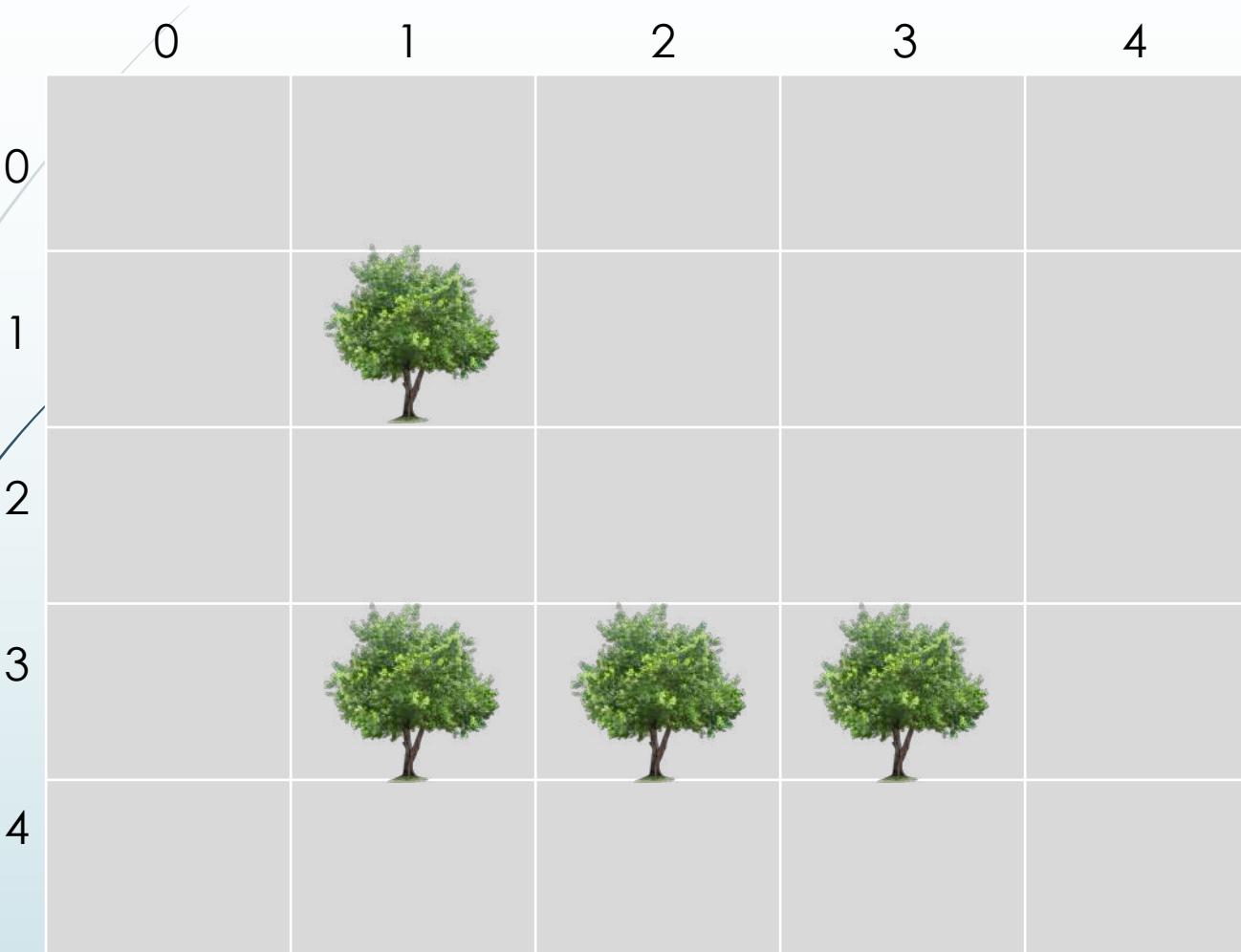
Obstacle



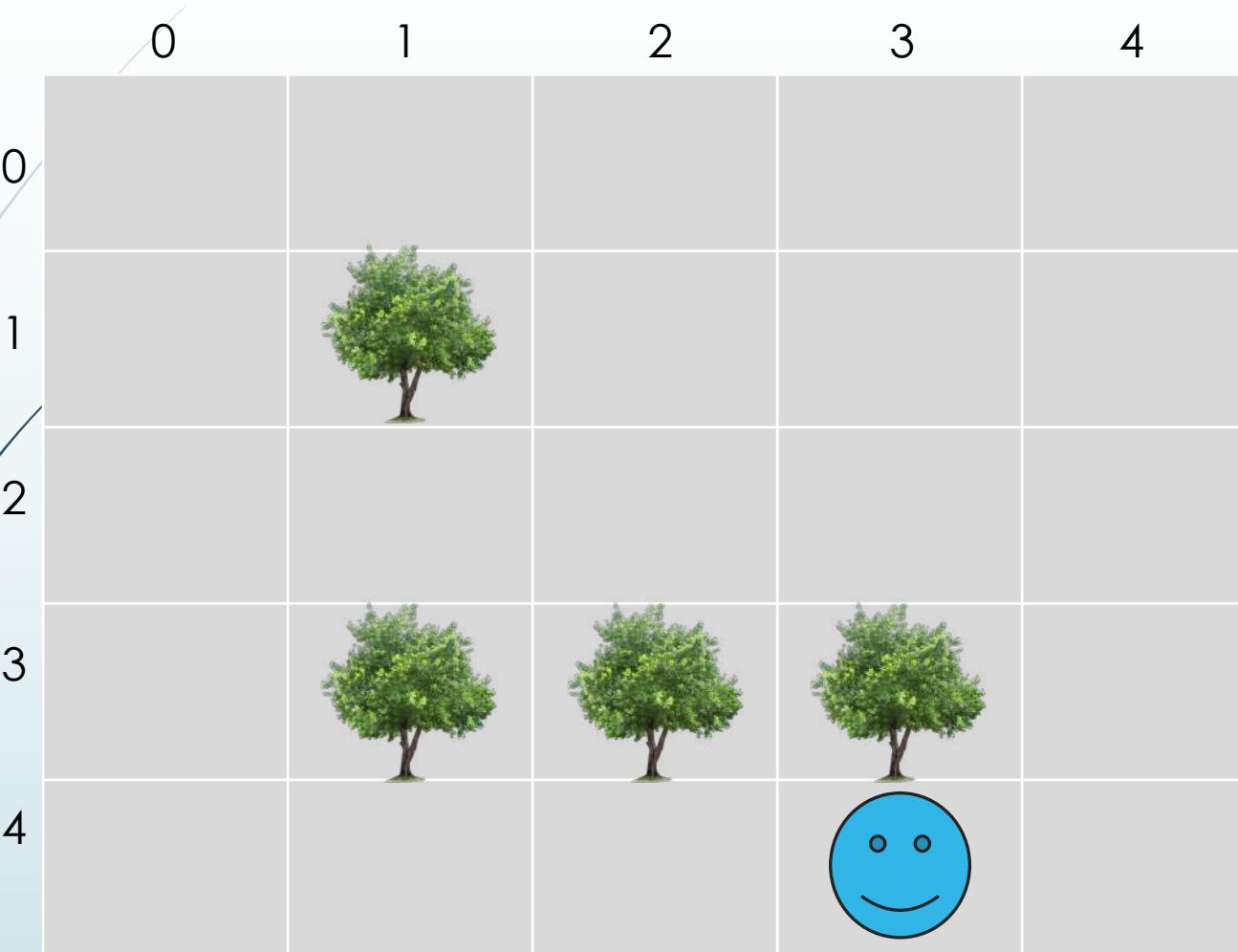
Goal



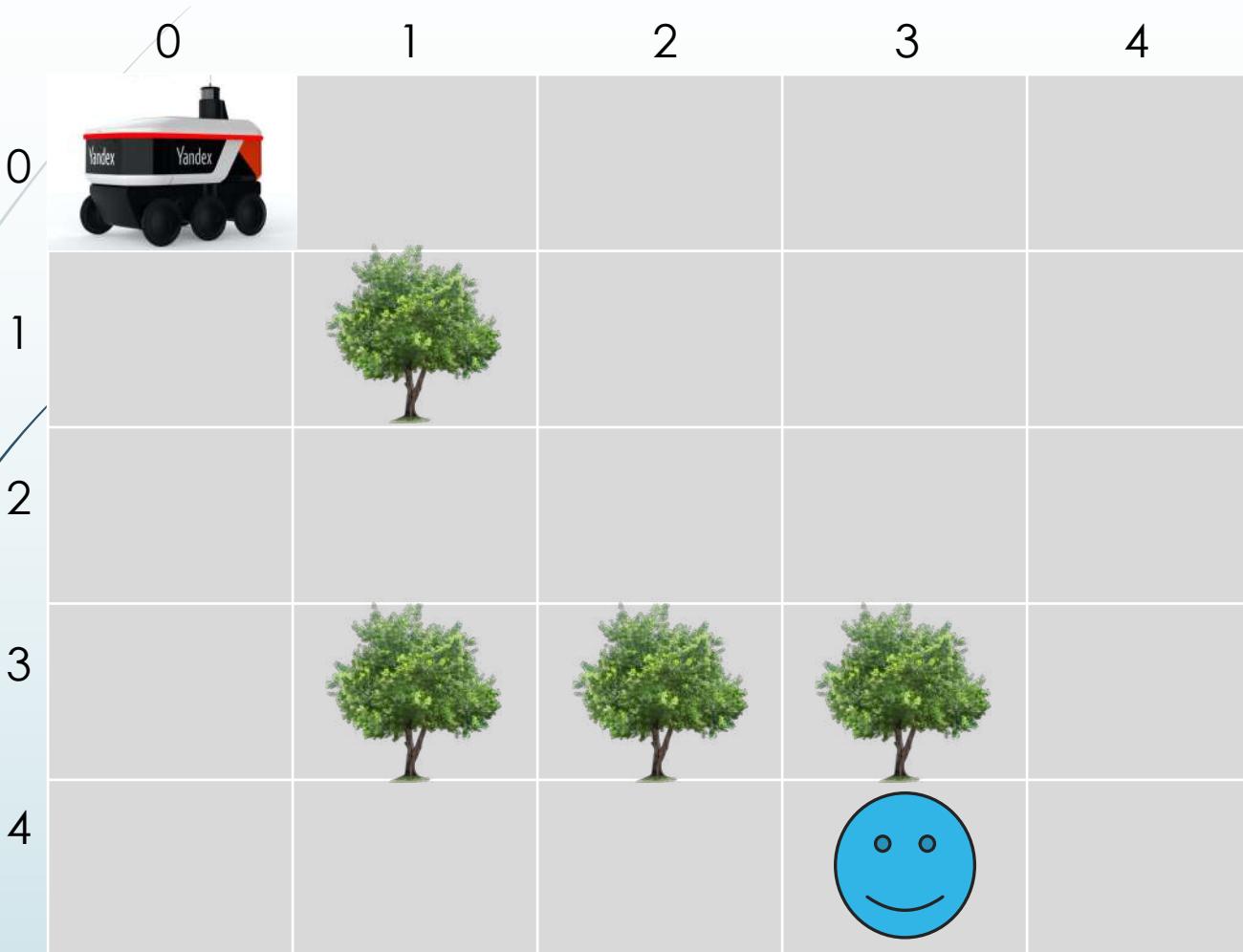
World with Obstacles



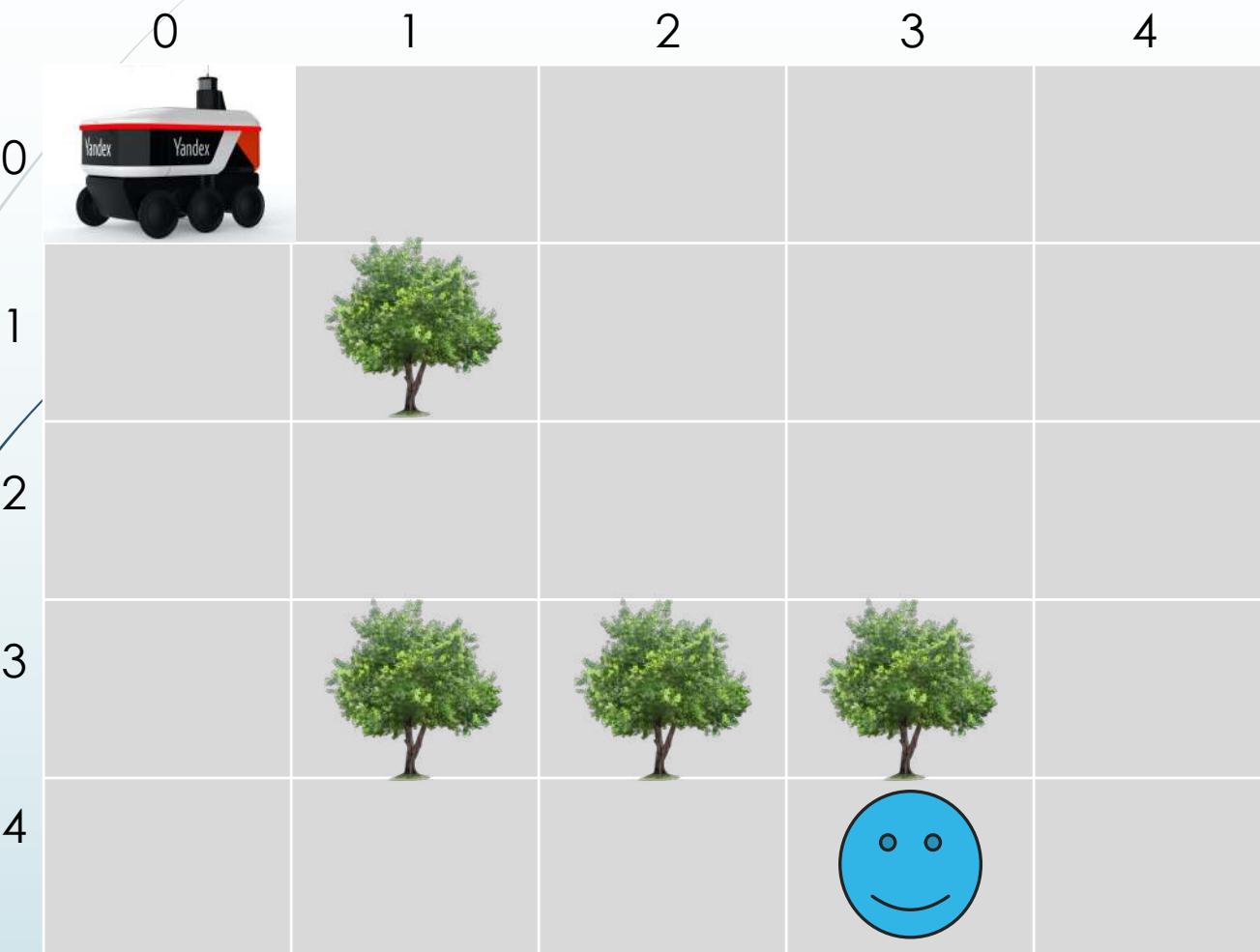
World with Obstacles and Goal



World with Obstacles, Goal and Starting Position



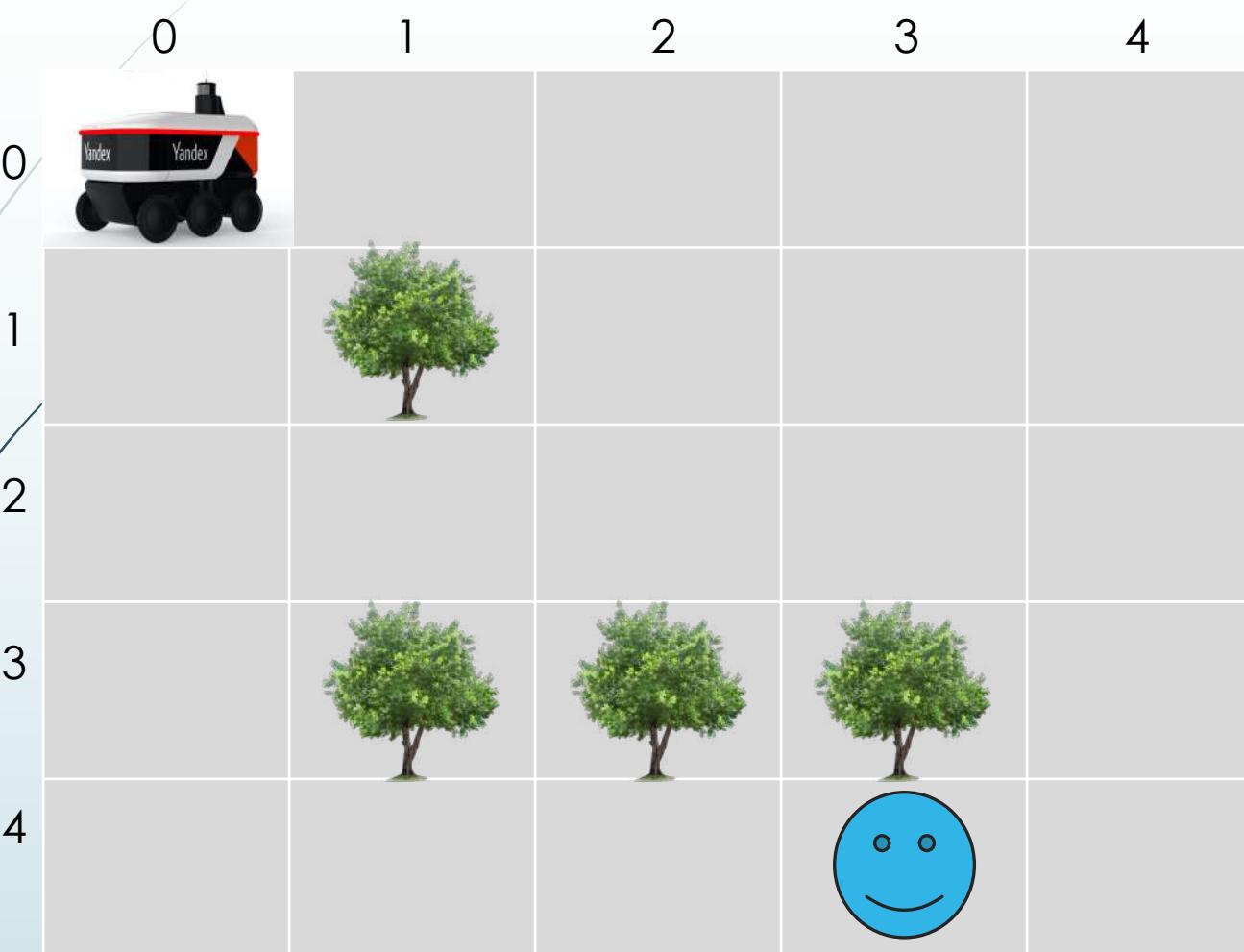
World with Obstacles, Goal and Starting Position



- To find the path to the customer Rover requires A* algorithm to be applied
- Assume that the Rover can move horizontally (10p), vertically(10p) and diagonally (14p)
- For heuristics we use a straight line distance from the current cell till the goal
- For each cell x costs $g(x)$, $h(x)$ and $f(x) = g(x) + h(x)$ have to be assigned

$g(x)$	$h(x)$
$f(x)$	

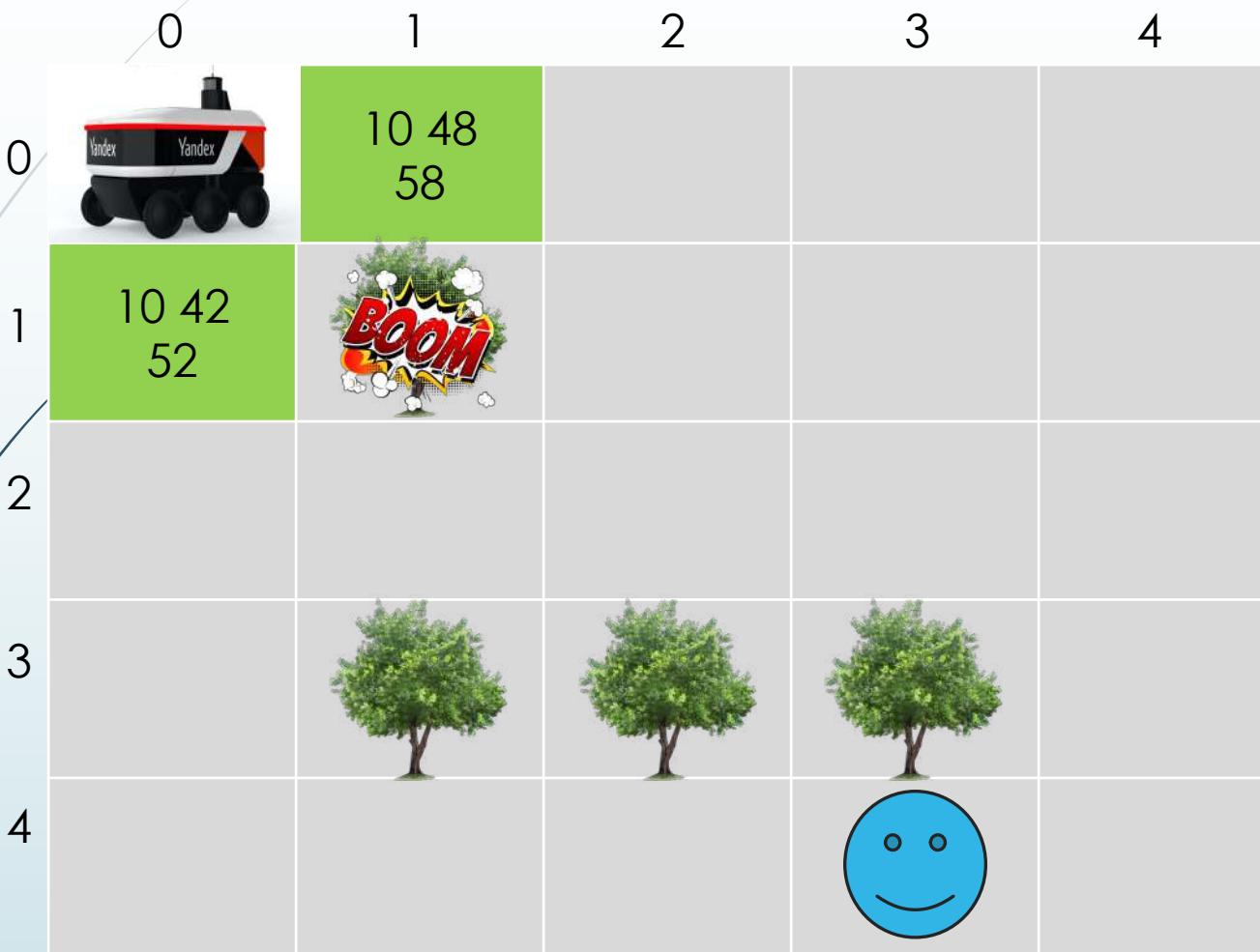
Add Initial Point to Open List and Assign Cost



CLOSED:
NULL

OPEN:
(0,0) – 0

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

(0,0) – 0 – NEW

OPEN:

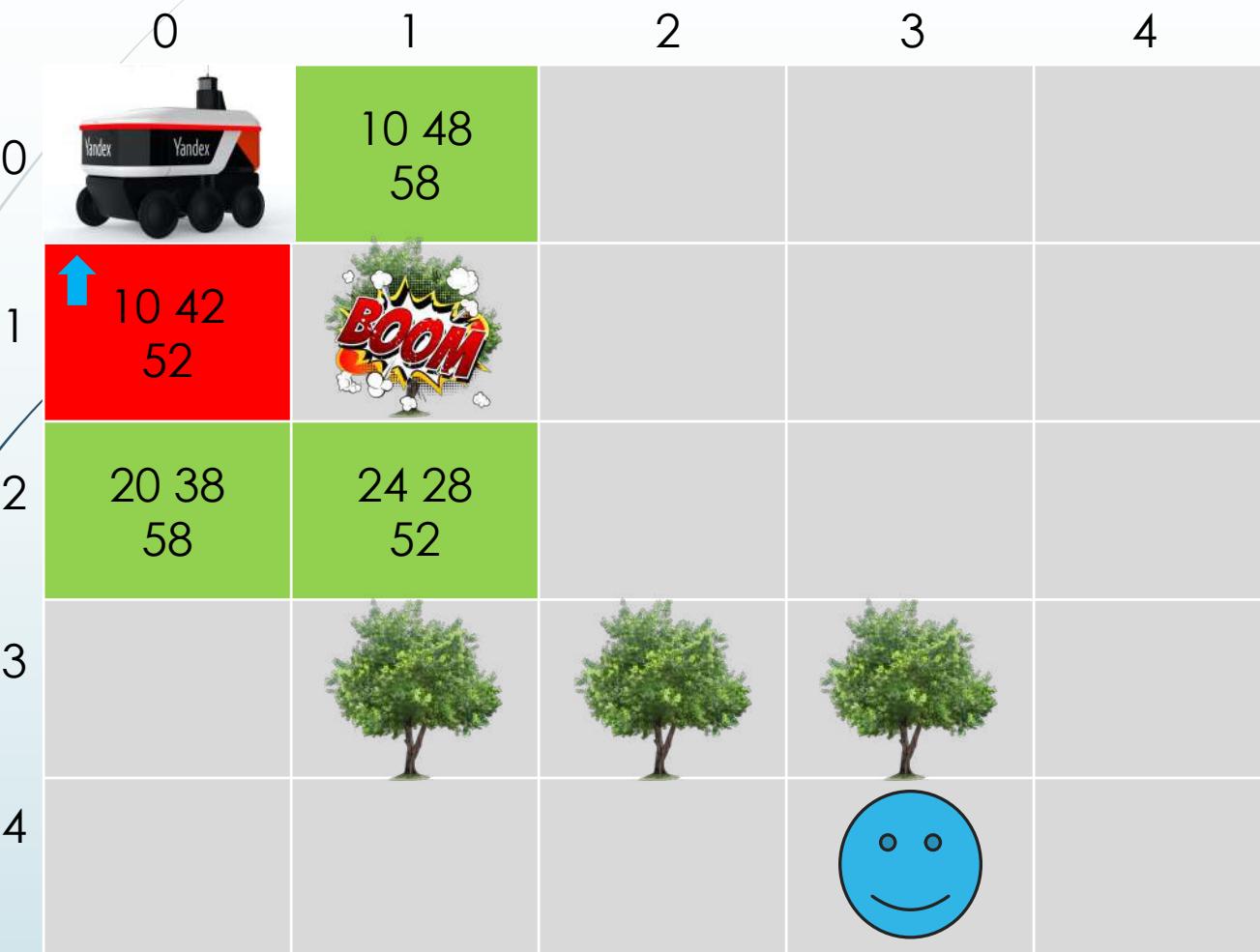
(1,0) – 52 – NEW

(0,1) – 58 – NEW

(1,1) – COLLISION

For simplicity we can assign infinitely big value to each obstacle cell

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



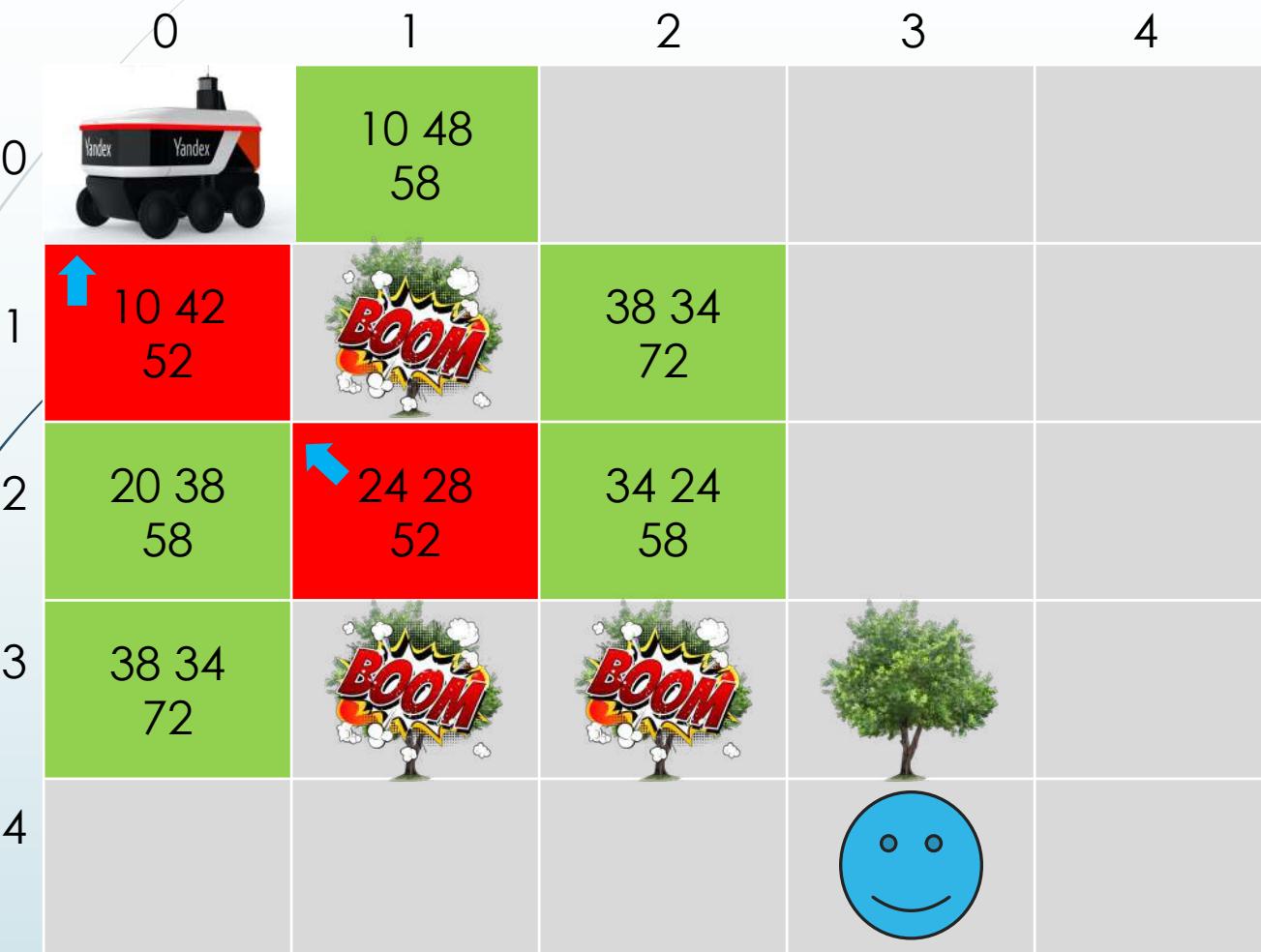
CLOSED:

(0,0) – 0
(1,0) – 52 – NEW

OPEN:

(0,1) – 58
(1,1) – ∞
(2,0) – 58 – NEW
(2,1) – 52 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



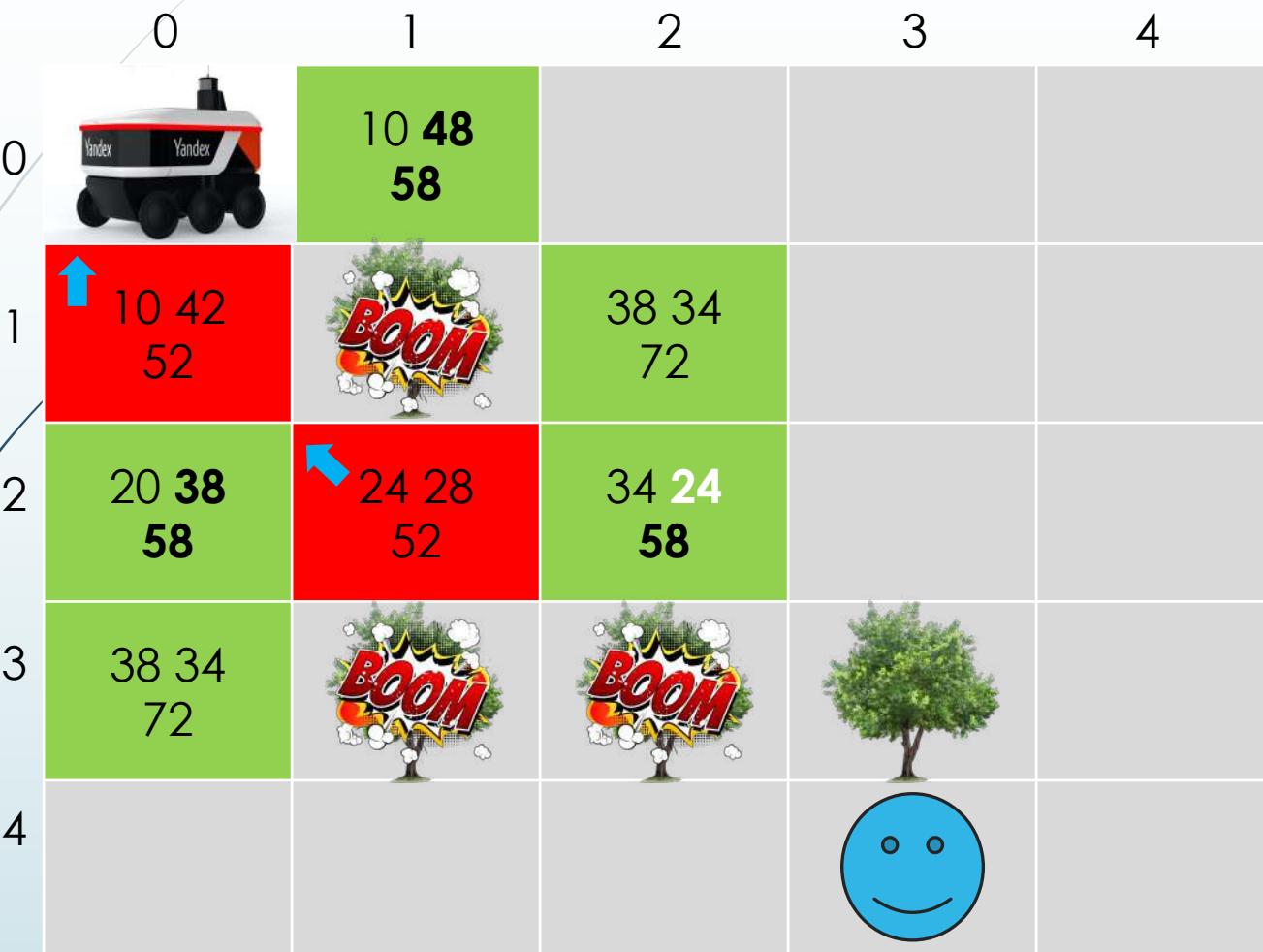
CLOSED:

(0,0) – 0
 (1,0) – 52
 (2,1) – 52 – NEW

OPEN:

(0,1) – 58
 (1,1) – ∞
 (2,0) – 58
 (3,1) – ∞ – NEW
 (3,2) – ∞ – NEW
 (3,0) – 72 – NEW
 (2,2) – 58 – NEW
 (1,2) – 72 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



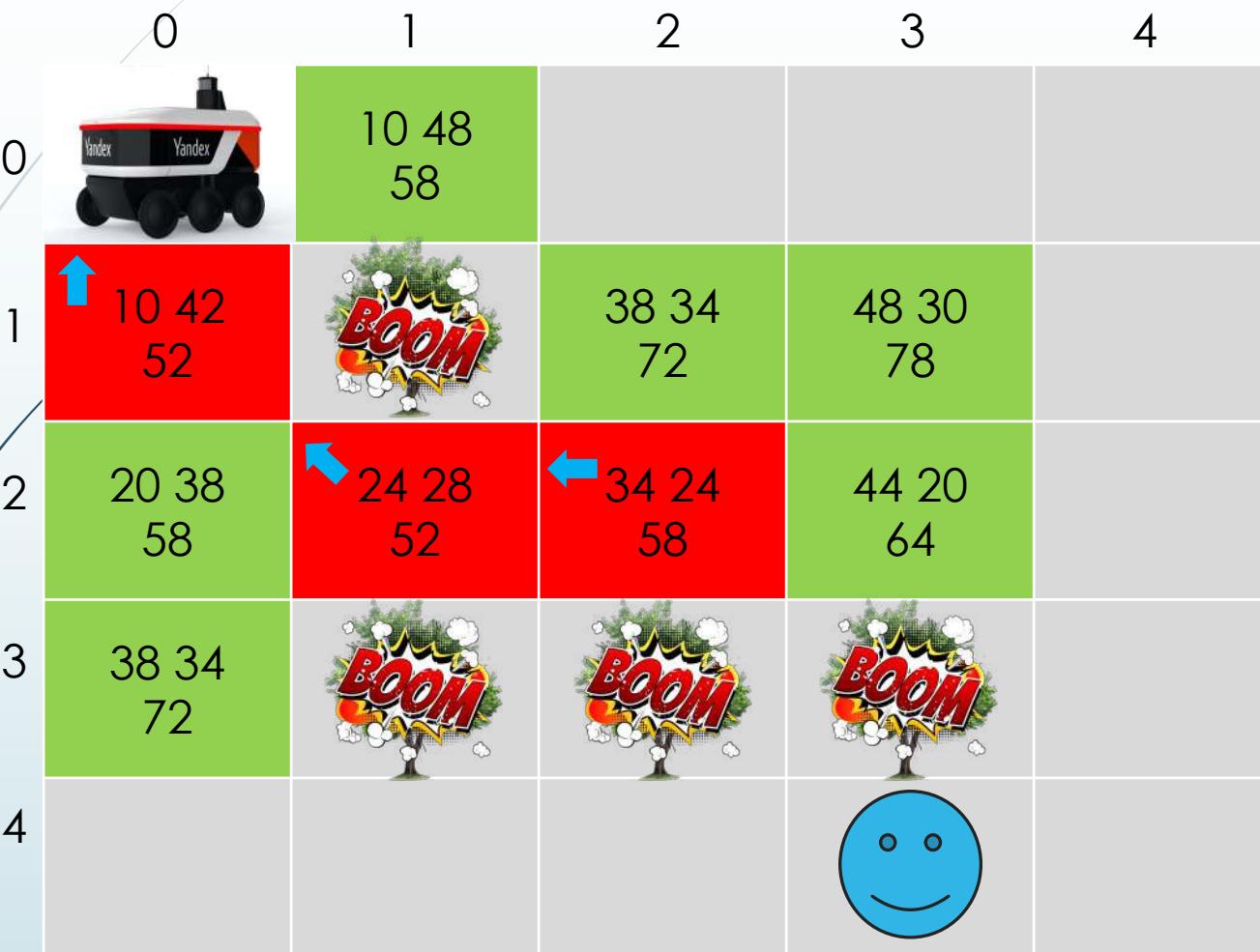
CLOSED:

(0,0) – 0
 (1,0) – 52
 (2,1) – 52 – NEW

OPEN:

(0,1) – 58
 (1,1) – ∞
 (2,0) – 58
 (3,1) – ∞ – NEW
 (3,2) – ∞ – NEW
 (3,0) – 72 – NEW
 (2,2) – 58 – NEW
 (1,2) – 72 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



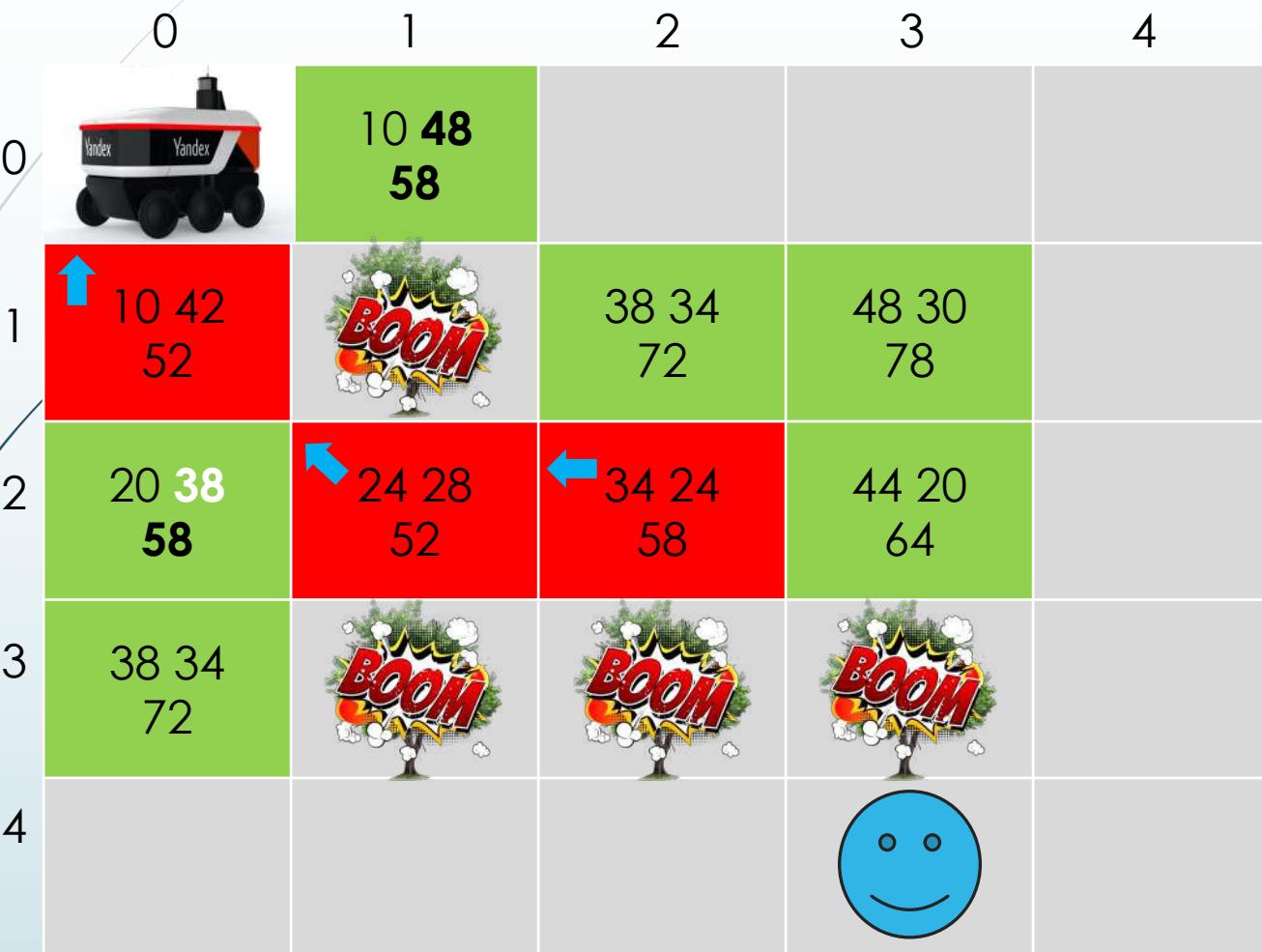
CLOSED:

$(0,0)$ – 0
 $(1,0)$ – 52
 $(2,1)$ – 52
 $(2,2)$ – 58 – NEW

OPEN:

$(0,1)$ – 58
 $(1,1)$ – ∞
 $(2,0)$ – 58
 $(3,1)$ – ∞
 $(3,2)$ – ∞
 $(3,0)$ – 72
 $(1,2)$ – 72
 $(2,3)$ – 64 – NEW
 $(1,3)$ – 78 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



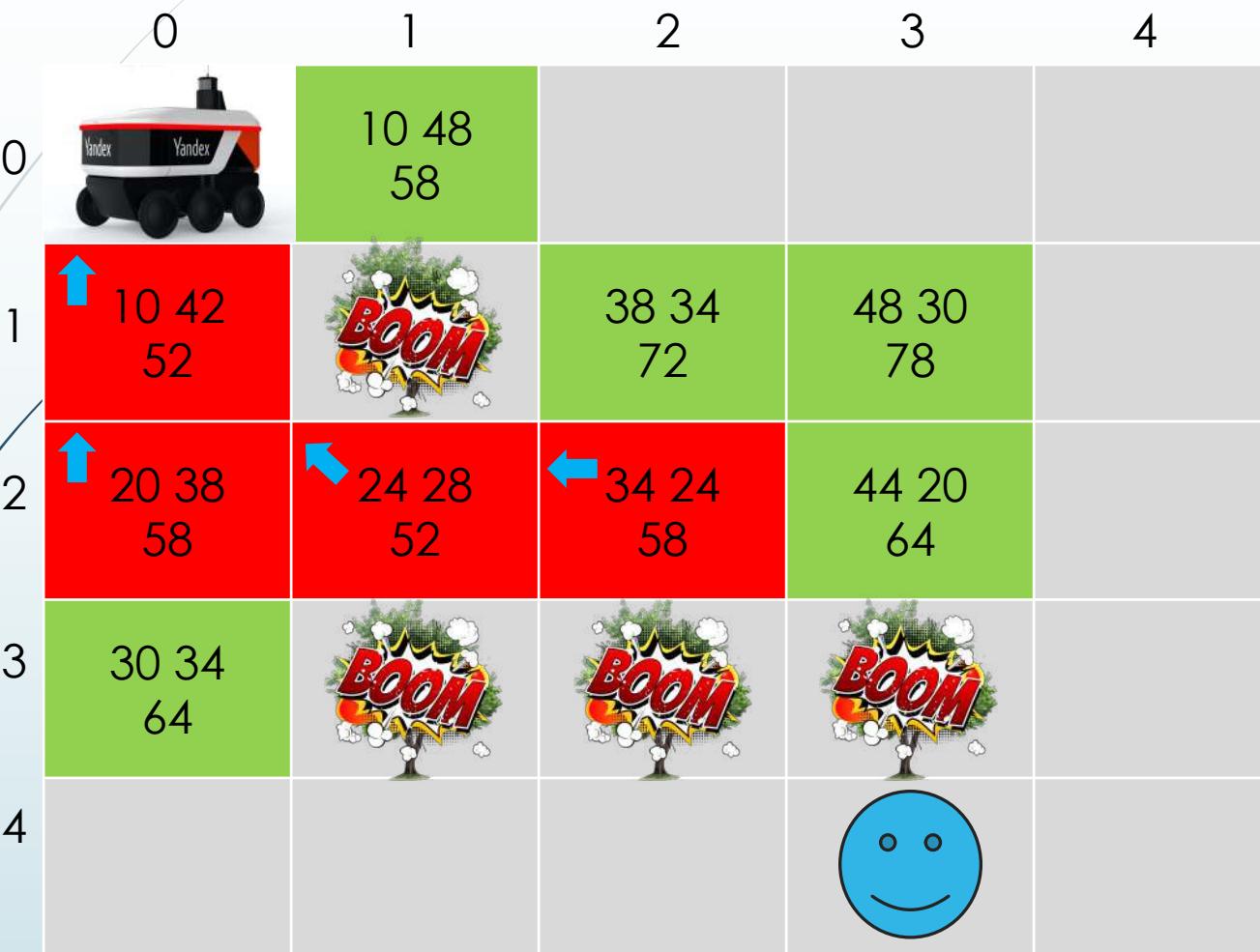
CLOSED:

$(0,0)$ – 0
 $(1,0)$ – 52
 $(2,1)$ – 52
 $(2,2)$ – 58 – NEW

OPEN:

$(0,1)$ – 58
 $(1,1)$ – ∞
 $(2,0)$ – 58
 $(3,1)$ – ∞
 $(3,2)$ – ∞
 $(3,0)$ – 72
 $(1,2)$ – 72
 $(2,3)$ – 64 – NEW
 $(1,3)$ – 78 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



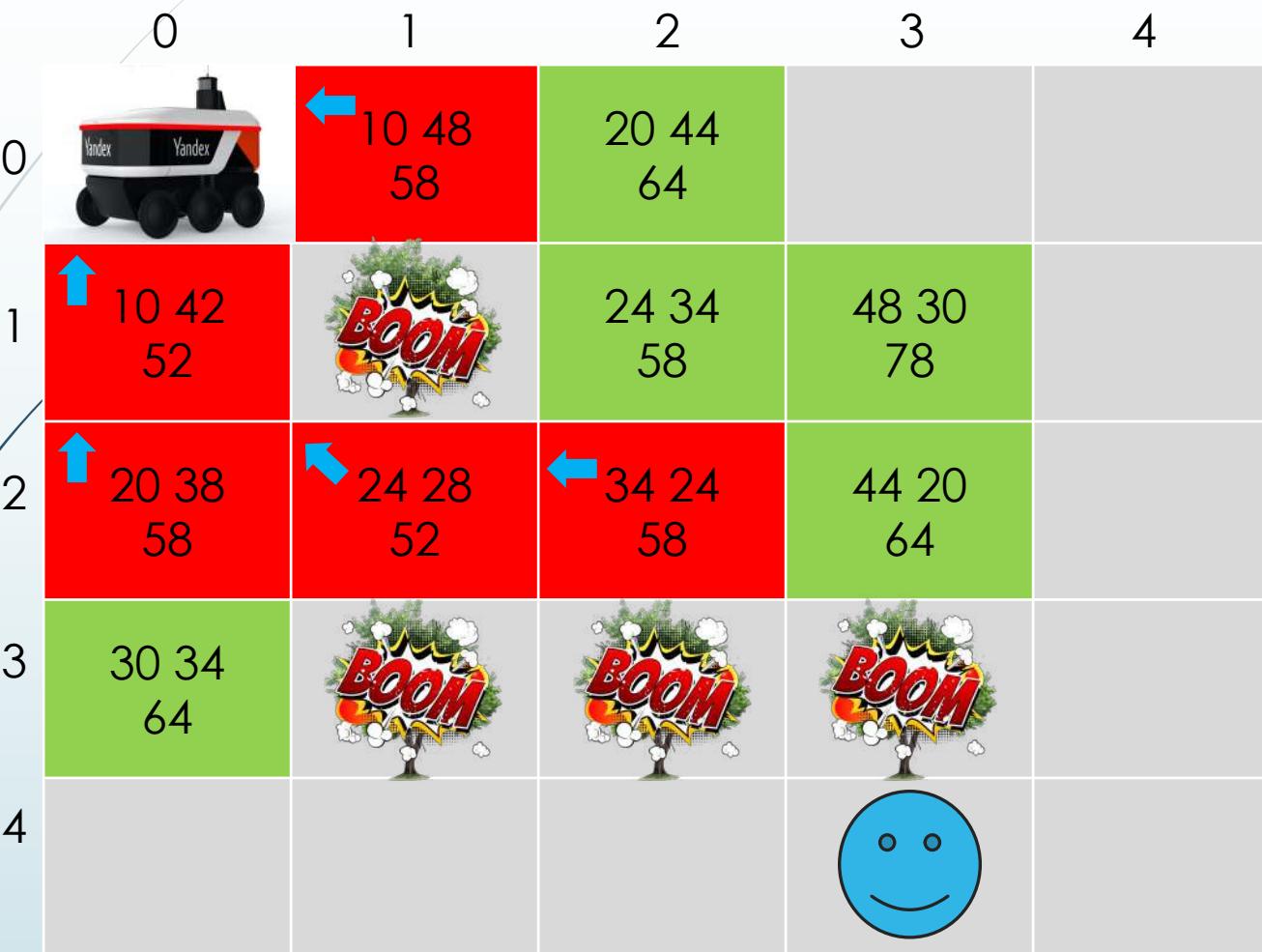
CLOSED:

$(0,0)$ – 0
 $(1,0)$ – 52
 $(2,1)$ – 52
 $(2,2)$ – 58
 $(2,0)$ – 58 - NEW

OPEN:

$(0,1)$ – 58
 $(1,1)$ – ∞
 $(3,1)$ – ∞
 $(3,2)$ – ∞
 $(3,0)$ – 64 – UPD
 $(1,2)$ – 72
 $(2,3)$ – 64
 $(1,3)$ – 78

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

$(0,0)$ – 0
 $(1,0)$ – 52
 $(2,1)$ – 52
 $(2,2)$ – 58
 $(2,0)$ – 58
 $(0,1)$ – 58 – NEW

OPEN:

$(1,1)$ – ∞
 $(3,1)$ – ∞
 $(3,2)$ – ∞
 $(3,0)$ – 64
 $(1,2)$ – 58 – UPD
 $(2,3)$ – 64
 $(1,3)$ – 78
 $(0,2)$ – 64 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

$(0,0)$ – 0
 $(1,0)$ – 52
 $(2,1)$ – 52
 $(2,2)$ – 58
 $(2,0)$ – 58
 $(0,1)$ – 58
 $(1,2)$ – 58 – NEW

OPEN:

$(1,1)$ – ∞
 $(3,1)$ – ∞
 $(3,2)$ – ∞
 $(3,0)$ – 64
 $(2,3)$ – 58 – UPD
 $(1,3)$ – 64 – UPD
 $(0,2)$ – 64
 $(0,3)$ – 78 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

(0,0) – 0

(1.0) - 52

(2.1) - 52

(22) - 58

(20) = 58

(2,0) = 68
(0,1) = 58

(0,1) 58
(1,2) 58

(2,3) – 58 – NEW

OPEN

(1,1) = ∞

(3.1) - ∞

$$(3,2) = \infty$$

$$(3,0) = 64$$

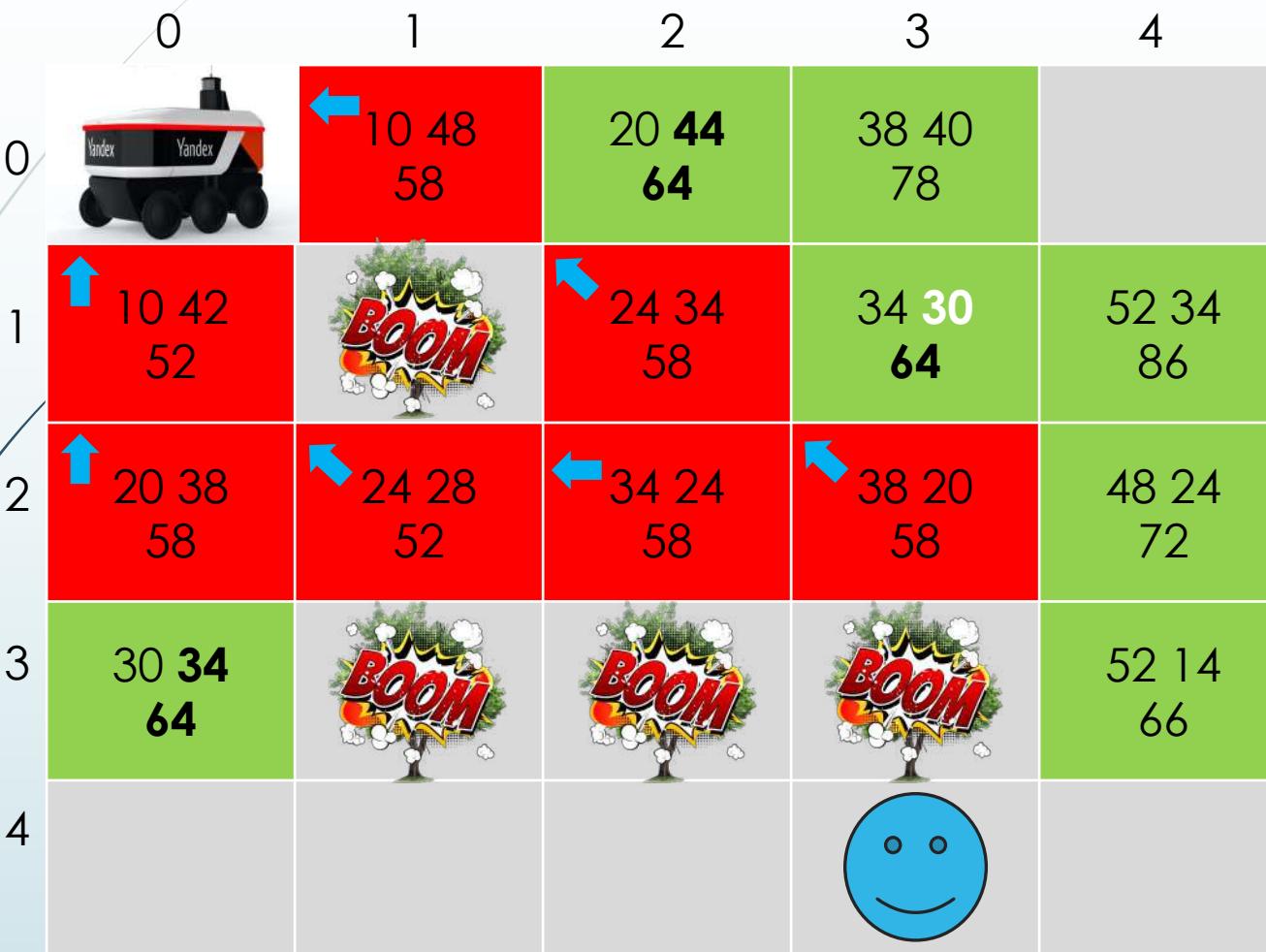
(13) - 64

(1,3) 61
(0,2) 64

(0,2) = 84
(0,3) = 78

(1,4) - 86 - NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

(0,0) – 0

(2,3) – 58 – NEW

(1,0) – 52

(2,1) – 52

(2,2) – 58

(2,0) – 58

(0,1) – 58

(1,2) – 58

OPEN:

(1,1) – ∞

(2,4) – 72 – NEW

(3,1) – ∞

(3,4) – 66 – NEW

(3,2) – ∞

(3,0) – 64

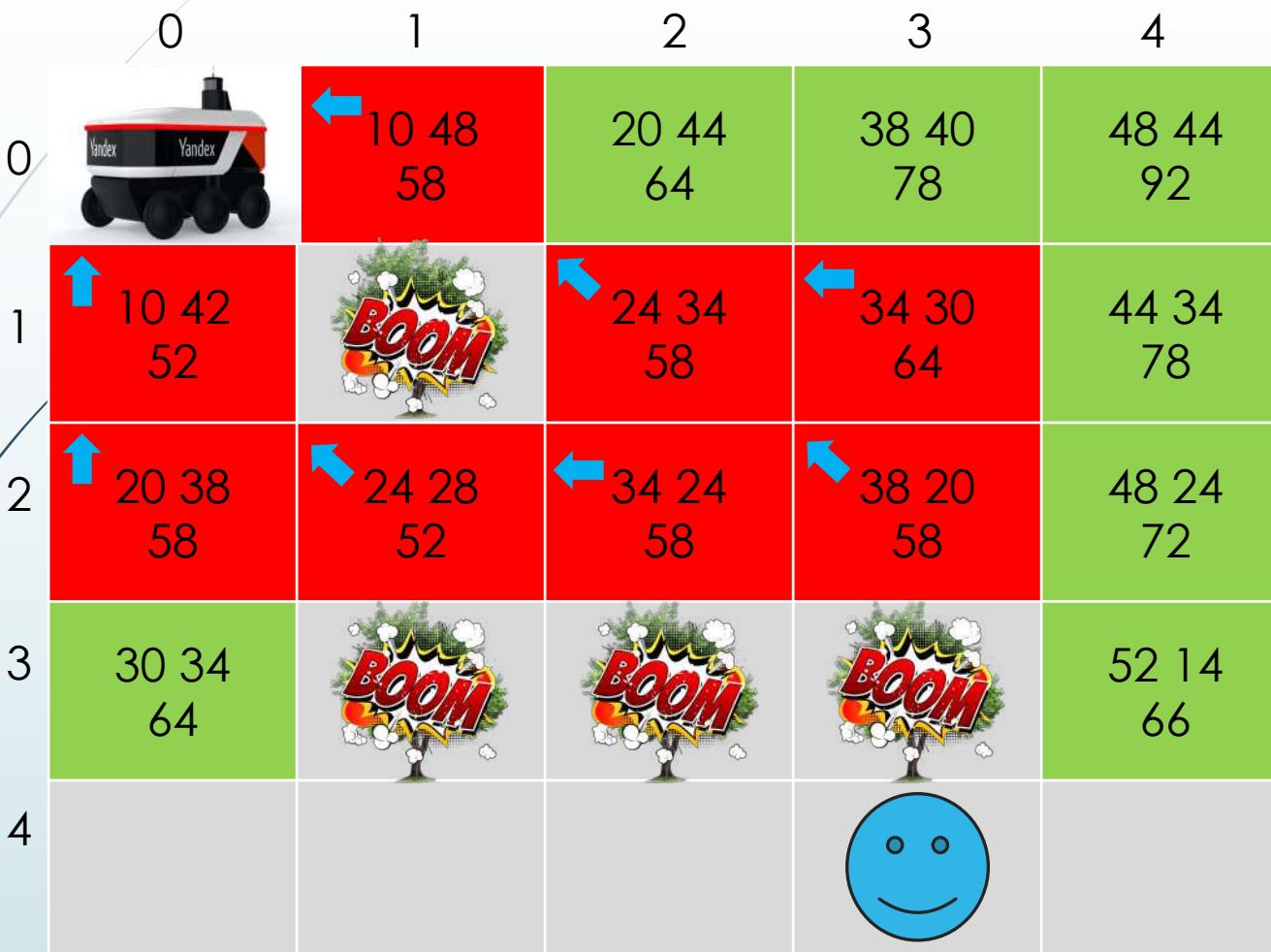
(1,3) – 64

(0,2) – 64

(0,3) – 78

(1,4) – 86 – NEW

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

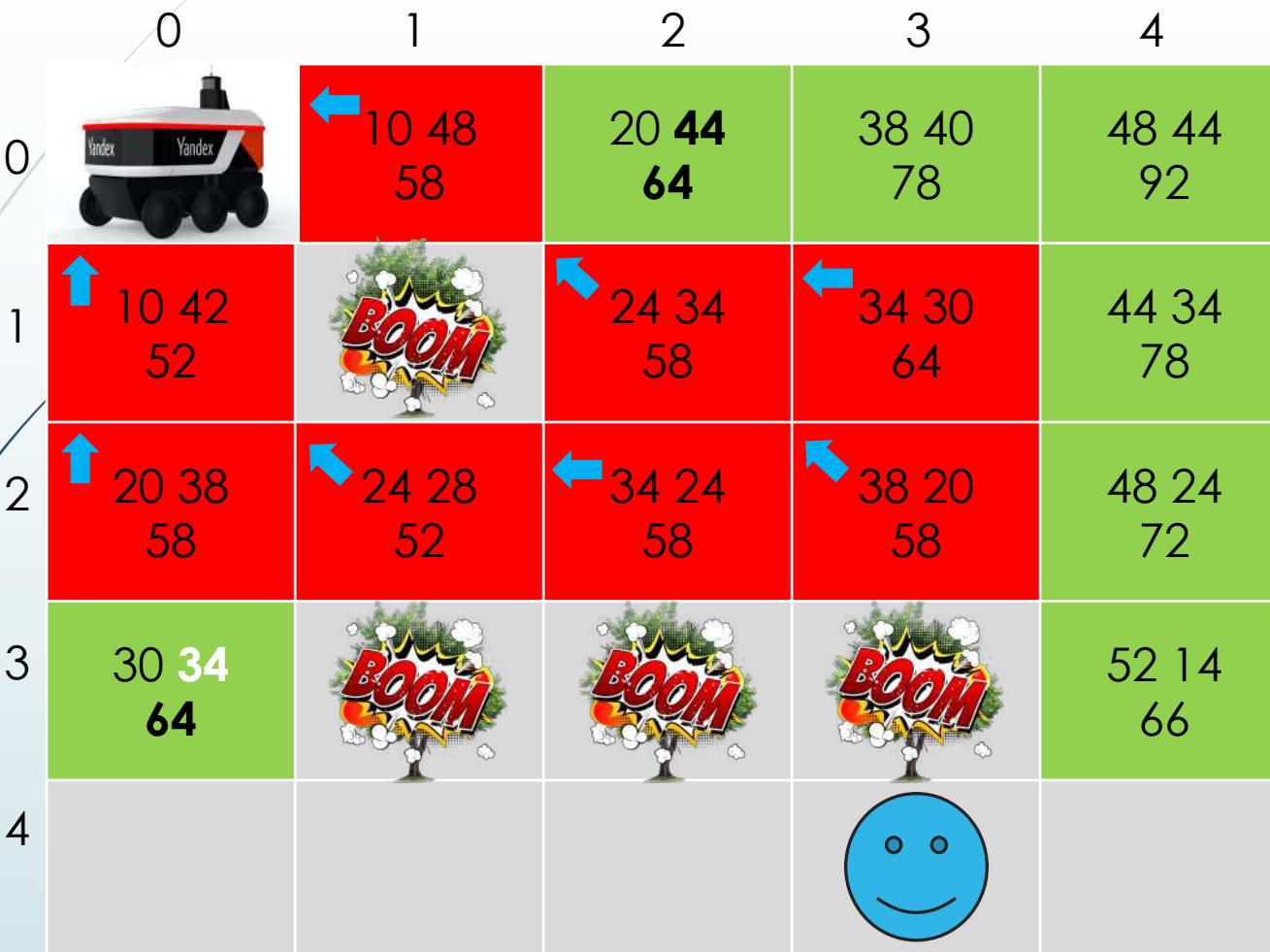
(0,0) – 0
 (1,0) – 52
 (2,1) – 52
 (2,2) – 58
 (2,0) – 58
 (0,1) – 58
 (1,2) – 58

(2,3) – 58
 (1,3) – 64 – NEW

OPEN:

(1,1) – ∞
 (3,1) – ∞
 (3,2) – ∞
 (3,0) – 64
 (0,2) – 64
 (0,3) – 78
 (1,4) – 78 – UPD
 (2,4) – 72

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

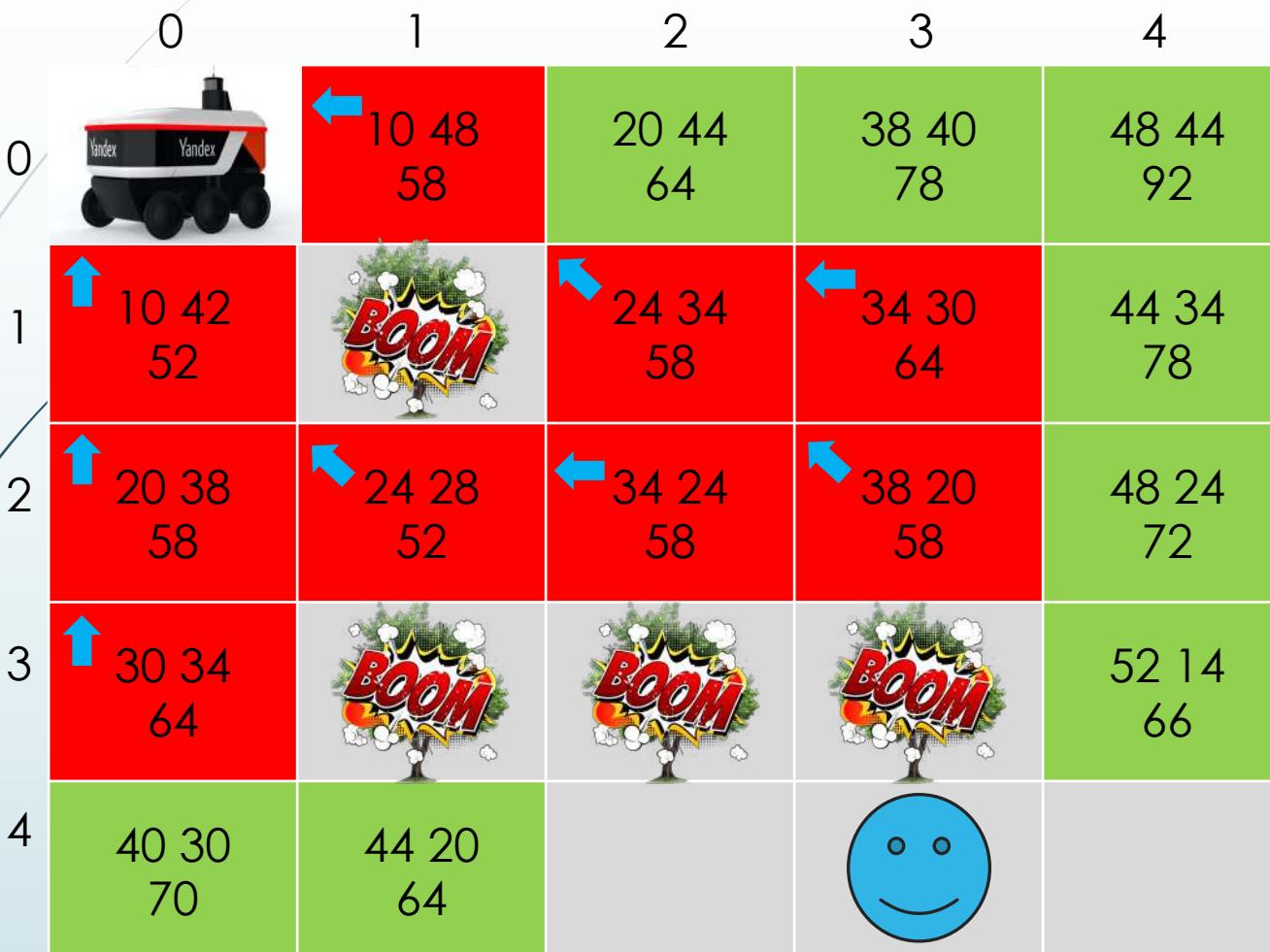
(0,0) – 0
 (1,0) – 52
 (2,1) – 52
 (2,2) – 58
 (2,0) – 58
 (0,1) – 58
 (1,2) – 58

(2,3) – 58
 (1,3) – 64 – NEW

OPEN:

(1,1) – ∞
 (3,1) – ∞
 (3,2) – ∞
 (3,0) – 64
 (0,2) – 64
 (0,3) – 78
 (1,4) – 78 – UPD
 (2,4) – 72

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



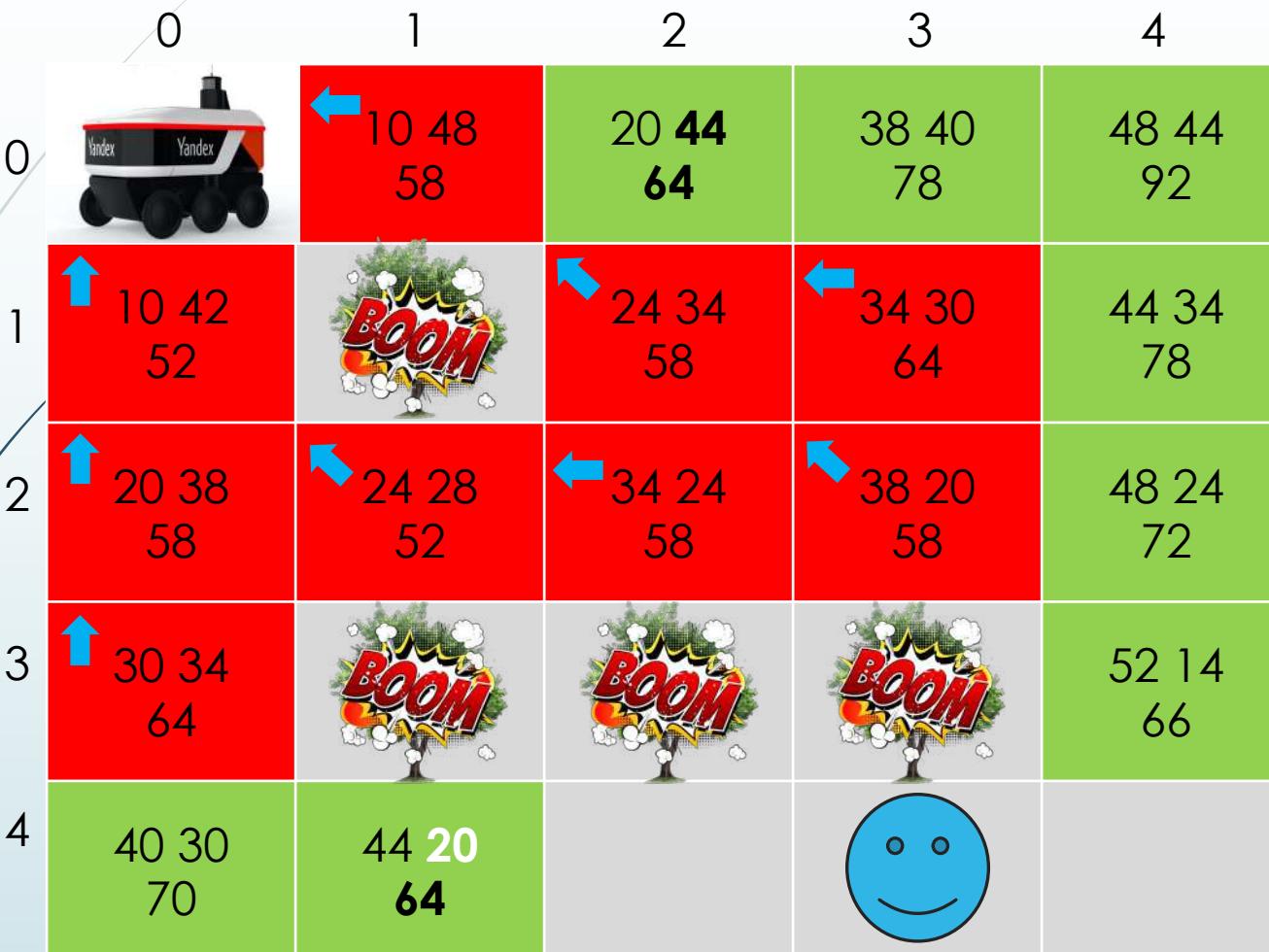
CLOSED:

(0,0) – 0	(2,3) – 58
(1,0) – 52	(1,3) – 64
(2,1) – 52	(3,0) – 64 – NEW
(2,2) – 58	
(2,0) – 58	
(0,1) – 58	
(1,2) – 58	

OPEN:

(1,1) – ∞	(0,4) – 92
(3,1) – ∞	(4,0) – 70 – NEW
(3,2) – ∞	(4,1) – 64 – NEW
(0,2) – 64	
(0,3) – 78	
(1,4) – 78	
(2,4) – 72	
(3,4) – 66	

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



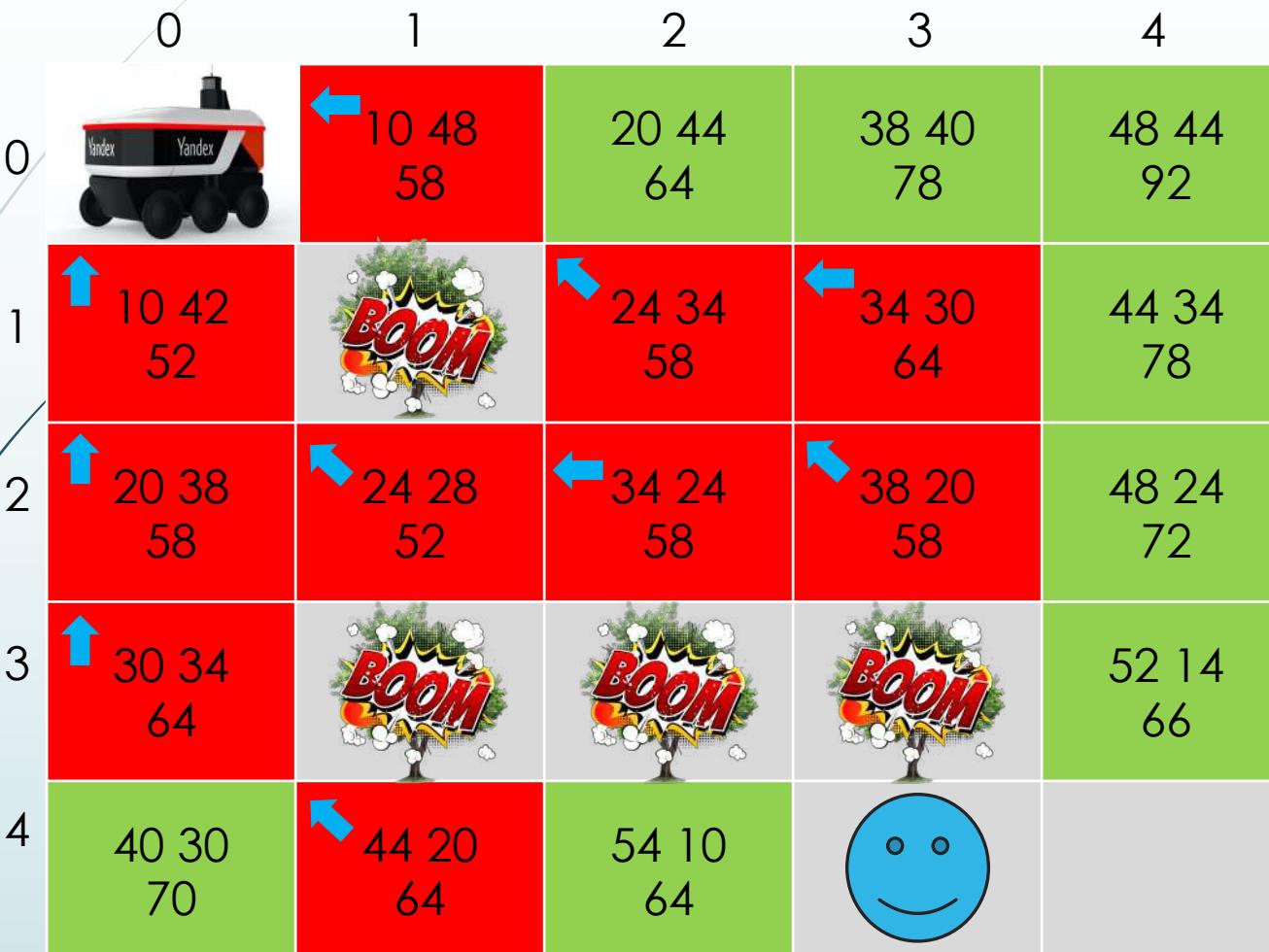
CLOSED:

$(0,0) - 0$ $(2,3) - 58$
 $(1,0) - 52$ $(1,3) - 64$
 $(2,1) - 52$ $(3,0) - 64 - \text{NEW}$
 $(2,2) - 58$
 $(2,0) - 58$
 $(0,1) - 58$
 $(1,2) - 58$

OPEN:

$(1,1) - \infty$ $(0,4) - 92$
 $(3,1) - \infty$ $(4,0) - 70 - \text{NEW}$
 $(3,2) - \infty$ $(4,1) - 64 - \text{NEW}$
 $(0,2) - 64$
 $(0,3) - 78$
 $(1,4) - 78$
 $(2,4) - 72$
 $(3,4) - 66$

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



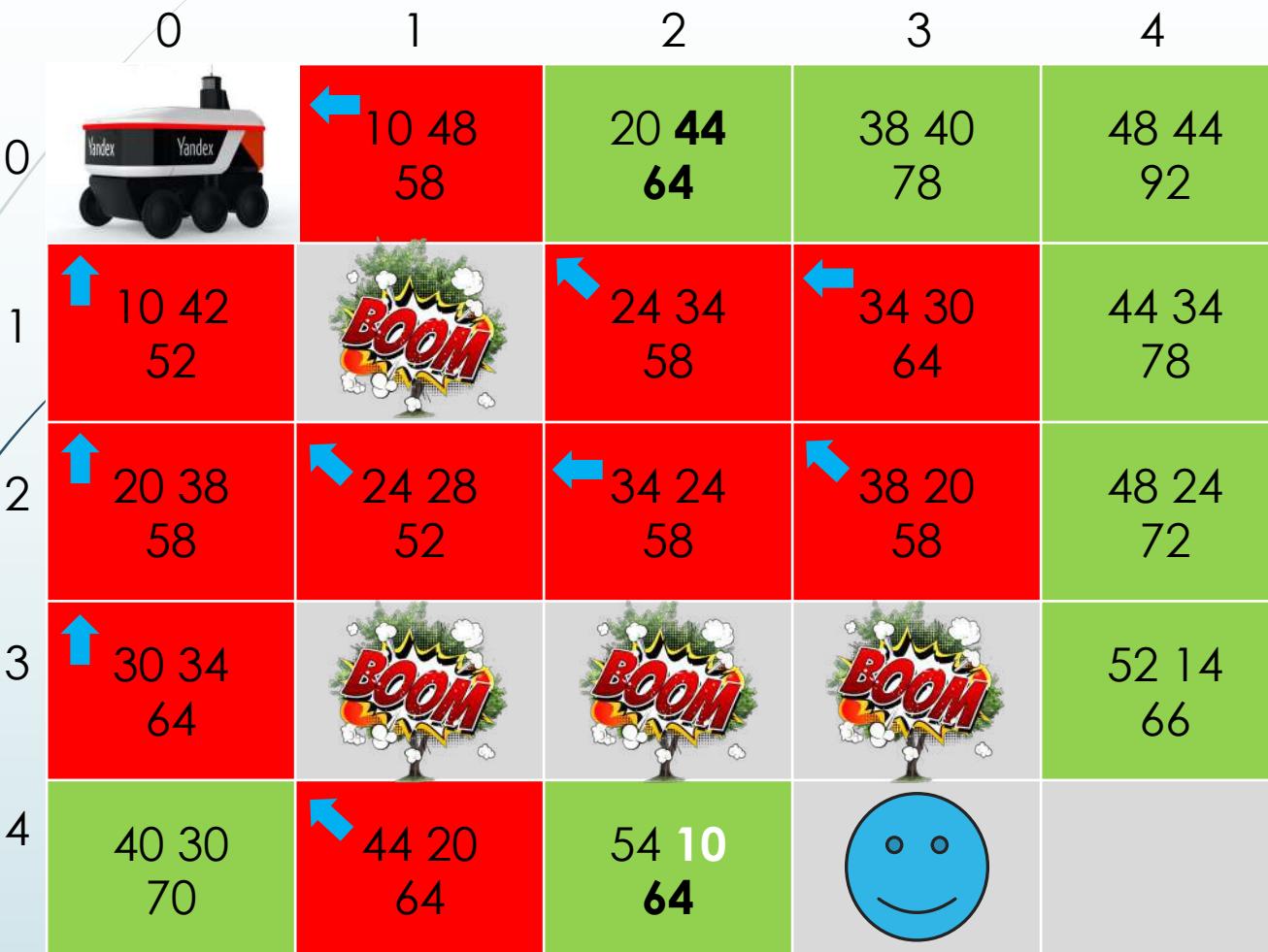
CLOSED:

(0,0) – 0	(2,3) – 58
(1,0) – 52	(1,3) – 64
(2,1) – 52	(3,0) – 64
(2,2) – 58	(4,1) – 64 – NEW
(2,0) – 58	
(0,1) – 58	
(1,2) – 58	

OPEN:

(1,1) – ∞	(0,4) – 92
(3,1) – ∞	(4,0) – 70
(3,2) – ∞	(4,2) – 64 – NEW
(0,2) – 64	
(0,3) – 78	
(1,4) – 78	
(2,4) – 72	
(3,4) – 66	

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



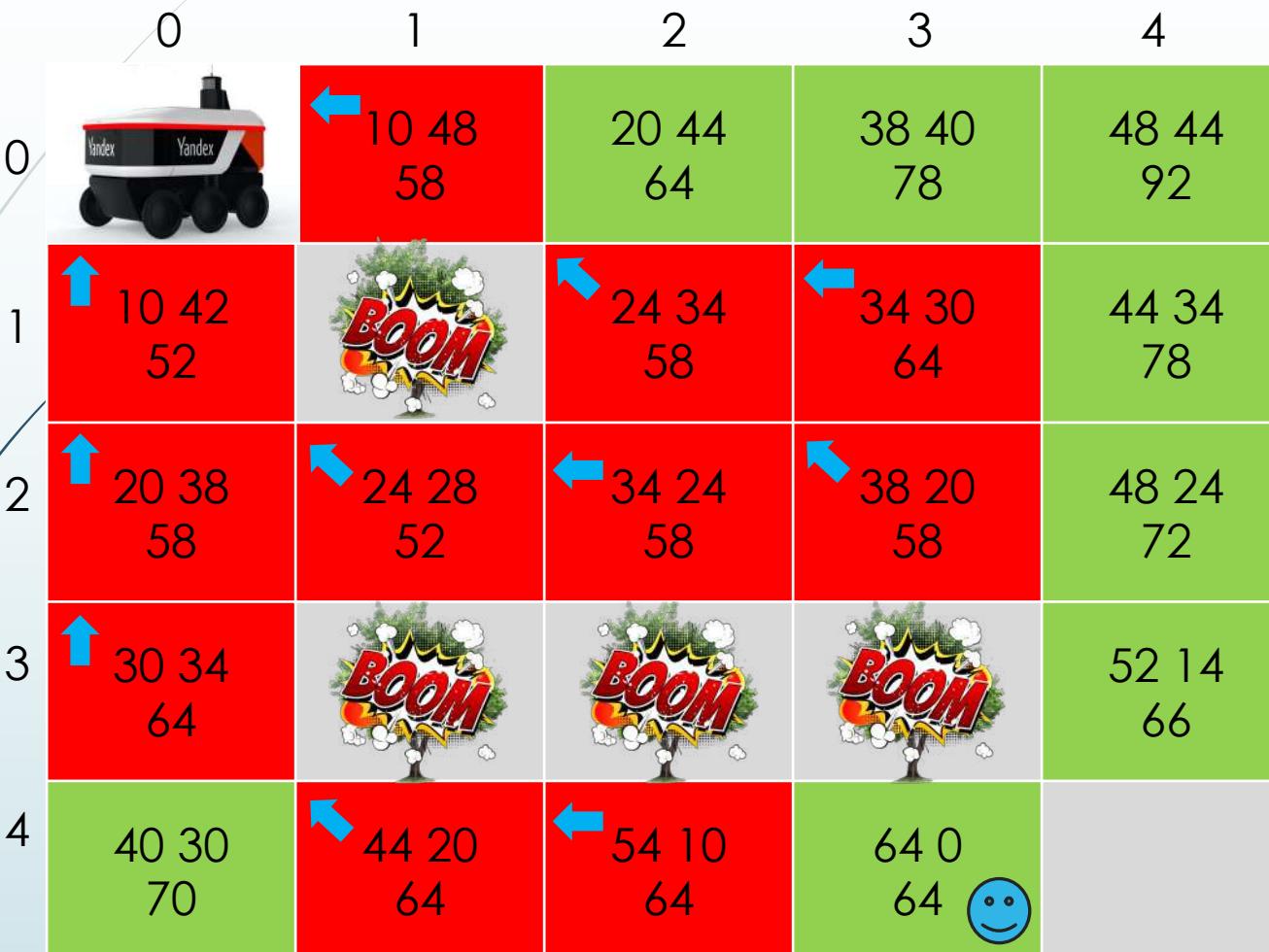
CLOSED:

(0,0) – 0	(2,3) – 58
(1,0) – 52	(1,3) – 64
(2,1) – 52	(3,0) – 64
(2,2) – 58	(4,1) – 64 – NEW
(2,0) – 58	
(0,1) – 58	
(1,2) – 58	

OPEN:

(1,1) – ∞	(0,4) – 92
(3,1) – ∞	(4,0) – 70
(3,2) – ∞	(4,2) – 64 – NEW
(0,2) – 64	
(0,3) – 78	
(1,4) – 78	
(2,4) – 72	
(3,4) – 66	

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



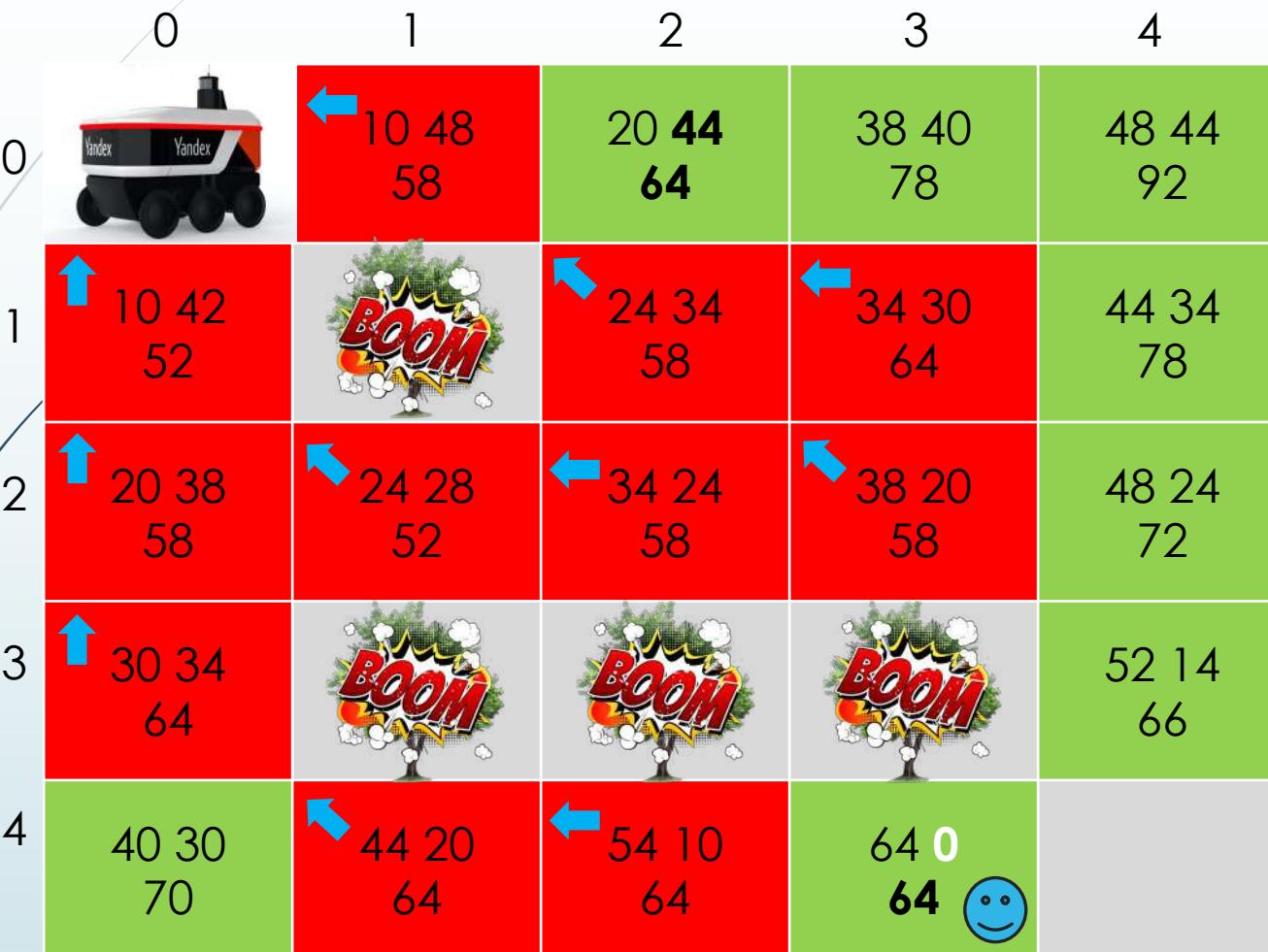
CLOSED:

(0,0) – 0	(2,3) – 58
(1,0) – 52	(1,3) – 64
(2,1) – 52	(3,0) – 64
(2,2) – 58	(4,1) – 64
(2,0) – 58	(4,2) – 64 – NEW
(0,1) – 58	
(1,2) – 58	

OPEN:

(1,1) – ∞	(0,4) – 92
(3,1) – ∞	(4,0) – 70
(3,2) – ∞	(4,3) – 64 – NEW
(0,2) – 64	
(0,3) – 78	
(1,4) – 78	
(2,4) – 72	
(3,4) – 66	

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



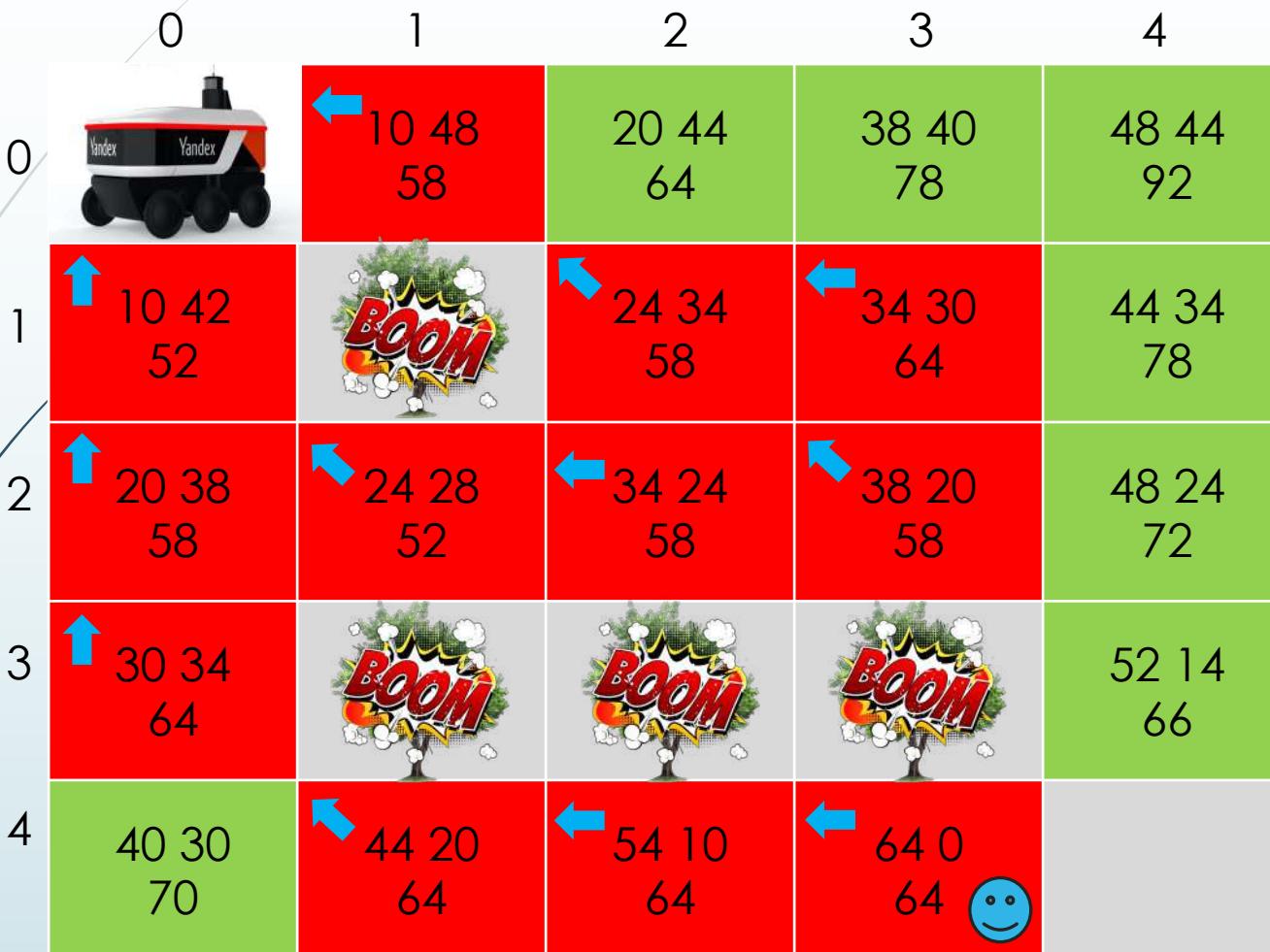
CLOSED:

(0,0) – 0	(2,3) – 58
(1,0) – 52	(1,3) – 64
(2,1) – 52	(3,0) – 64
(2,2) – 58	(4,1) – 64
(2,0) – 58	(4,2) – 64 – NEW
(0,1) – 58	
(1,2) – 58	

OPEN:

(1,1) – ∞	(0,4) – 92
(3,1) – ∞	(4,0) – 70
(3,2) – ∞	(4,3) – 64 – NEW
(0,2) – 64	
(0,3) – 78	
(1,4) – 78	
(2,4) – 72	
(3,4) – 66	

Move Lowest Cost to Closed with Smaller Heuristics and Search Neighbours



CLOSED:

(0,0) – 0	(2,3) – 58
(1,0) – 52	(1,3) – 64
(2,1) – 52	(3,0) – 64
(2,2) – 58	(4,1) – 64
(2,0) – 58	(4,2) – 64
(0,1) – 58	(4,3) – 64 – AIM
(1,2) – 58	

OPEN:

(1,1) – ∞	(0,4) – 92
(3,1) – ∞	(4,0) – 70
(3,2) – ∞	
(0,2) – 64	
(0,3) – 78	
(1,4) – 78	
(2,4) – 72	
(3,4) – 66	

Goal is Found

	0	1	2	3	4
0		10 48 58	20 44 64	38 40 78	48 44 92
1	↑ 10 42 52		24 34 58	34 30 64	44 34 78
2	↑ 20 38 58	24 28 52	34 24 58	38 20 58	48 24 72
3	↑ 30 34 64				52 14 66
4	40 30 70	44 20 64	54 10 64	64 0 64	

CLOSED:

- (0,0) – 0
- (1,0) – 52
- (2,1) – 52
- (2,2) – 58
- (2,0) – 58
- (0,1) – 58
- (1,2) – 58
- (2,3) – 58
- (1,3) – 64
- (3,0) – 64
- (4,1) – 64
- (4,2) – 64
- (4,3) – 64 – AIM

OPEN:

- (1,1) – ∞
- (3,1) – ∞
- (3,2) – ∞
- (0,2) – 64
- (0,3) – 78
- (1,4) – 78
- (2,4) – 72
- (3,4) – 66
- (0,4) – 92
- (4,0) – 70

Improvements of A*

- ▶ Dealing with other units
 - ▶ Treat stopped units as unpassable
 - ▶ Treat moving units nearby as unpassable
 - ▶ More complexity – prediction of their movement for avoidance – query their path
- ▶ Cost of movement
 - ▶ Taken into account in the cost function
- ▶ Non-grid setting
- ▶ Open World
 - ▶ Enforce a grid or graph
 - ▶ Develop pathways explicitly for longer movements

Tabu Search

- ▶ Glover 1980s
- ▶ Idea coming from a social tabu – never return to where I once was
 - ▶ Heuristic – where I have been is not where I want to go
- ▶ Search is conducted by single point changes in the candidate solution with a list of previous places that the search was at, a Tabu List
- ▶ The tabus are feature sets of the parameters
- ▶ Searches locally, then places the locality on the tabu areas of the search – forcing it to move away

Tabu Advantages and Disadvantages

- ▶ Advantages:
 - ▶ Accepts non-improving solutions to escape from local optimum
 - ▶ Can be applied to both discrete and continuous solution spaces
- ▶ Disadvantages:
 - ▶ Too many parameters to be determined
 - ▶ Number of iterations may be very large
 - ▶ Global optimum is not guaranteed to be found

Tabu Loop

- ▶ Set a current position, set best to this position, set a tabu list to null
- ▶ While not at a stopping condition
 - ▶ Create a candidate list from neighbours
 - ▶ Check if the candidates are on the tabu list. If they are, remove them
 - ▶ Find the best candidate
 - ▶ Move to this as current position
 - ▶ Test the best candidate against the best
 - ▶ If it is the new best – replace
 - ▶ Create a tabu by looking at the differences in the features between this new best and the previous best
 - ▶ Produce a tabu restriction based on this difference
 - ▶ Add this tabu to the tabu list
 - ▶ Check tabu list length and remove old tabus

Example - Tabu

- ▶ String of 5 parameters – containing numbers 0-9
- ▶ Tabu List is *,*,3,*,*
- ▶ New Tabus are created by adding one point where the best strings matched
- ▶ Best is 0,3,3,2,1 scoring 14 (the way of scoring is omitted for simplicity)
- ▶ Local neighbourhood is +/-1 from each parameter
- ▶ Generate the new list
 - ▶ 0,2,3,2,0 - 17
 - ▶ 0,3,4,2,1 - 18
 - ▶ 0,4,4,2,0 - 20

Compare and Add Tabu

- ▶ **0,3,3,**2**,1** – previous best (score 14)
- ▶ **0,4,4,**2**,0** – new best (score 20)

Compare and Add Tabu

- ▶ **0,3,3,**2**,1** – previous best (score 14)
- ▶ **0,4,4,**2**,0** – new best (score 20)
- ▶ 0,*,*,2,*
- ▶ Add the tabu of 0,*,*,*,* or *,*,*,2,*
- ▶ New Tabu list of *,*,3,*,* and (0,*,*,*,* or *,*,*,2,*)

Issues with Too Many Tabus

- ▶ Small sets of tabus might make it impossible to move
- ▶ In our example, with 3 tabus all happening about the same value, it would lock movement completely (lock upper and lower move, then move between and make that tabu)
- ▶ Rationalize the Tabus – Costly
- ▶ Only allow a small enough number of Tabus – where is this necessary value? Is it too low? How often will it happen?
- ▶ Aspiration criteria – Accept Tabu moves if they SIGNIFICANTLY increase fitness

Summary

- ▶ Simulated annealing
- ▶ Exploration and exploitation
- ▶ A*
- ▶ Tabu search

References

- ▶ [A* pathfinding](#)
- ▶ [Tabu search](#)



Introduction to Artificial Intelligence

Week 6

Traveling Salesman Problem (TSP)

- "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"

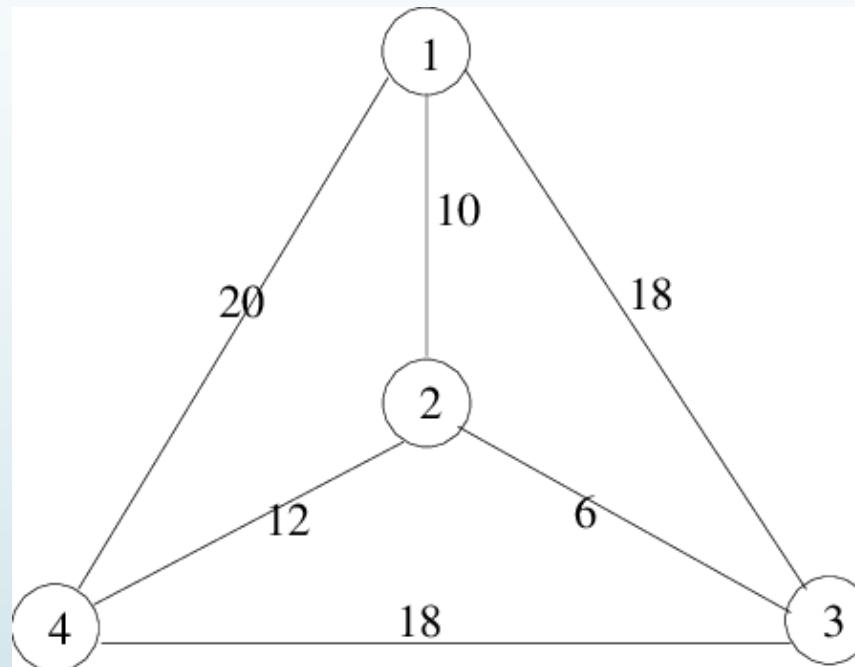


Traveling Salesman Problem (2)

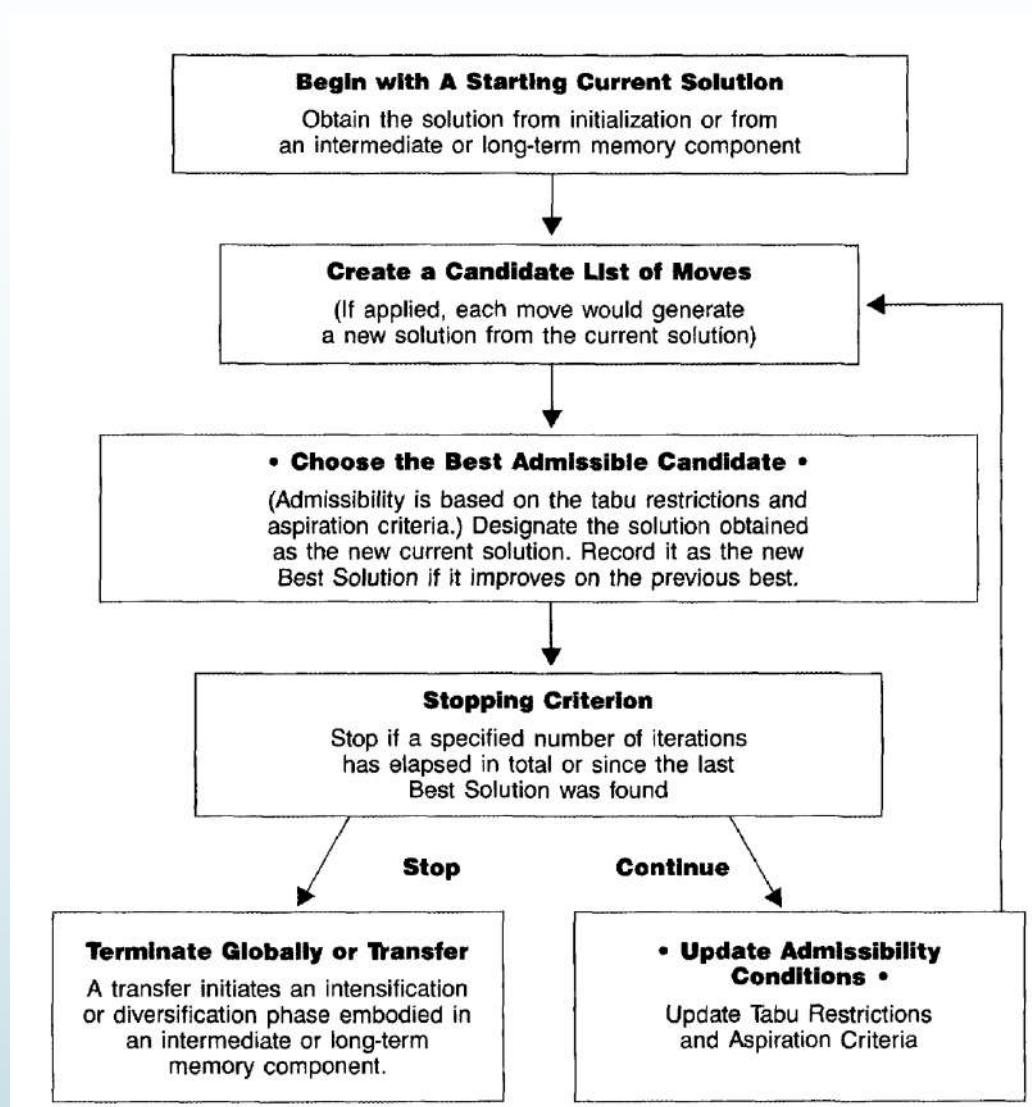
	C1	C2	C3	C4	C5
C1	0	132	217	164	158
C2	132	0	290	201	79
C3	217	290	0	113	303
C4	164	201	113	0	196
C5	158	79	303	196	0

The solution can be written as C2->C5->C1->C4->C3->C2. An itinerary that begins and ends at the same city and visits each city once is called a tour.

TSP – Graph Representation



Tabu Search



TSP – Tabu Search

Suppose the initial solution to our problem is the shown as:

2	5	1	4	3	2
---	---	---	---	---	---

The ordering specifies that city 2 is placed in the first position, followed by city 5, etc. Tabu search methods operate to identify the neighbourhood that can be reached from any current solution. Identifying moves that lead from one solution to the next. In our problem, a swap exchanges the position of two cities.

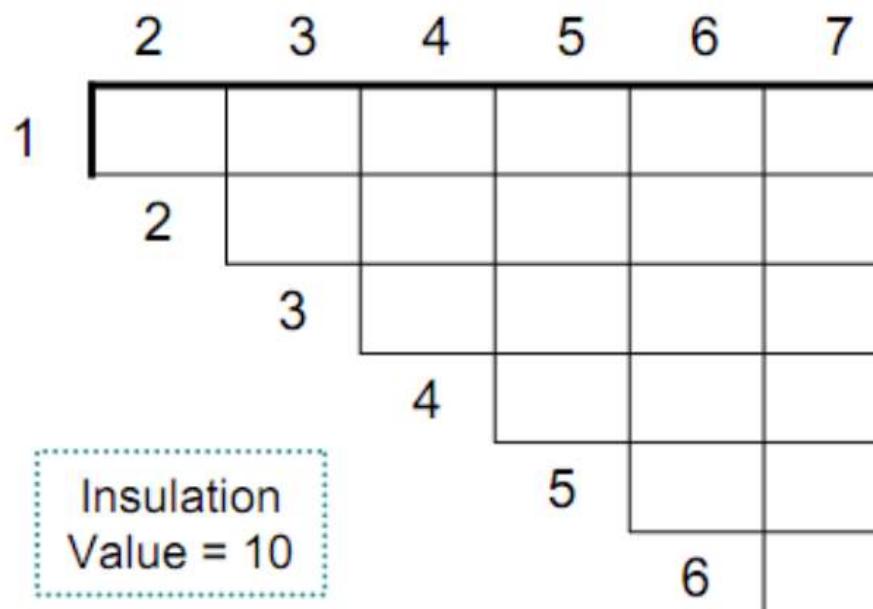
2	4	1	5	3	2
---	---	---	---	---	---

Resolution for the TSP using Tabu Search

Current
Solution

2
5
7
3
4
6
1

Tabu
Structure



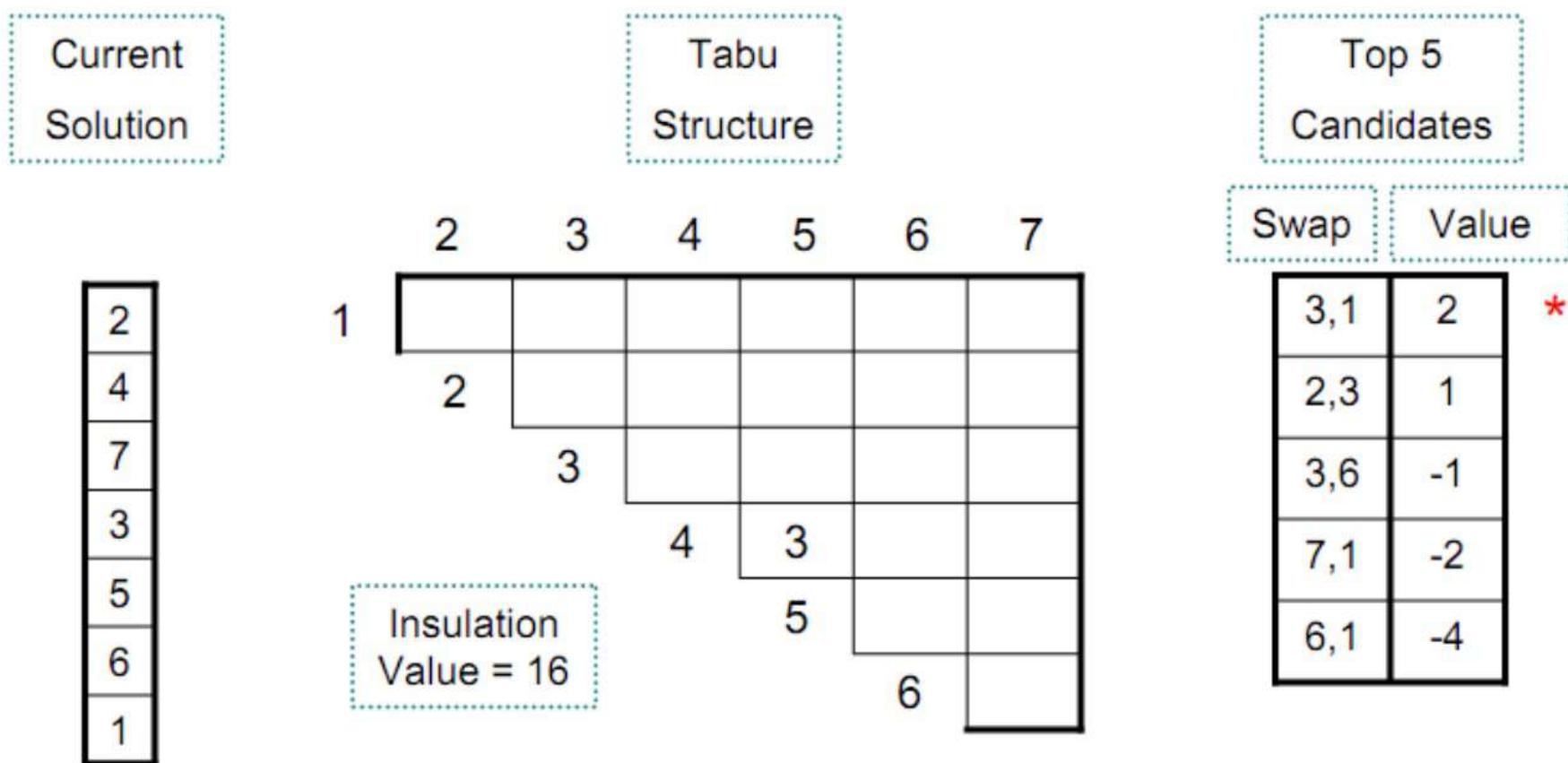
Top 5
Candidates

Swap Value

5,4	6
7,4	4
3,6	2
2,3	0
4,1	-1

*

Resolution for the TSP using Tabu Search (2)

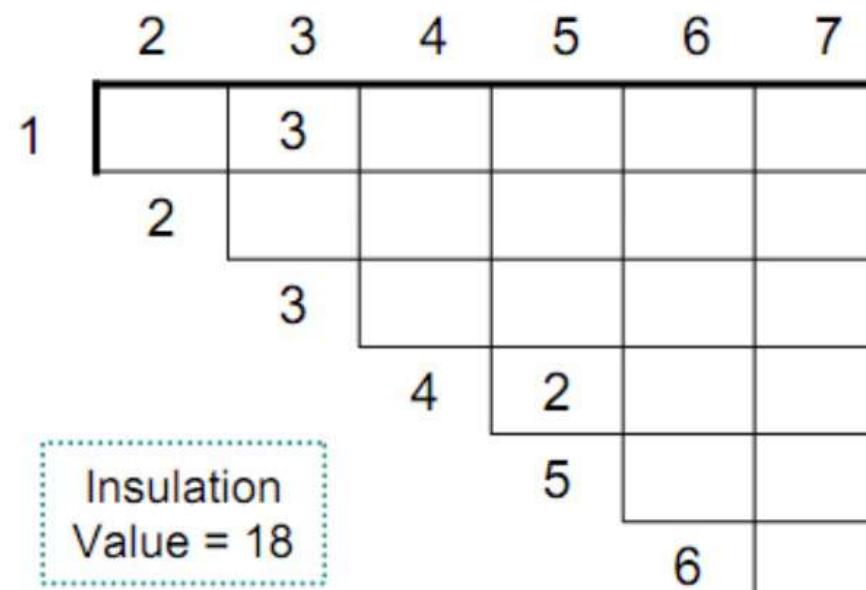


Resolution for the TSP using Tabu Search (3)

Current
Solution

2
4
7
1
5
6
3

Tabu
Structure

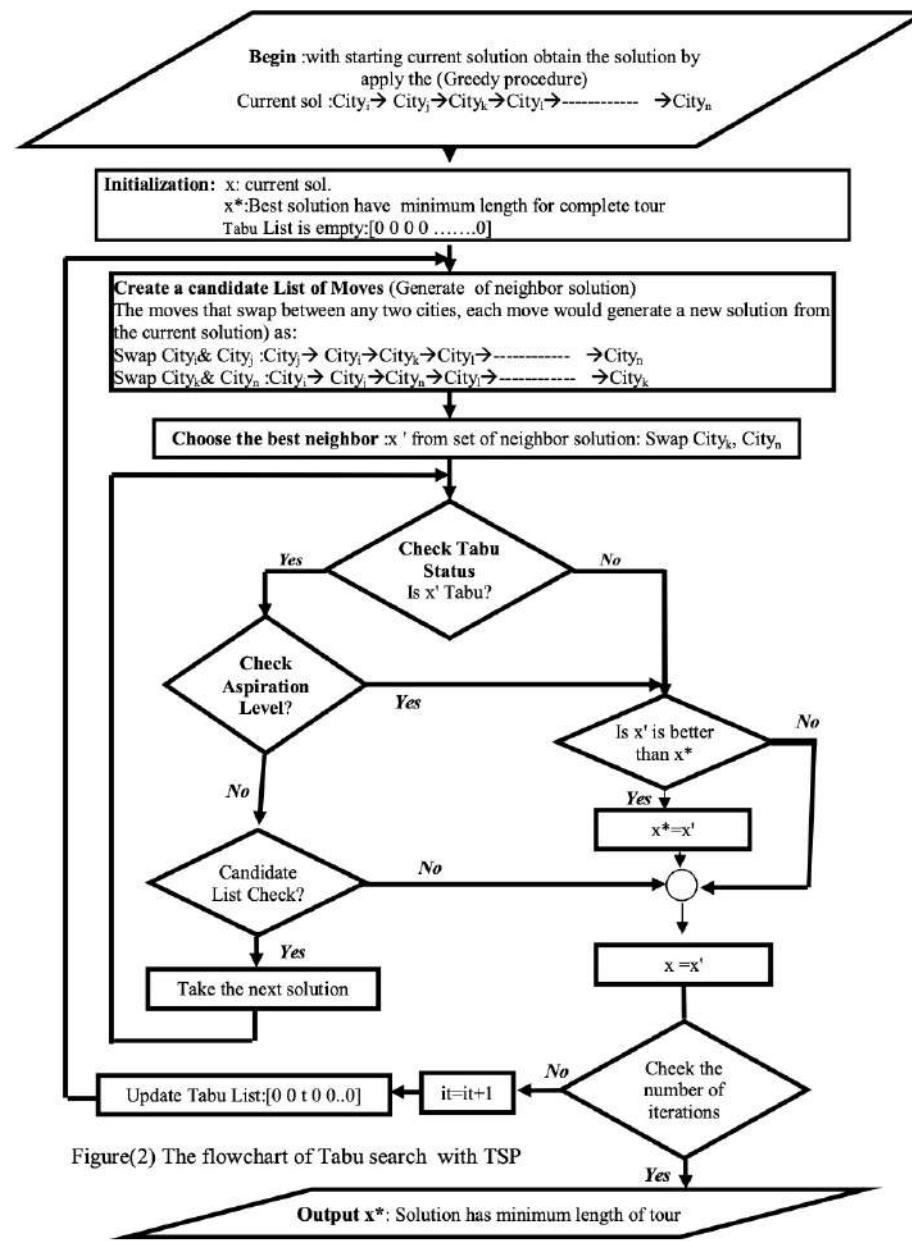


Top 5
Candidates

Swap	Value	
1,3	-2	T
2,4	-4	*
7,6	-6	
4,5	-7	T
5,3	-9	

SOLUTION

Tabu Search Method for Solving the Travel salesmen Problem



Figure(2) The flowchart of Tabu search with TSP

Output x^* : Solution has minimum length of tour

Tabu Search Characteristics

Aspiration Criterion

In order to override the tabu list when there is a good tabu move, aspiration criterion is used, the tabu move is accepted if it produces better solution than the best obtained so far.

Termination criterion

The most commonly used stopping criteria in tabu search are:

- 1- Terminate the search after number of iterations.
- 2- After some number of iterations without an improvement in the objective function value.
- 3- When the objective reaches a pre-specified threshold value. In this research the first criteria is used for stopping criteria.

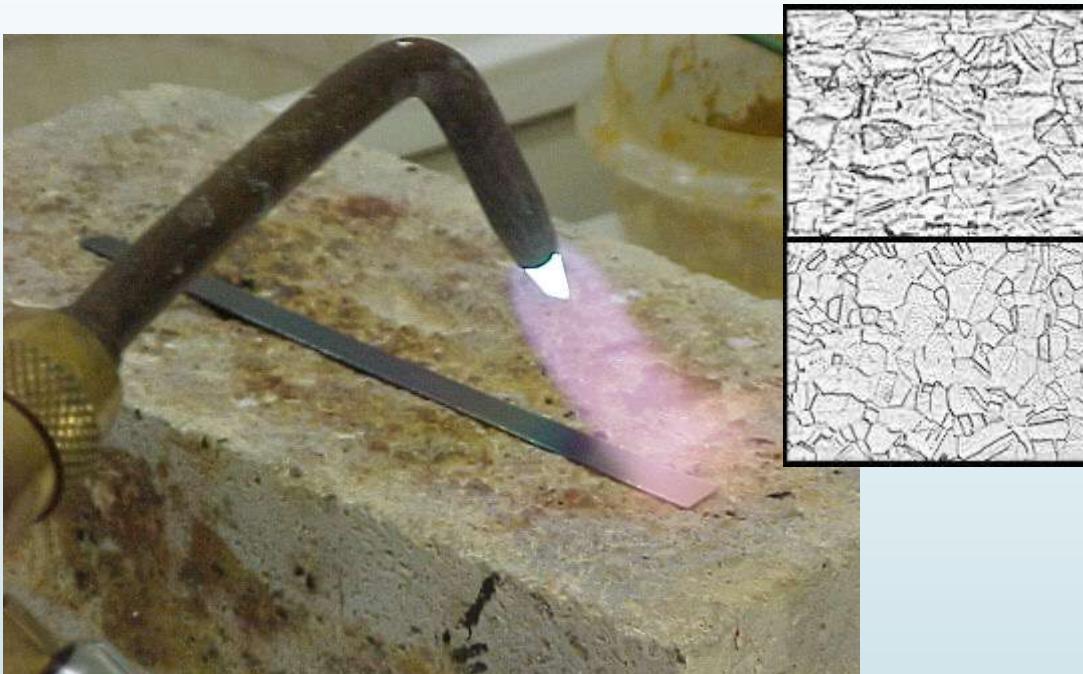
Travelling Salesman Problem – Tabu Search (Python Code)

► DEMO:

<https://smartmobilityalgorithms.github.io/book/content/TrajectoryAlgorithms/TabuSearch.html>

Simulated Annealing

- ▶ Based on the properties of Metal working
- ▶ Metal is heated and cooled in order for it to be made workable



Simulated Annealing

- ▶ Start at an initial temperature value (usually something hot) and create a random initial solution
- ▶ Until temperature is cool
 - ▶ Make a small change to the current solution (neighbour)
 - ▶ Decide if we move to this neighbour solution
 - ▶ Neighbour solution is better or
 - ▶ $\text{Rand_Distribution}() < \text{temp}$
 - ▶ Decrease the temperature

Simulated Annealing

1. Start with a random tour through the selected cities.
2. Pick a new candidate tour at random from all neighbors of the existing tour. One way to pick a neighboring tour “is to choose two cities on the tour randomly, and then reverse the portion of the tour that lies between them.” This candidate tour might be better or worse compared to the existing tour, i.e. shorter or longer.
3. If the candidate tour is better than the existing tour, accept it as the new tour.
4. If the candidate tour is worse than the existing tour, still maybe accept it, according to some probability. The probability of accepting an inferior tour is a function of how much longer the candidate is compared to the current tour, and the temperature of the annealing process. A higher temperature makes you more likely to accept an inferior tour.
5. Go back to step 2 and repeat many times, lowering the temperature a bit at each iteration, until you get to a low temperature and arrive at your (hopefully global, possibly local) minimum. If you’re not sufficiently satisfied with the result, try the process again, perhaps with a different temperature cooling schedule.

Travelling Salesman Problem - Simulated Annealing (Pseudocode)

- ▶ **SIMULATED- ANNEALING**(*source, destination, schedule, num_iterations*) **return** a route
current \leftarrow random route from *source* to *destination*
for *num_iterations* do
 T \leftarrow *schedule()*
 neighbours \leftarrow *current.NEIGHBOURS*
 next \leftarrow randomly choose one state from *neighbours*
 $\Delta E \leftarrow$ *next.COST* - *current.COST*
 if $\Delta E < 0$ then *current* \leftarrow *next*
 else *current* \leftarrow *next* only with probability of $e^{-\Delta E/T}$
 route \leftarrow *current*
return *route*

Travelling Salesman Problem - Simulated Annealing (Python Code)

- DEMO:

<https://smartmobilityalgorithms.github.io/book/content/TrajectoryAlgorithms/SimulatedAnnealing.html>

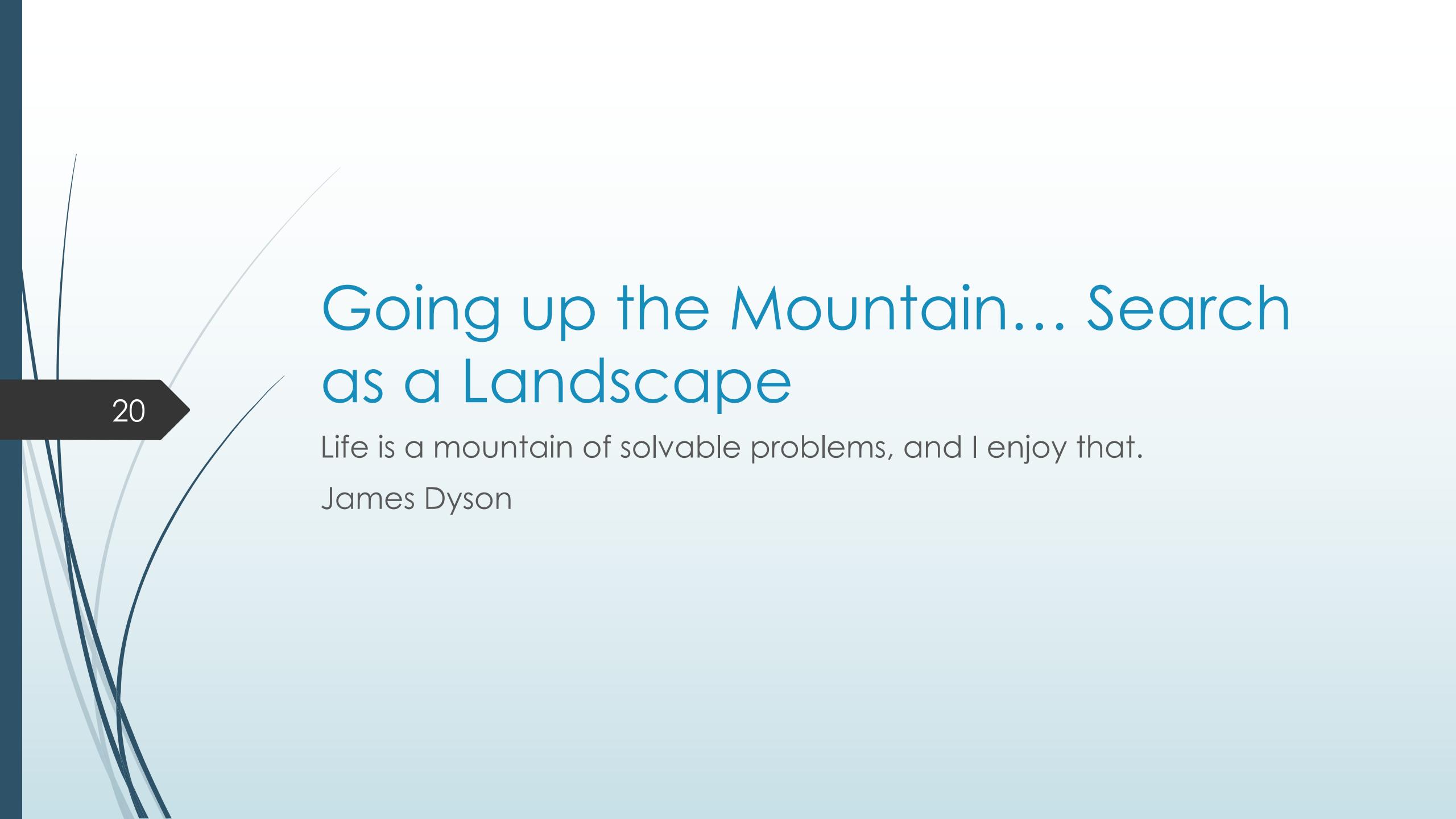
<https://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/>

Tabu search vs. semi-deterministic approach

- While tabu search uses a semi-deterministic approach to escape local optima, simulated annealing uses a complete stochastic approach

Visualization of metaheuristics for the TSP

► <https://www.youtube.com/watch?v=rYjxtmt8g9A>



20

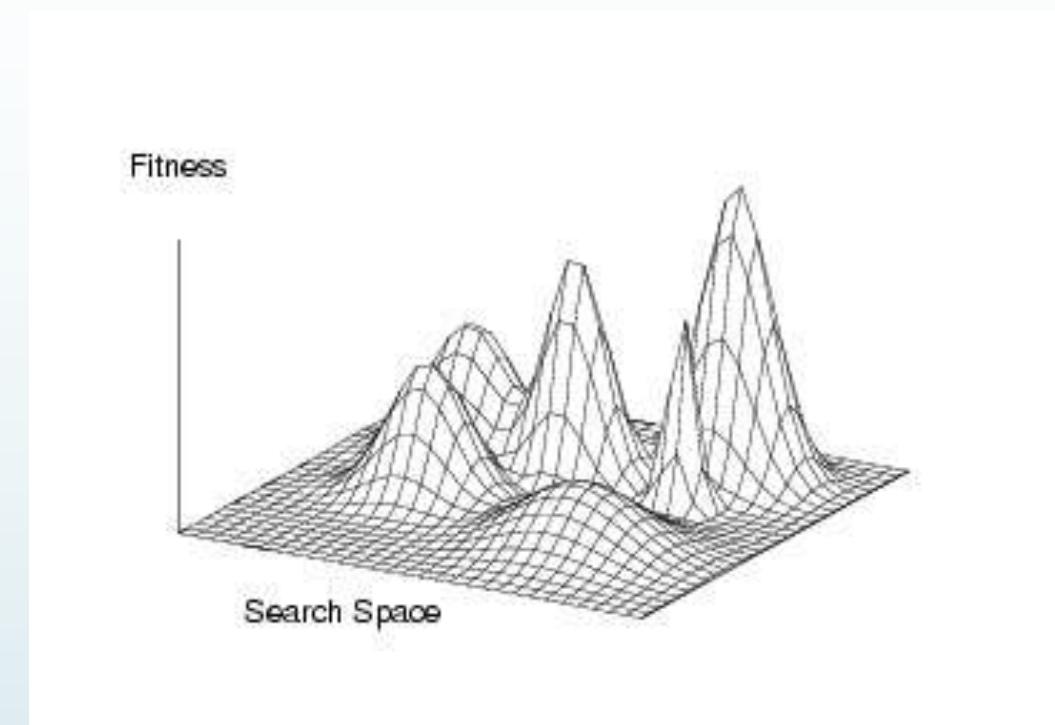
Going up the Mountain... Search as a Landscape

Life is a mountain of solvable problems, and I enjoy that.

James Dyson

Search Space

- ▶ We have already heard the term Search Space
- ▶ The tuple collection of parameter inputs and outputs for those parameters
- ▶ Can we see this as some other shape?



Fitness Landscapes

- ▶ Evolutionary biologist Sewall Wright came up with the idea of fitness landscapes back in 1932. His maps displayed reproductive success, or fitness, of individual organisms as a function of genotype (an organism's genetic constitution) or phenotype (or organism's observable physical characteristics)

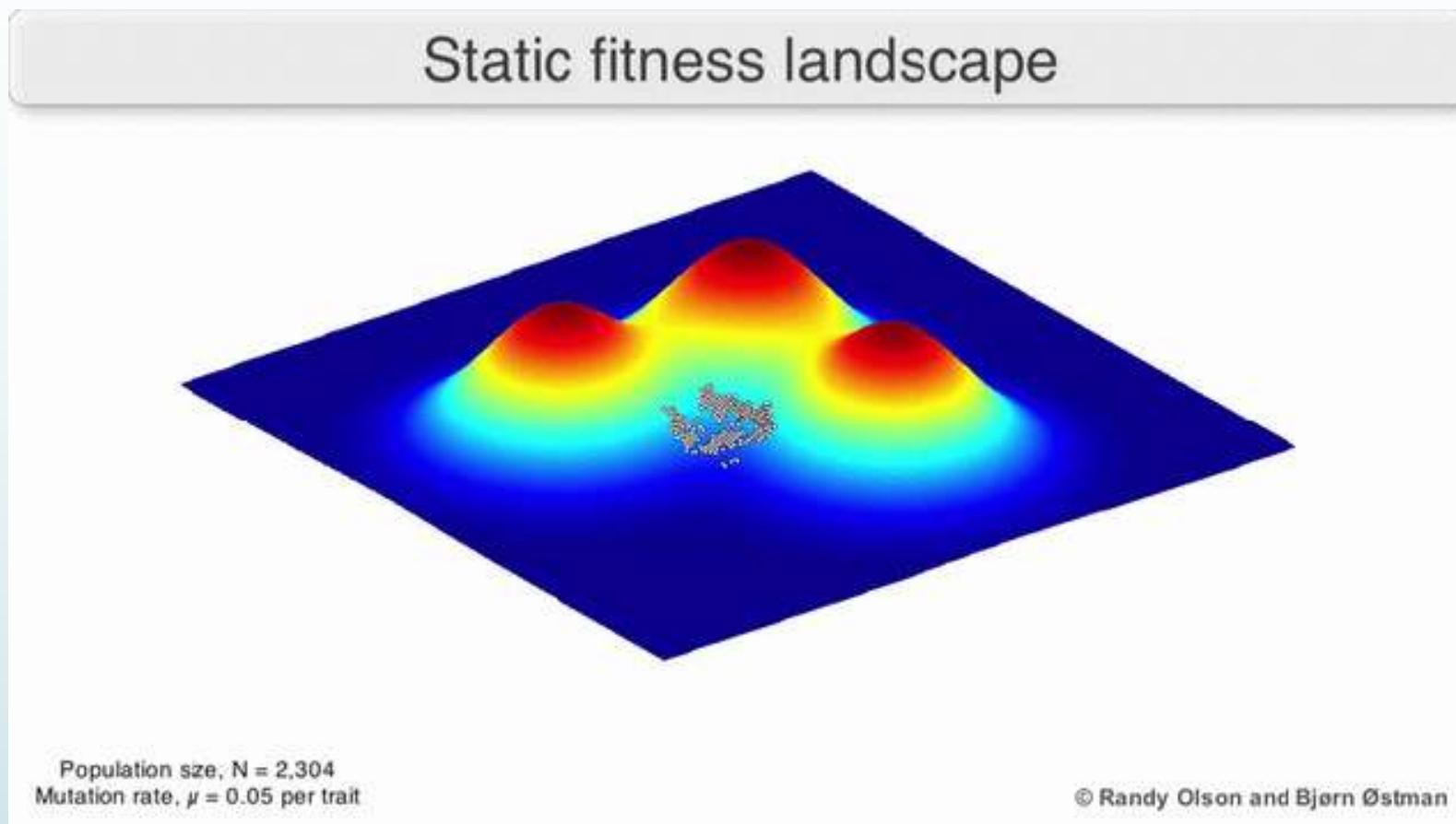
Quantify the Problem

- ▶ Is our problem smooth or rough?
 - ▶ How smooth or rough?
 - ▶ Can we quantify this?
- ▶ Do we have gaps?
 - ▶ Areas where our parameters
- ▶ Can we visualize this?
 - ▶ Visuals are helpful to our understanding
 - ▶ Demonstrate problems to others

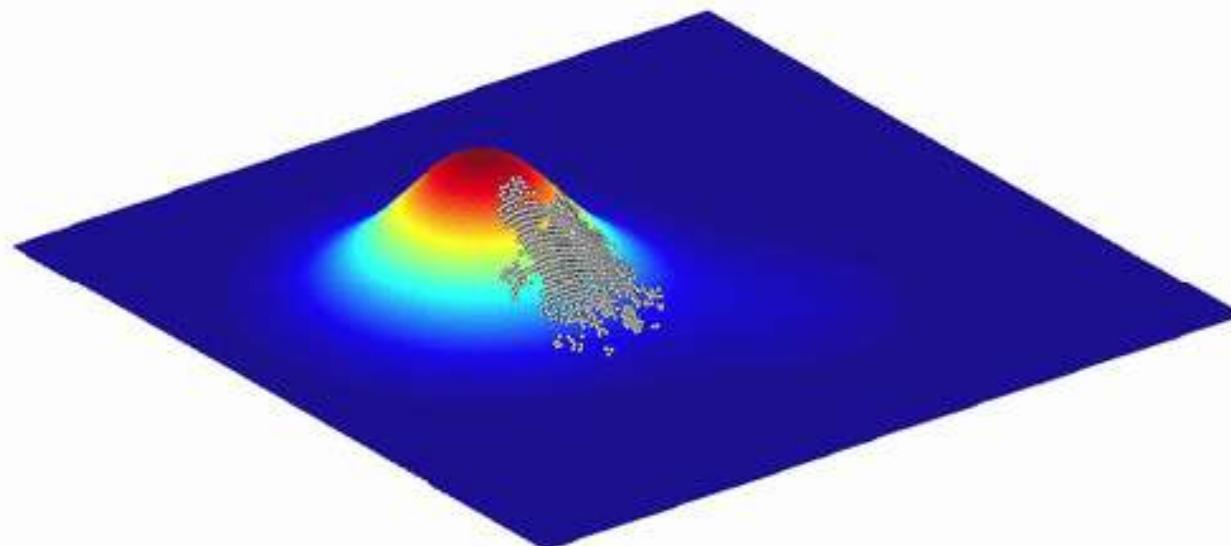
Fitness Landscape

- ▶ A fitness landscape is a Tuple $L = (S, f, N)$ where S is the search space of feasible solutions, $f: S \rightarrow R$ is a fitness function, and $N(s)$ is the set of neighbouring points for a member of S such that they can be reached by a single application of the search operation
- ▶ This defines a 3 dimensional space which can be projected as a mountain

Visualization by R.Olson and B.Ostman



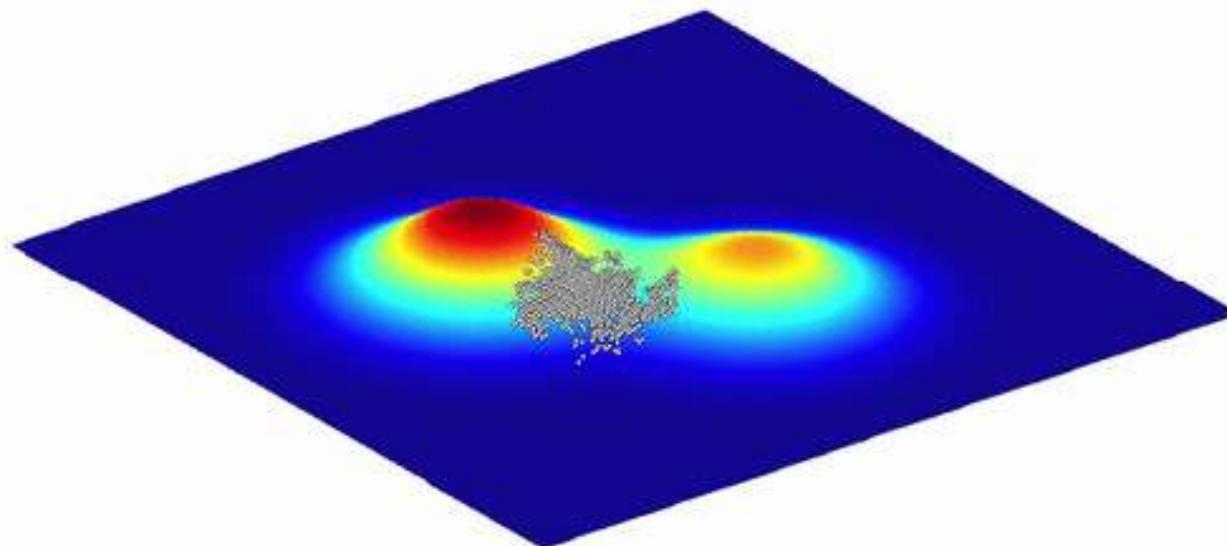
Dynamic fitness landscape



Population size, $N = 2,304$
Mutation rate, $\mu = 0.5$ per trait

© Randy Olson and Bjørn Østman

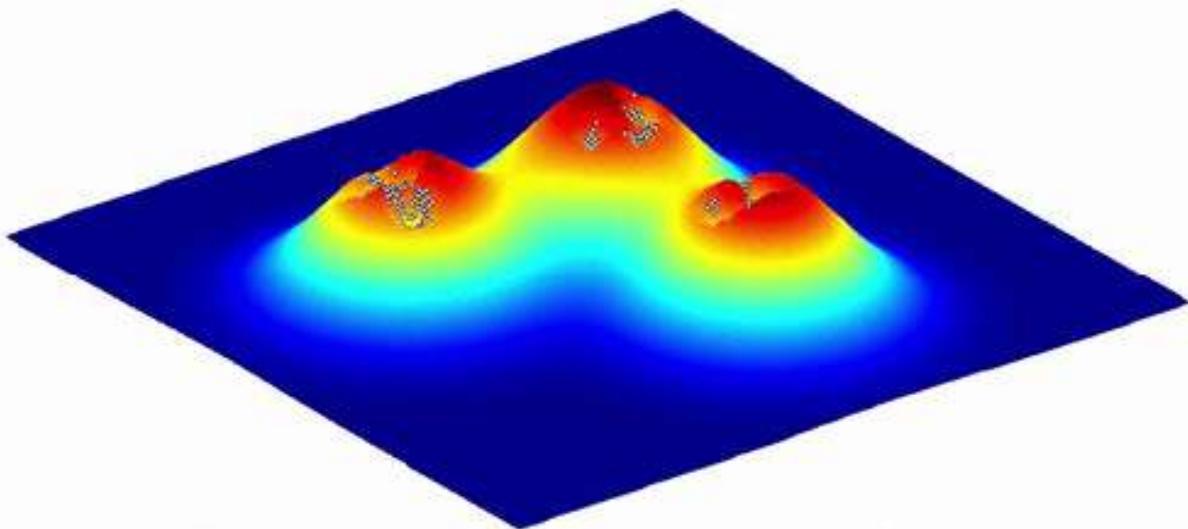
Dynamic fitness landscape changes 3x faster



Population size, $N = 2,304$
Mutation rate, $\mu = 0.5$ per trait

© Randy Olson and Bjørn Østman

Density-dependent fitness landscape



The population can split into subpopulations (species)
when fitness depends on the density of the organisms

Population size, $N = 2,304$
Mutation rate, $\mu = 0.05$ per trait

© Randy Olson and Bjørn Østman

Local Maxima

- 
- ▶ A solution s in S is said to be a local maximum if its fitness is greater than or equal to all of neighbours, i.e. $f(s) \geq f(w) \forall w \in N(s)$
 - ▶ If there is a relatively high number of such maximums, then the space is called rugged
 - ▶ If there is a low number of maximums, then the space is declared to be smooth
 - ▶ Note that this highly depends on how the neighbourhood is defined

Fitness Neutrality

- ▶ If the fitness is relatively neutral, then the search will randomly walk about this area
- ▶ If this is the case, then our movements will for the most part will not progress to a local optimum
- ▶ If the space has many local optima, then it may be likely to be stuck, by falling into a pit we cannot move out of based on our neighbor graph

Quantifying the Space

- ▶ Take a random walk about the space – we are not interested in the optimization but knowing how the space looks to the operators
- ▶ Take the connections mixed with the fitness values, who is a neighbour of who
- ▶ Use this measure to get an idea of how the space looks – surveying the space

Information Content

- ▶ Uses principles from Information theory to give a measure of the ruggedness of the space
- ▶ Transform the space or area in a space into a single quantifiable value based on the Entropy
- ▶ Idea is to make random walks over the space, score them based on the gradient, then check the amount of variation in these changes

Three Modalities

- ▶ We move from higher to lower fitness beyond a tolerance
 - ▶ Score as a -1
- ▶ We keep the same fitness within a tolerance
 - ▶ Score as a 0
- ▶ We move from a lower to higher fitness beyond a tolerance
 - ▶ Score as a 1
- ▶ A smooth space will have a very little change in these values
(0,0,1,1,1,1,1,1,0)
- ▶ A rugged space will have a 'randomness' in the values (0,1,-1,-1,0,1,1,0)

Questions?



Introduction to Artificial Intelligence

Week 6



Learning by Searching

Part II

Exploring and Exploiting

- ▶ Exploration – the want for a search method to find new areas of the search space which have not been visited yet
- ▶ Exploitation – the want for a search method to locally search about the best known areas in the space in order to refine these current solutions
- ▶ All search algorithms will have some mix of both properties
- ▶ Note that if we only have a limited number of objective tests available, then we have a zero-sum game – we can explore or we can search with a test

Exploration vs Exploitation

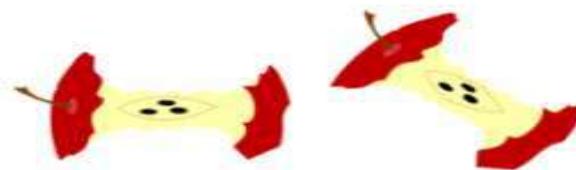
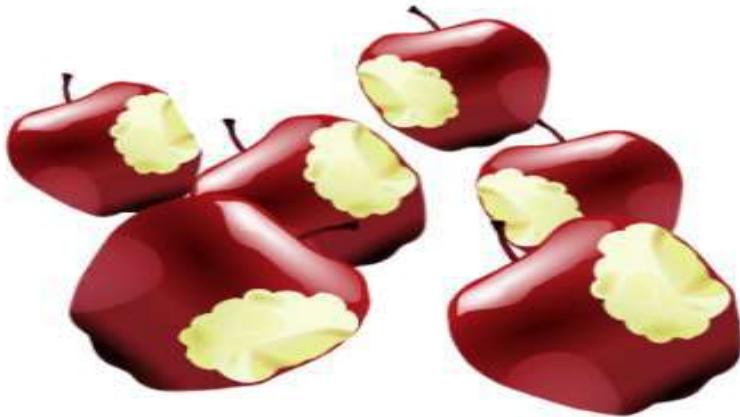
- Before visiting a new city, we searched for the restaurants where we can buy sushi.

Exploration

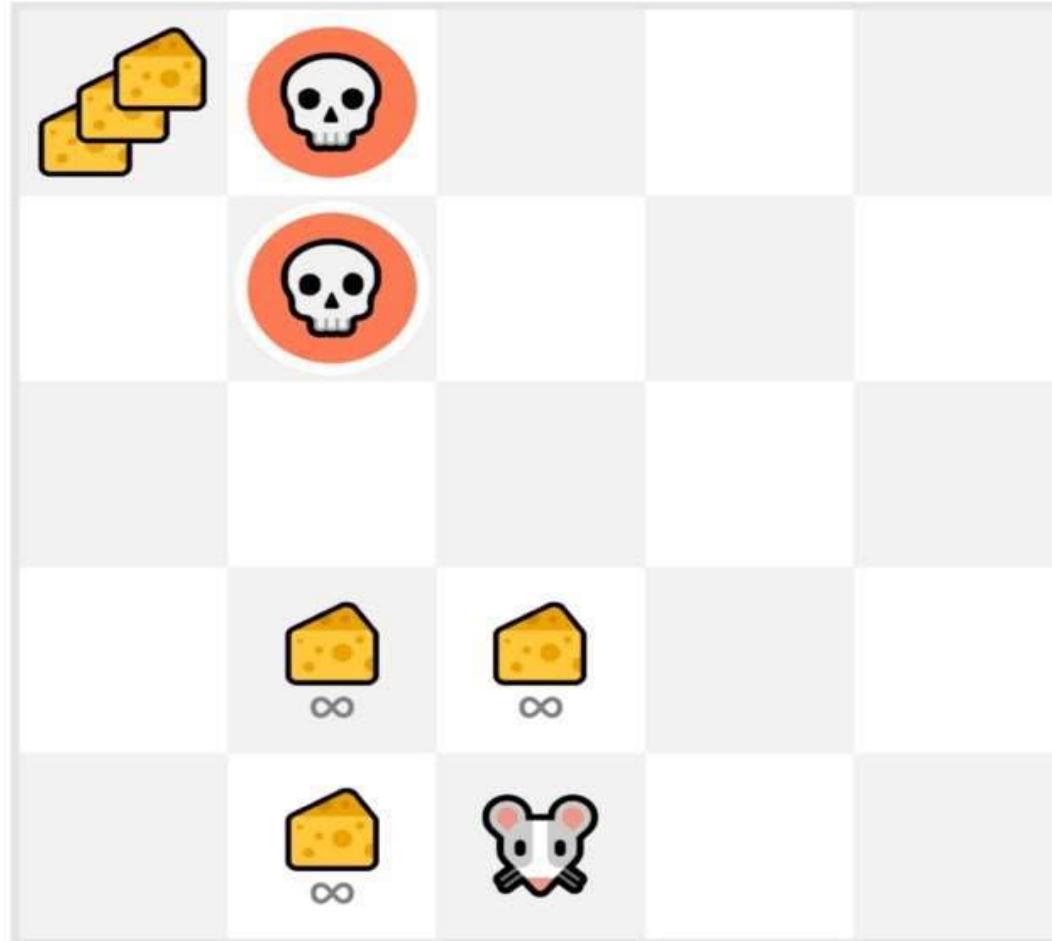
- ▶ Before visiting a new city, we searched for the restaurants where we can buy sushi.
- ▶ It's a new city, we have **no information** about it before.

Exploration vs Exploitation

Do you
Explore or Exploit?



Exploration vs Exploitation



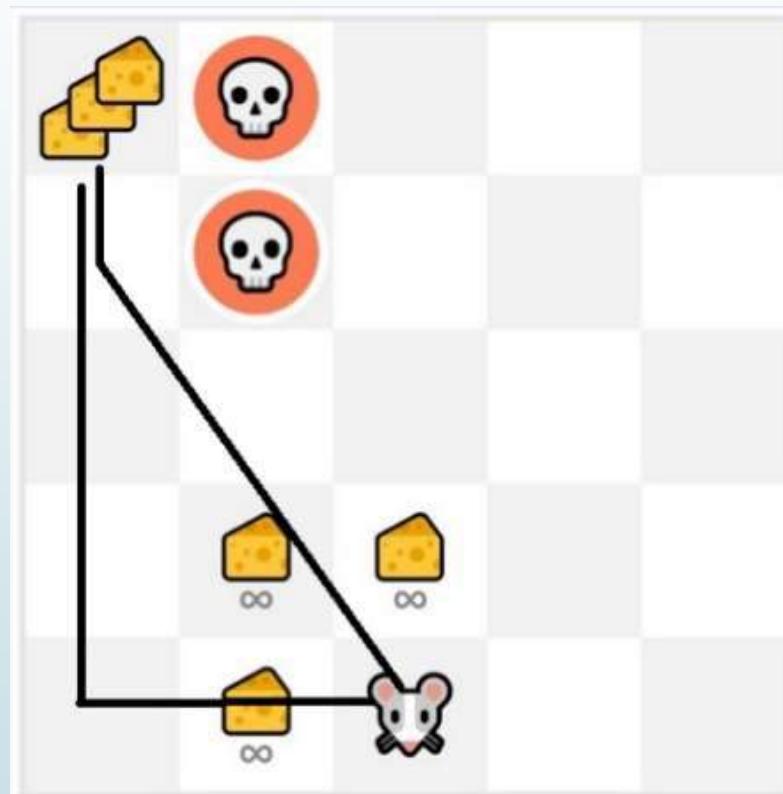
Exploration vs Exploitation

- ▶ In this game, our mouse can have an infinite amount of small cheese (+1 piece per day). But at the top of the maze, there is a gigantic sum of cheese (+10000).
- ▶ Now, let's delve into the motivation behind the possible mouse's actions: What motivates the mouse to explore the maze further, and under what conditions would it choose to do so? How does the desire for a better solution or a change in satisfaction level influence its decision to explore or exploit the available cheese sources?

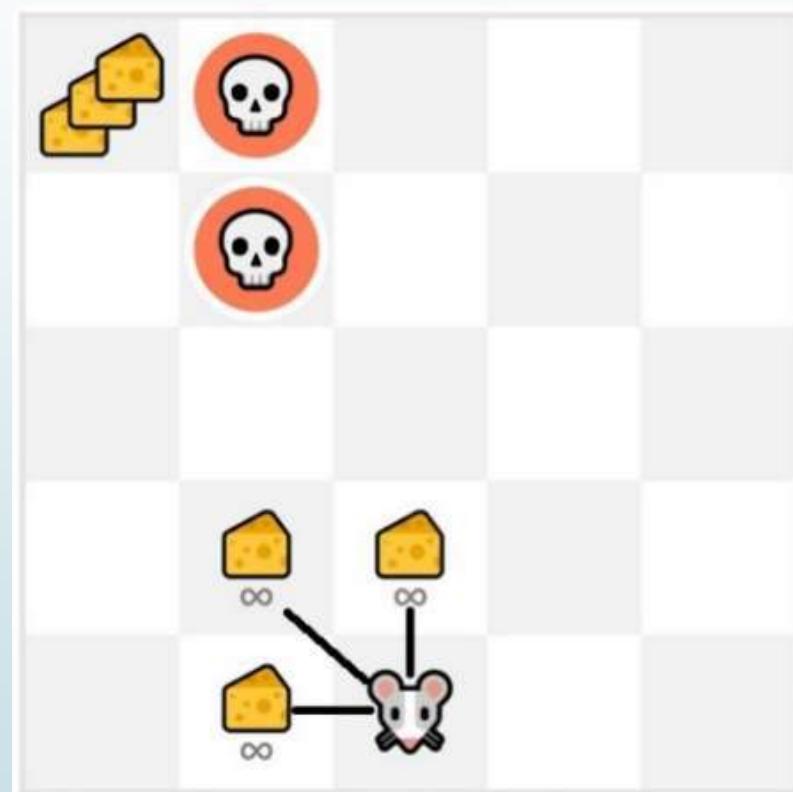
Exploration vs Exploitation

- ▶ If we only focus on exploitation, our agent will never reach the gigantic sum of cheese. Instead, it will only exploit the nearest source of rewards, even if this source is small (exploitation).
- ▶ But if our agent does a little bit of exploration, it can discover the big reward (the pile of big cheese).
- ▶ This is what we call the exploration/exploitation trade-off. We need to balance how much we explore the environment and how much we exploit what we know about the environment.

Exploration



Exploitation



Help the Prince to find Cinderella

- ▶ The prince will be at some position and the surrounding boxes will have some costs:
 - ▶ Diagonals cost = **14**
 - ▶ Non-Diagonals cost = **10**

14	10	14
10	Prince	10
14	10	14

Help the Prince to find Cinderella

- There will be three costs in each square (**Total**, G, H):
 - Cost of square from the Prince = G
 - Cost of square from Cinderella = H (straight line distance)
 - Total cost (F) = G + H
 - Our goal is to minimize the total cost

				Cinderella	
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)			
(62, 10, 52)	Prince	(48, 10, 38)			
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- ▶ Prince will move to the square with least costs.
- ▶ After moving to next square, new costs will be calculated, and Prince will move to the one with least cost.

				Cinderella	
	(62, 28, 34)	(48, 24, 24)	(42, 28, 14)		
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)		
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)		
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- ▶ Prince will move to the square with least costs.
- ▶ After moving to next square, new costs will be calculated, and Prince will move to the one with least cost.

				Cinderella	
	(62, 28, 34)	(48, 24, 24)	(42, 28, 14)		
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)		
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)		
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- ▶ Prince will move to the square with least costs.
- ▶ After moving to next square, new costs will be calculated, and Prince will move to the one with least cost.

		(62, 42, 20)	(48, 38, 10)	(42, 42, 0)	
	(62, 28, 34)	(48, 24, 24)	(42, 28, 14)	(48, 38, 10)	
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)	(62, 42, 20)	
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)		
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- If there is some obstacle, the prince will change it's direction based on the scores which are explored:
 - i.e. it will move to the square with least costs.
 - If there are several with same least cost, then it will go to the one with least cost from Cinderella.

				Cinderella	
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)			
(62, 10, 52)	Prince	(48, 10, 38)			
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- We will move to the square with least cost, i.e. **(42, 14, 28)**

				Cinderella	
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)		
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)		
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- We will move to the square with least total cost and in this case we have three of such squares, so we will move to the square with least total cost and least cost of heuristics (cost from Cinderella). i.e. **(48,24,24)**, and calculate new scores

				Cinderella	
					(48, 38, 10)
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)	(54, 34, 20)	
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)	(68, 38, 30)	
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- Now we can see that the lowest total cost is 48 for three squares, so we'll choose based on heuristics, which is **(48, 38, 10)**.

			(62, 52, 10)	Cinderella	(62, 52, 10)
				(48, 38, 10)	(62, 48, 14)
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)	(54, 34, 20)	(76, 52, 24)
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)	(68, 38, 30)	
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Help the Prince to find Cinderella

- This is our shortest path by following A* Algorithm with total cost of **48**.

			(62, 52, 10)	(48, 48, 0)	(62, 52, 10)
				(48, 38, 10)	(62, 48, 14)
(62, 14, 48)	(48, 10, 38)	(42, 14, 28)	(48, 24, 24)	(54, 34, 20)	(76, 52, 24)
(62, 10, 52)	Prince	(48, 10, 38)	(62, 28, 34)	(68, 38, 30)	
(70, 14, 56)	(62, 10, 52)	(62, 14, 48)			

Task 1: Help Prince

- Divide into teams of 4-5 people and find out the shortest path for Prince to meet Cinderella. Follow the A* algorithm and write down the costs at each visited square. Stepmom, Stepsisters are obstacles for the Prince.

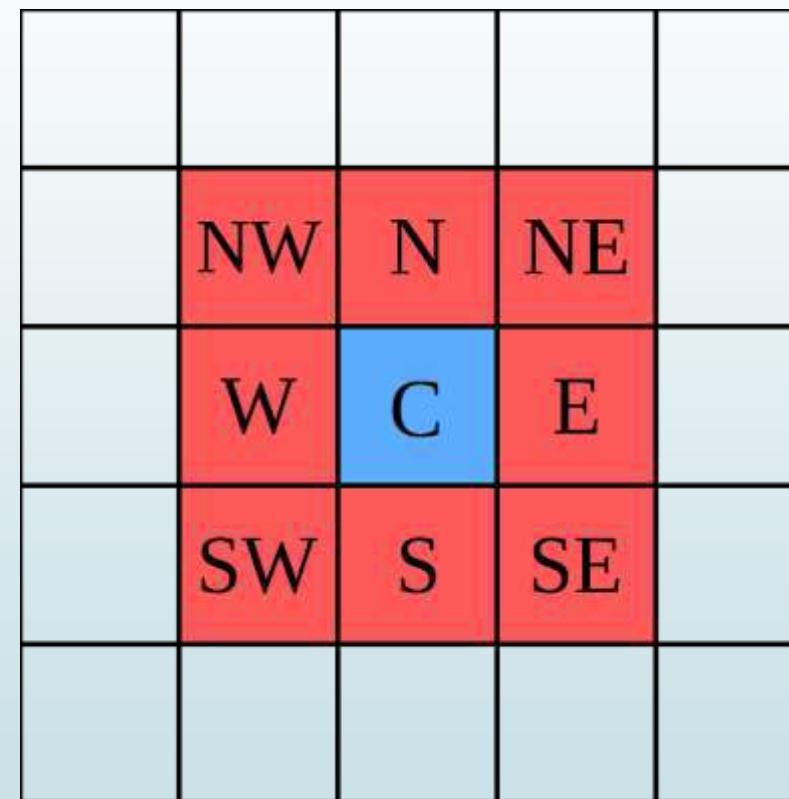
9									
8						Cinderella			
7									
6					Stepsis1	Stepsis2	Stepsis3		
5									
4	Stepmom								
3									
2		Prince							
1									
	1	2	3	4	5	6	7	8	

Task 1: Solution

► The least cost is **80**

9									
8			(100, 80, 20)	(86, 76, 10)	(80, 80, 0)				
7		(100, 66, 34)	(86, 62, 24)	(80, 66, 14)	(86, 76, 10)				
6		(100, 62, 38)	(80, 52, 28)	Stepsis1	Stepsis2	Stepsis3			
5		(108, 66, 42)	(86, 48, 38)	(72, 38, 34)	(72, 42, 30)	(86, 52, 34)			
4	Stepmom			(72, 28, 44)	(78, 38, 40)	(100, 56, 44)			
3	(80, 14, 66)	(72, 10, 62)	(72, 14, 58)	(78, 24, 54)	(92, 42, 50)				
2	(86, 10, 76)	Prince	(78, 10, 68)	(92, 28, 64)					
1	(100, 14, 86)	(92, 10, 82)	(92, 14, 78)						
	1	2	3	4	5	6	7	8	

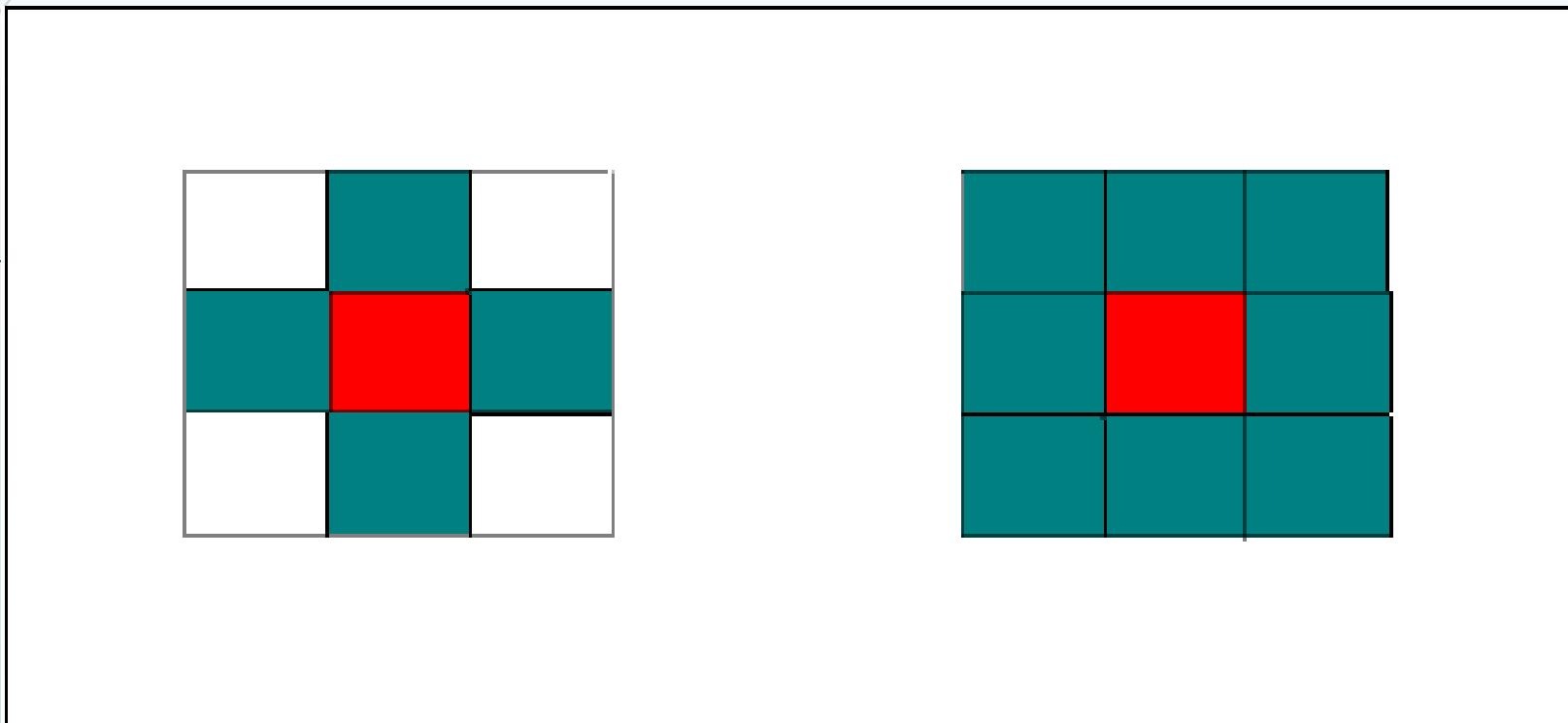
Moore Neighborhood Algorithm Range 1



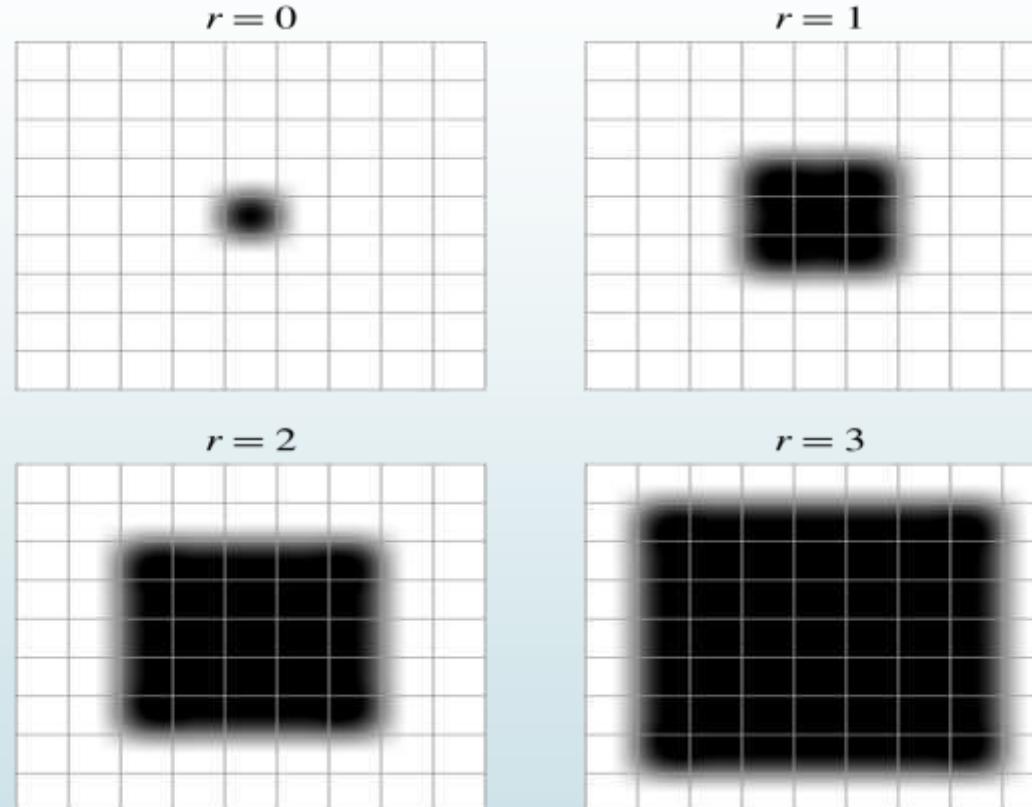
Moore Neighborhood Algorithm

- ▶ Using Moore Neighborhood algorithm, the agent(the prince in our example) can check all 8 cells of his immediate neighborhood (exploring the environment around him).
- ▶ If any of there cells contain an obstacle such as the stepmom or stepsisters the prince will be aware of them.
- ▶ Based on this perception, the prince can decide how to move in order to avoid coming into a cell containing an obstacle.

Neuman vs Moore Range 1

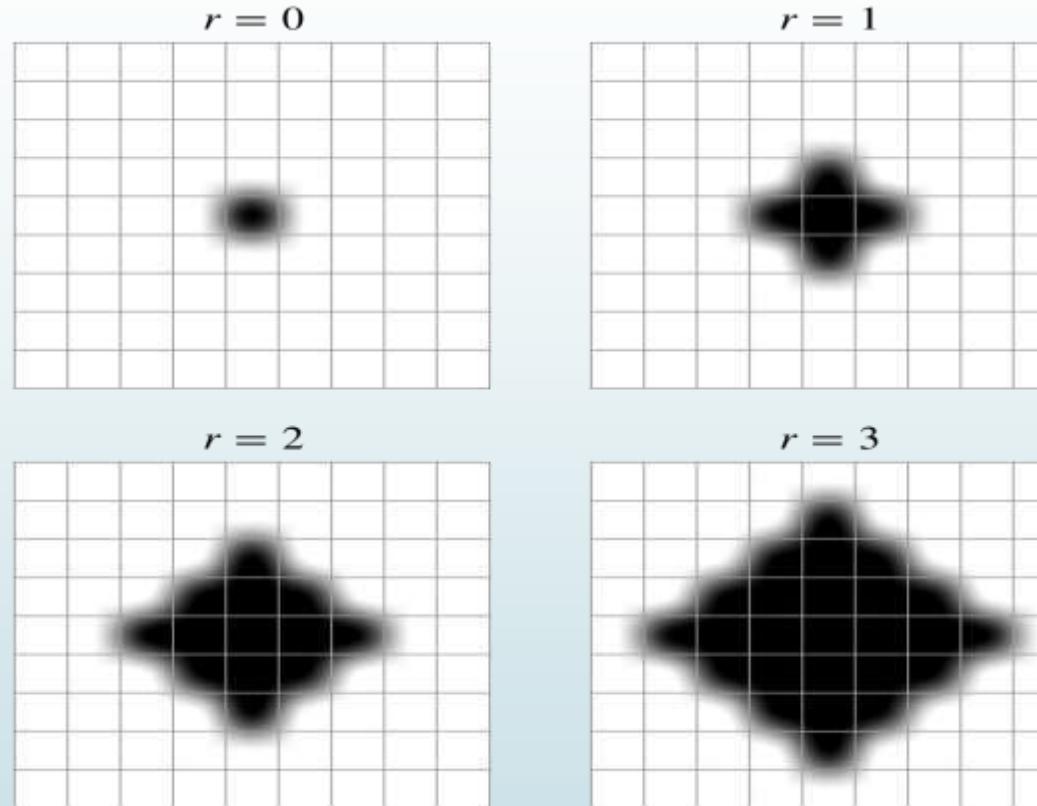


Moore Neighborhood Ranges



Moore neighborhoods for ranges $r = 0, 1, 2$, and 3 are illustrated above. The number of cells in the Moore neighborhood of range r is $(2r+1)^2$, the square of odd number.

Von Neumann Neighborhood Ranges



Von Neumann neighborhoods for ranges $r = 0, 1, 2$, and 3 are illustrated above. The number of cells in the von Neumann neighborhood of range r is $2r(r+1)+1$

Task 2: Help Prince

- Repeat the same game with obstacles. Prince is able to perceive obstacles via Moore neighborhood. The **obstacles** are hidden, ask the instructor for obstacle after making a move.

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

Task 2: Solution

► The least cost is 72

9									
8				(86, 76, 10)	(72, 72, 0)		(86, 76, 10)		
7			(86, 62, 24)	(72, 58, 14)	(72, 62, 10)		(86, 72, 14)		
6			Stepsis1	(72, 48, 24)	Stepsis2	Stepsis3			
5			(80, 42, 38)	(72, 38, 34)	(72, 42, 30)		(86, 52, 34)		
4	Stepmom			(72, 28, 44)	(78, 38, 40)	(100, 56, 44)			
3	(80, 14, 66)	(72, 10, 62)	(72, 14, 58)	(78, 24, 54)	(92, 42, 50)				
2	(86, 10, 76)	Prince	(78, 10, 68)	(92, 28, 64)					
1	(100, 14, 86)	(92, 10, 82)	(92, 14, 78)						
	1	2	3	4	5	6	7	8	

Extra Exercise

- ▶ Implement A* algorithm for considered world with obstacles using C/C++/C#/Java/Python.



Introduction to Artificial Intelligence

Week 8

A decorative graphic in the top-left corner features several thin, light-colored lines that intersect and curve across the frame. A larger, dark grey arrow points from the bottom-left towards the center. Inside this arrow is a white number '2'.

2

Statistics

What is Statistics?

- ▶ Statistics is the study of collecting, organizing, analyzing, and interpreting data in order to make decisions (Larson and Farber, 2006)
- ▶ Statistics is a measurement system for understanding randomness of variables and methods to compare randomness of systems

Applications

- ▶ Hard Sciences require statistics to understand the role of random events in the experimental process. There are often factors which are not controlled for or cannot be definitively measured, and experiments control for this variation. Quite often these experiments use hypothesis testing, where two groups, one with treatment and one without (control group) are compared against each other for an observable difference
- ▶ Statistics measures and quantifies the difference, and gives us an expectation on the likelihood that the results were due to the treatment and not due to chance

Applications

- ▶ Engineers work to control variance in processes, and will work to identify and remove sources of variance to ensure a process puts out a quality product, without defect. Root cause analysis techniques and quality control, especially SixSigma, use statistics to guide process improvements
- ▶ The study of statistics is not only useful for the hard sciences and engineering. Study of the soft sciences (economics, politics) often use polling data to understand the beliefs and preferences of consumers and voter

Descriptive v. Inferential Statistics

- Descriptive - organization, summarization, and display of data
- The mean temperature of Kazan over the year (Wikipedia)

Period	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
Daily mean °C (°F)	-10.4 (13.3)	-10.2 (13.6)	-4.0 (24.8)	5.5 (41.9)	13.3 (55.9)	18.1 (64.6)	20.2 (68.4)	17.6 (63.7)	11.7 (53.1)	4.8 (40.6)	-3.4 (25.9)	-8.5 (16.7)	4.6 (40.3)

- Inferential - using a sample to make conclusions about a population
- The mean temperature in Kazan has been increasing over the last 30 years and it is likely we will see an increase every year from now on

Population v. Sample

- ▶ Often in statistical tests we refer to the population and sample statistic
- ▶ The sample is any subset of the results seen in the full population, it is an approximation of the population
- ▶ The population is the measurements on the entire group that is of interest
- ▶ Imagine we are building car bumpers and wish to have an expectation on the number of defects which will occur. If one day we test one bumper of every 10 made (with 1000 produced), our sample is these 100 bumpers
- ▶ Is our population 1000 bumpers made that day?
- ▶ Our population is ALL the bumpers that we have and will be made on this production line

Data Collection Methods

- ▶ Observational Study – data is gathered by observing a part of the population in their normal actions
- ▶ Experimental Study – treatment is applied to a group (experimental group) and a second group is not given any treatment and observed (control). These groups are of a similar disposition in order to allow for a comparison between groups
- ▶ Simulation – use of computer, mathematical, or physical models to represent a situation. Usually done when data collection is dangerous or costly (e.g. crash testing cars)
- ▶ Survey – an investigation into the characteristics of a population by asking questions. This may be done by census, asking everyone in the population, but more commonly done by taking samples, a count or measure of part of the population

Measures of Variation

- ▶ Mean – the average
 - ▶ Sample Mean – the average of a set of numbers x of cardinality N : $\bar{x} = \frac{\sum_i^N x_i}{N}$
 - ▶ Population mean μ
- ▶ Mode – the most frequent element
- ▶ Median – the middle of the ordered set
- ▶ Unbiased Sample Variance of a set of numbers x of cardinality N
- ▶ $s^2 = \frac{\sum_i^N (\bar{x}-x_i)^2}{N-1}$
- ▶ Corrected Sample Standard Deviation $s = \sqrt{s^2} = \sqrt{\frac{\sum_i^N (\bar{x}-x_i)^2}{N-1}}$

Hypothesis Testing

- ▶ Two actions, the null hypothesis is that things are equal or that there is no discernible difference and the alternative is that there is a difference
- ▶ $H_o: a = b; a \leq b; a \geq b.$
- ▶ $H_a: a \neq b; a > b; a < b.$
- ▶ We can either reject or fail to reject the null hypothesis
 - ▶ We can accept the alternative
 - ▶ WE NEVER ACCEPT THE NULL! YOU CAN NEVER PROVE THE NULL HYPOTHESIS! YOU CANNOT PROVE THAT TWO DISTRIBUTIONS ARE THE SAME WITH A SAMPLE BASED TEST
 - ▶ If there is one difference, they are not the same – meaning you would have to look at the entire population

Example Hypothesis

- Two evolutionary algorithms (X, Y) were run on the same problem 30 times. We wish to test if there is a difference in the mean fitness value of the final generation.

Formulate the Null Hypothesis

- $H_o: \mu_x = \mu_y$
- There is no significant difference in the mean fitness value between the two algorithms

Formulate the Alternative Hypothesis

- $H_a: \mu_x \neq \mu_y$
- There is a significant difference in the mean fitness value between algorithm X and Y

Confidence

- ▶ A measure of the acceptance of Type I errors:
 - ▶ Rejecting the Null hypothesis when in reality we should not have rejected
 - ▶ Saying two things are different when they are in fact the same
 - ▶ Accusing an innocent man!
 - ▶ Saying someone has HIV when in reality they do not!
 - ▶ False Positive
- ▶ Confidence is measured by an $\alpha - value$ with 95% being the most common value in scientific literature. Most statistical tests aim to control confidence by a variable in the test

Power

- ▶ A measure of the acceptance of Type II errors:
 - ▶ Failing to reject the Null hypothesis when in reality we should have rejected
 - ▶ Saying two things are the same when in fact they are different
 - ▶ Acquitting an guilty man!
 - ▶ Saying someone does not have HIV when in reality they do!
 - ▶ False Negative
- ▶ Power is measured by the $\beta - value$ with most scientific measures ignoring it. The majority of statistical tests do not control for this, though increasing the sample size increases power

Student's T-test



- ▶ Historically, was developed by a chemist, W. S. Gosset, for the Guinness Brewery, Dublin, Ireland which would not allow it to be published. The Guinness plant hired some of what were then the best graduates of Oxford and Cambridge. The Company was well known also for their policy of "study-leaves" for technical staff
- ▶ Was published under a pen name "Student"
- ▶ Can easily see why in quality control we want to use a small number of samples and not a z-test. We would have some very 'happy' quality control officers in the middle of our brewery

Test Against a Known Mean

- ▶ t-test for a target mean (μ):
- ▶ t value =
$$\frac{\bar{x} - \mu}{s / \sqrt{N}}$$
- ▶ Check against t-Table, with degrees of freedom = N-1

Test Between Two Sample Means

- ▶ t-test between two sample means:
- ▶ $t \text{ value} = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_{\bar{x}_1 - \bar{x}_2}}$, where $s_{\bar{x}_1 - \bar{x}_2} = \sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}$
- ▶ Check against t-Table with degrees of freedom as the smaller of $N_1 - 1$ and $N_2 - 1$

P-value

- ▶ Often we will want to figure the probability of making a Type I error exactly! The test is only ensuring it is meeting with set confidence. For this we compute a P-value of the test
- ▶ We say that tests which have a:
 - ▶ P-value < 0.05 are “statistically significant” -> $\alpha > .95$ or 95% confidence
 - ▶ P-value < 0.01 are “extremely (very) statistically significant” -> $\alpha > .99$ or 99% confidence
- ▶ Note again: Statistical significance only measures that their difference is not caused by chance. It might be a very tiny effect, e.g. statistical significance vs. clinical significance of being exposed to a weak carcinogen

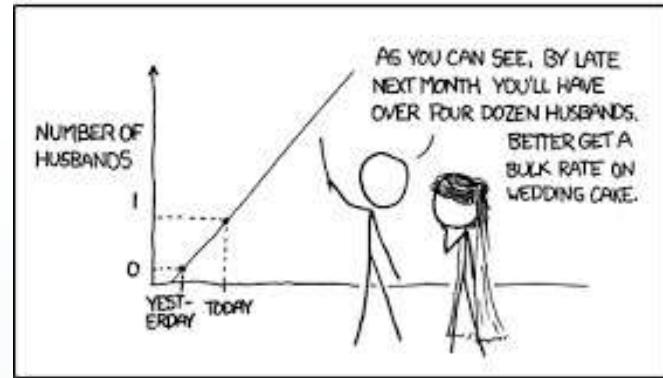
Confidence Interval

- ▶ Visualization of where the mean will be, upper gate and lower gate
- ▶ $\bar{x} - t_c \frac{s}{\sqrt{N}} < \mu < \bar{x} + t_c \frac{s}{\sqrt{N}}$
- ▶ 95% CI on a mean with $n>30$
- ▶ $\bar{x} - 1.96 \frac{s}{\sqrt{N}} < \mu < \bar{x} + 1.96 \frac{s}{\sqrt{N}}$
- ▶ If Confidence Intervals do not overlap we can be sure that a t-test would find it is significant at the X-confidence level
- ▶ If they do overlap we do not necessarily know if we cannot reject the null hypothesis – for that we need to examine the rejection gates which are smaller than the CI

Look-Elsewhere Effect

- ▶ P-value of .05 means that 1/20 times we have demonstrated a difference
- ▶ If we do multiple tests with the same data against a group of other, non-independent trials such as checking multiple, we have a higher likelihood of seeing a Type I error (demonstrating a difference when no difference exists) across the group. This is the family error rate of the group of hypotheses, which we would like to reduce
- ▶ When comparing groups of hypothesis, this issue is controlled by increasing the sample sizes and applying a corrective measure – such as Bonferroni correction. Note this reduces the POWER of the test, which is increased by raising the number of samples

Lies, Damn Lies, Statistics



- ▶ Learning statistics protects you from abuses of data. Politicians, activists, advertisers, sales people, and anyone wanting to convince you to do something will use data in a way that benefits their arguments – they are not aiming to be objective to the data
- ▶ “One half of all students in our country are below the average math score, we need more funding for Education” – ‘Expert’ on a Canadian News Magazine show
- ▶ Learning statistics makes you more defensible to such tactics – it is a brain anti-virus for bad data
- ▶ “Our brand of cereal has 1/3 the fat of the next leading brand”
 - ▶ How much fat was there before? 1/3 of a small amount is not that much of an improvement
 - ▶ How much sugar did you add?
- ▶ “In the last year our consumer base has quadrupled – you can’t pass up such an investment”
 - ▶ So you went from having one customer to four?



57% of 9-19 year-olds have seen pornography online from

"My teen watches porn – Parentchannel.tv"

- ▶ What message is being given:
- ▶ "My teen watches porn" "57% of teens have watched porn online"
- ▶ Is 9-19 a good sample group? How many were sampled of each age?
 - ▶ 100% of their sampled 18-19 year olds could be porn watchers
 - ▶ 0% of 9-17 year olds have seen any
- ▶ Are parents going to be worried about a ADULT (18-19) seeing pornography?
- ▶ What is the definition of SEEN?
 - ▶ Actively watched, seeking for it
 - ▶ Have had an advertisement pop-up
- ▶ What is the definition of PORN?
 - ▶ Does a love seen in a Hollywood movie count?
 - ▶ Does classical paintings, or nudity in art count?
- ▶ How was the data collected?
 - ▶ Was there a reporting bias?

Have to ask how a study was conducted

- ▶ Did it collect sensitive details anonymously?
 - ▶ People will LIE. Some to reduce (don't want to get in to trouble), some to inflate (maybe some 18-19 year olds trying to look cool to peers)
- ▶ Have to collect answers in a way which does not lead to a conclusion, and that adequately accounts for the possible beliefs or actions. So we could correct the phasing of our poll by asking:
 - ▶ In the last month how many times have you intentionally viewed an image or video in which the plot revolved about sexual actions including but not limited to sexual intercourse (i.e. pornographic materials)
- ▶ Have to ensure a sample which is REPRESENTATIVE of your population, and preferably one which is randomly selected
 - ▶ In a poll of 9-17 year old, with 30 members in each age group, asked the previous question 57% of them responded having at least one instance or more. The following is a breakdown by age group...
 - ▶ Making a poll of Farmers on how best to utilize land does not tell you how someone in a city will wish it used

Statistical Methods

**Darko Bozhinoski,
Ph.D. in Computer Science**

What is a statistical test ?

It is a decision method that helps us to validate or invalidate a statistical hypothesis with a certain degree of confidence.

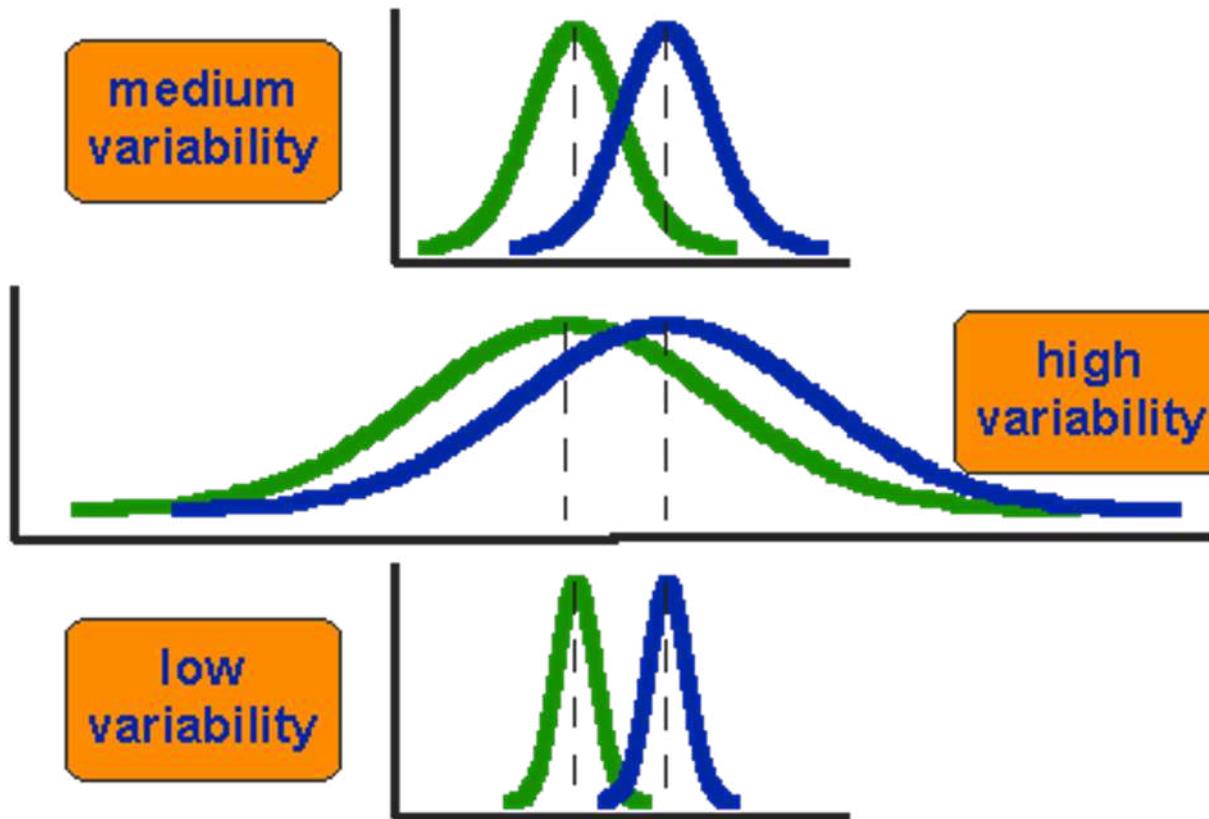
Statistical Tests

Statistical tests verify the truthfulness of statistical hypotheses => a process called statistical inference

- hypothesis H0 (or null hypothesis) data have no connections between them/ compared values do not differ
- hypothesis H1 (or alternate hypothesis) data have connections between them/ compared values do differ

T-Test

The t-test assesses whether the means of two groups are statistically different from each other. This analysis is appropriate whenever you want to compare the means of two groups.

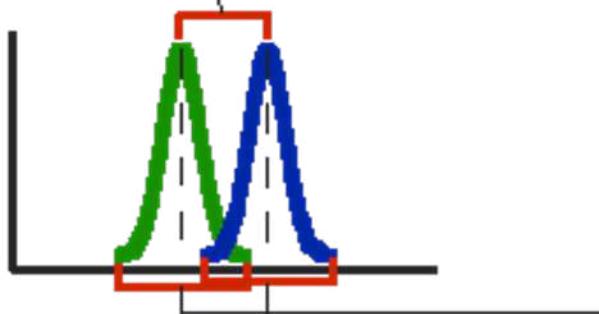


signal
noise = difference between group means
 variability of groups

=

$$\frac{\bar{X}_T - \bar{X}_C}{SE(\bar{X}_T - \bar{X}_C)}$$

= t-value



signal
noise

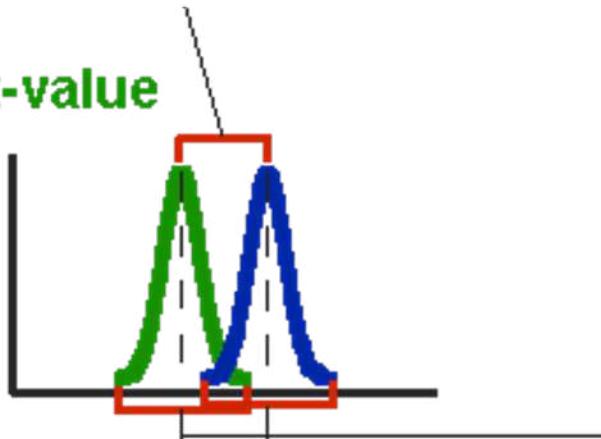
= diff

=

$$t = \frac{\bar{X}_T - \bar{X}_C}{\sqrt{\frac{\text{var}_T}{n_T} + \frac{\text{var}_C}{n_C}}}$$

s

= **t-value**



- Once you compute the t-value you have to look it up in a table of significance to test whether the ratio is large enough to say that the difference between the groups is not random. To test the significance, you need to set a risk level (called the alpha level). In most research, the “rule of thumb” is to set the alpha level at .05.
- You also need to determine the degrees of freedom (df) for the test. In the t-test, the degrees of freedom is the sum of individual data in both groups minus 2. Given the alpha level, the df, and the t-value, you can look the t-value up in a standard table of significance to determine whether the t-value is large enough to be significant. If it is, you can conclude that the difference between the means for the two groups is different.

Non-parametric test

- ❖ These are tests that do not make assumptions about data distribution.
- ❖ They can be applied in all circumstances, if the measurements are independent
- ❖ Usually they are used when we are not allowed to apply a parametric test
- ❖ These tests have less accurate results than parametric tests, so we apply them only when it is not possible to apply a parametric test

Wilcoxon Rank Sum Test

Wilcoxon Rank Sum Test is used to test whether two samples are likely to derive from the same population (i.e., that the two populations have the same shape). Some investigators interpret this test as comparing the medians between the two populations.

H_0 : The two populations are equal versus

H_1 : The two populations are not equal.

Example

Consider a Phase II clinical trial designed to investigate the effectiveness of a new drug to reduce symptoms of asthma in children. A total of n=10 participants are randomized to receive either the new drug or a placebo. Participants are asked to record the number of episodes of shortness of breath over a 1 week period following receipt of the assigned treatment. The data are shown below.

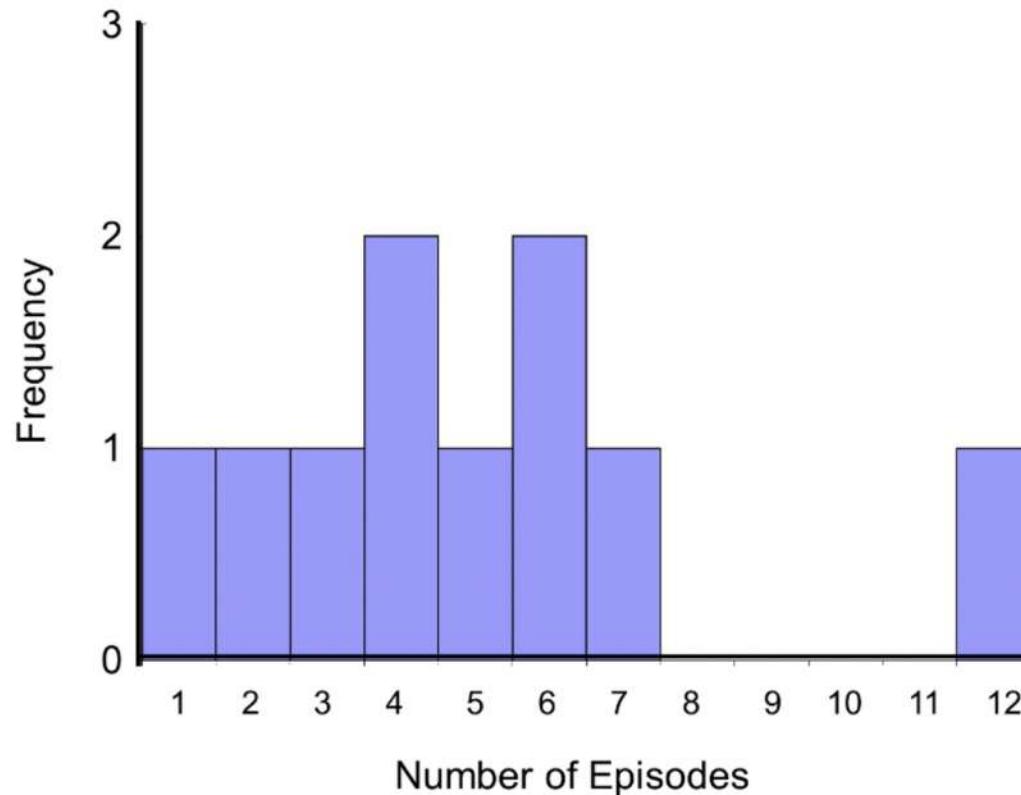
Placebo	7	5	6	4	12
New Drug	3	6	4	2	1

Example (2)

Is there a difference in the number of episodes of shortness of breath over a 1 week period in participants receiving the new drug as compared to those receiving the placebo? By inspection, it appears that participants receiving the placebo have more episodes of shortness of breath, but is this statistically significant?

Placebo	7	5	6	4	12
New Drug	3	6	4	2	1

Frequency Histogram of Number of Episodes of Shortness of Breath



Hypothesis

H_0 : The two populations are equal versus

H_1 : The two populations are not equal.

Note that if the null hypothesis is true (i.e., the two populations are equal), we expect to see similar numbers of episodes of shortness of breath in each of the two treatment groups, and we would expect to see some participants reporting few episodes and some reporting more episodes in each group. This does not appear to be the case with the observed data. A test of hypothesis is needed to determine whether the observed data is evidence of a statistically significant difference in populations.

Wilcoxon Rank Sum Test Procedure

Step 1. The first step is to assign ranks and to do so we order the data from smallest to largest. This is done on the combined or total sample (i.e., pooling the data from the two treatment groups ($n=10$)), and assigning ranks from 1 to 10, as follows. We also need to keep track of the group assignments in the total sample.

		Total Sample (Ordered Smallest to Largest)		Ranks	
Placebo	New Drug	Placebo	New Drug	Placebo	New Drug
7	3		1		1
5	6		2		2
6	4		3		3
4	2	4	4	4.5	4.5
12	1	5		6	
		6	6	7.5	7.5
		7		9	
		12		10	

Wilcoxon Rank Sum Test Procedure (2)

Note that the lower ranks (e.g., 1, 2 and 3) are assigned to responses in the new drug group while the higher ranks (e.g., 9, 10) are assigned to responses in the placebo group. The goal of the test is to determine whether the observed data support a difference in the populations of responses. In parametric tests when comparing means between two groups, we analyzed the difference in the sample means relative to their variability and summarized the sample information in a test statistic.

		Total Sample (Ordered Smallest to Largest)		Ranks	
Placebo	New Drug	Placebo	New Drug	Placebo	New Drug
7	3		1		1
5	6		2		2
6	4		3		3
4	2	4	4	4.5	4.5
12	1	5		6	
		6	6	7.5	7.5
		7		9	
		12		10	

Wilcoxon Rank Sum Test Procedure (3)

Step 2. Calculate Test Statistics U. Choose the lower value of U1 and U2.

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

where R_1 = sum of the ranks for group 1 and R_2 = sum of the ranks for group 2.

$$U_1 = 5(5) + \frac{5(6)}{2} - 37 = 3$$

$$U_2 = 5(5) + \frac{5(6)}{2} - 18 = 22$$

Smaller values of U support the research hypothesis, and larger values of U support the null hypothesis.

Wilcoxon Rank Sum Test Procedure (4)

Step 3. In every test, we must determine whether the observed U supports the null or research hypothesis. This is done following the same approach used in parametric testing. Specifically, we determine a critical value of U such that if the observed value of U is less than or equal to the critical value, we reject H_0 in favor of H_1 and if the observed value of U exceeds the critical value we do not reject H_0 .

Table of Critical Values for the Wilcoxon Rank-Sum Test

http://plantsys.elte.hu/oktatas/Biometria/tablazatok/Wilcoxon_table_ketmuntas_probahoz.pdf

Example

To determine the appropriate critical value we need sample sizes (for our Example: $n_1=n_2=5$) and our two-sided level of significance ($\alpha=0.05$).

- The critical value is 2, and the decision rule is to reject H_0 if $U \leq 2$. We do not reject H_0 because $3 > 2$. We do not have statistically significant evidence at $\alpha = 0.05$, to show that the two populations of numbers of episodes of shortness of breath are not equal. However, in this example, the failure to reach statistical significance may be due to low power.

Critical Values of the Mann-Whitney U
(Two-Tailed Testing)

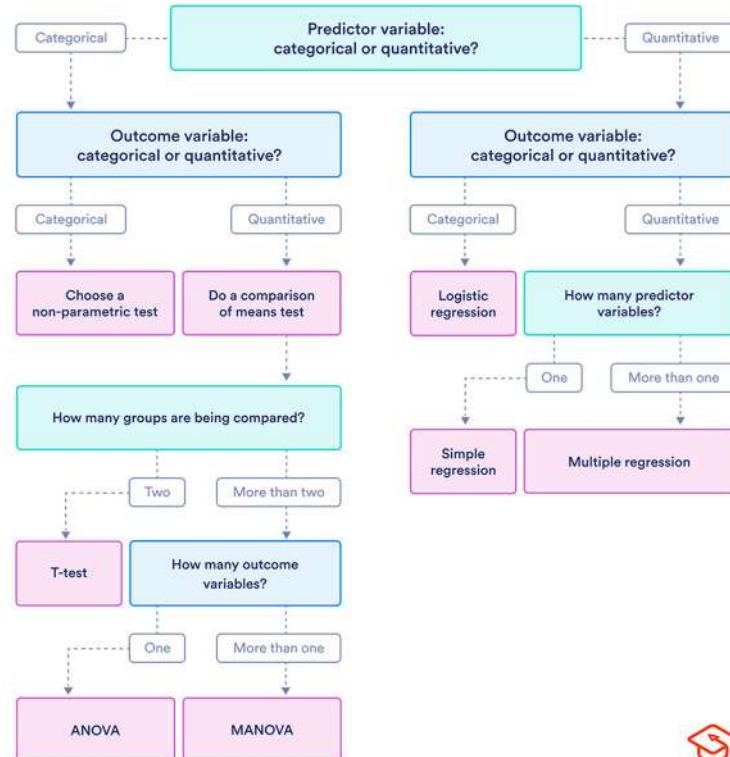
n_2	α	n ₁																		
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
3	.05	--	0	0	1	1	2	2	3	3	4	4	4	5	5	6	6	7	7	8
	.01	--	0	0	0	0	0	0	0	0	1	1	1	2	2	2	3	3	3	3
4	.05	--	0	1	2	3	4	4	5	6	7	8	9	10	11	11	12	13	14	
	.01	--	0	0	0	1	1	2	2	3	3	4	5	5	6	6	7	8		
5	.05	0	1	2	3	5	6	7	8	9	11	12	13	14	15	17	18	19	20	
	.01	--	0	1	1	2	3	4	5	6	7	8	9	10	11	12	13	13	14	
6	.05	1	2	3	5	6	8	10	11	13	14	16	17	19	21	22	24	25	27	
	.01	--	0	1	2	3	4	5	6	7	9	10	11	12	13	15	16	17	18	
7	.05	1	3	5	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	
	.01	--	0	1	3	4	6	7	9	10	12	13	15	16	18	19	21	22	24	
8	.05	2	4	6	8	10	13	15	17	19	22	24	26	29	31	34	36	38	41	
	.01	--	1	2	4	6	7	9	11	13	15	17	18	20	22	24	26	28	30	
9	.05	2	4	7	10	12	15	17	20	23	26	28	31	34	37	39	42	45	48	
	.01	0	1	3	5	7	9	11	13	16	18	20	22	24	27	29	31	33	36	
10	.05	3	5	8	11	14	17	20	23	26	29	33	36	39	42	45	48	52	55	
	.01	0	2	4	6	9	11	13	16	18	21	24	26	29	31	34	37	39	42	
11	.05	3	6	9	13	16	19	23	26	30	33	37	40	44	47	51	55	58	62	
	.01	0	2	5	7	10	13	16	18	21	24	27	30	33	36	39	42	45	48	
12	.05	4	7	11	14	18	22	26	29	33	37	41	45	49	53	57	61	65	69	
	.01	1	3	6	9	12	15	18	21	24	27	31	34	37	41	44	47	51	54	
13	.05	4	8	12	16	20	24	28	33	37	41	45	50	54	59	63	67	72	76	
	.01	1	3	7	10	13	17	20	24	27	31	34	38	42	45	49	53	56	60	
14	.05	5	9	13	17	22	26	31	36	40	45	50	55	59	64	67	74	78	83	
	.01	1	4	7	11	15	18	22	26	30	34	38	42	46	50	54	58	63	67	
15	.05	5	10	14	19	24	29	34	39	44	49	54	59	64	70	75	80	85	90	
	.01	2	5	8	12	16	20	24	29	33	37	42	46	51	55	60	64	69	73	
16	.05	6	11	15	21	26	31	37	42	47	53	59	64	70	75	81	86	92	98	
	.01	2	5	9	13	18	22	27	31	36	41	45	50	55	60	65	70	74	79	
17	.05	6	11	17	22	28	34	39	45	51	57	63	67	75	81	87	93	99	105	
	.01	2	6	10	15	19	24	29	34	39	44	49	54	60	65	70	75	81	86	
18	.05	7	12	18	24	30	36	42	48	55	61	67	74	80	86	93	99	106	112	
	.01	2	6	11	16	21	26	31	37	42	47	53	58	64	70	75	81	87	92	
19	.05	7	13	19	25	32	38	45	52	58	65	72	78	85	92	99	106	113	119	
	.01	3	7	12	17	22	28	33	39	45	51	56	63	69	74	81	87	93	99	
20	.05	8	14	20	27	34	41	48	55	62	69	76	83	90	98	105	112	119	127	
	.01	3	8	13	18	24	30	36	42	48	54	60	67	73	79	86	92	99	105	

Example 2.

A new approach to prenatal care is proposed for pregnant women living in a rural community. The new program involves in-home visits during the course of pregnancy in addition to the usual or regularly scheduled visits. A pilot randomized trial with 15 pregnant women is designed to evaluate whether women who participate in the program deliver healthier babies than women receiving usual care. The outcome is the **APGAR score** measured 5 minutes after birth. Recall that APGAR scores range from 0 to 10 with scores of 7 or higher considered normal (healthy), 4-6 low and 0-3 critically low. The data are shown below.

Usual Care	8	7	6	2	5	8	7	3
New Program	9	9	7	8	10	9	6	

Choosing a Statistical Test



Link:

<https://www.scribbr.com/statistics/statistical-tests/>

DEMO - SWARM ROBOTICS

Examples of using statistical tests in swarm robotics:

- 1.<https://iridia.ulb.ac.be/supp/IridiaSupp2020-009/index.html>
- 2.<https://iridia.ulb.ac.be/supp/IridiaSupp2020-007/index.html>

References

1. https://www.statstutor.ac.uk/resources/uploaded/tutorsquickguideto_statistics.pdf
2. <https://www.scribbr.com/statistics/statistical-tests/>
3. <https://www.youtube.com/watch?v=qvPWQ-e03tQ&t=2s>
4. https://www.youtube.com/watch?v=_S68TAmFw4U



The image shows three wooden blocks standing upright on a light-colored wooden surface. The blocks are made of light-colored wood with visible grain. They are arranged side-by-side to spell out the letters 'Q', '&', and 'A'. The letter 'Q' is on the left block, '&' is on the middle block, and 'A' is on the right block. All three letters are printed in a large, bold, black sans-serif font. The background is a soft-focus view of a green and yellow outdoor scene, possibly a garden or park.

Q & A



Introduction to Artificial Intelligence

Week 8



Statistics

Mean

- ▶ Mean – the average
 - ▶ Sample Mean – the average of a set of numbers x of cardinality N
 - ▶ $\bar{x} = \sum_i^N x_i / N$
 - ▶ Population mean μ

Variance

- ▶ **Population** is all members of a specified group. **Sample** is a part of a population used to describe the whole group
- ▶ In many practical situations, the true variance of a population is not known *a priori* and must be computed somehow. When dealing with extremely large populations, it is not possible to count every object in the population, so the computation must be performed on a sample of the population. Sample variance can also be applied to the estimation of the variance of a continuous distribution from a sample of that distribution
- ▶ **Sample Variance** of a set of numbers x of cardinality N is $\sigma^2 = \frac{\sum_i^N (x_i - \bar{x})^2}{N-1}$
- ▶ **Population Variance** can be counted using this formula $\sigma^2 = \frac{\sum_i^N (x_i - \bar{x})^2}{N}$
- ▶ Standard Deviation $\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum_i^N (x_i - \bar{x})^2}{N}}$

Median

- ▶ It is the middle of ordered set. For example, if we know the set of numbers {7, 4, 8, 5, 9}, then ascending ordered set is {4, 5, 7, 8, 9}, **median is 7**. It can be used for **discrete series**
- ▶ For **interval variation series** median can be counted using next formula

$$M_e = x_0 + h \frac{\frac{\sum f_i}{2} - S_{m-1}}{f_m},$$

- ▶ where
 - ▶ M_e - median
 - ▶ x_0 - the lower bound of the interval that contains the median
 - ▶ h - interval value
 - ▶ $\sum f_i$ - the sum of the frequencies or the number of terms in the series
 - ▶ f_m - median interval frequency
 - ▶ S_{m-1} - the sum of the accumulated frequency intervals preceding the median interval

Mode

- Mode is the most frequent series value. For **discrete series** mode is the value with the highest frequency. For example, for series {6, 5, 6, 7} **6 is mode**. It is possible to have several values in mode like in next example {6, 5, 6, 7, 5}, where **mode is 5 and 6**. It is called multimodality
- For **interval variation series** mode can be counted using next formula

$$M_0 = x_o + h \frac{f_m - f_{m-1}}{(f_m - f_{m-1}) + (f_m - f_{m+1})}$$

- where
 - M_0 - mode
 - x_o - the lower bound of modal interval
 - h - interval value
 - f_m - modal interval frequency
 - f_{m-1} - the frequency of the interval preceding the modal interval
 - f_{m+1} - the frequency of the interval following the modal interval

Exercise 1: Find Mean

- Jane is 5 years old, Jack and John are both 6, Leo is 4. Find mean for their ages

Exercise 1: Solution

- ▶ Jane is 5 years old, Jack and John are both 6, Leo is 4. Find mean for their ages
- ▶ **Answer:** $\bar{x} = \sum_i^N x_i / N = (5+6+6+4)/4=5.25$

Exercise 2: Find Variance and Standard Deviation

- ▶ Ruby's class has a total population of six students. The students have the following reading speeds: 12, 8, 10, 10, 8, 12. Find population variance and standard deviation

Exercise 2: Solution

- ▶ Ruby's class has a total population of six students. The students have the following reading speeds: 12, 8, 10, 10, 8, 12. Find population variance and standard deviation
- ▶ **Answer:** $\sigma^2 = \frac{\sum_i^N (x_i - \bar{x})^2}{N} = \frac{(12-10)^2 + (8-10)^2 + (10-10)^2 + (10-10)^2 + (8-10)^2 + (12-10)^2}{6} = 2.67$
- ▶ $\sigma = \sqrt{\sigma^2} = \sqrt{2.67} = 1.634$

Exercise 3: Find Mode and Median Age

Age groups	Amount of students	Sum of accumulated frequencies ΣS
Less than 20 years	346	346
20 — 25	872	1218
25 — 30	1054	2272
30 — 35	781	3053
35 — 40	212	3265
40 — 45	121	3386
45 years and more	76	3462
Total	3462	

Exercise 3: Solution (1)

Age groups	Amount of students	Sum of accumulated frequencies ΣS
Less than 20 years	346	346
20 — 25	872	1218
25 — 30	1054	2272
30 — 35	781	3053
35 — 40	212	3265
40 — 45	121	3386
45 years and more	76	3462
Total	3462	

Exercise 3: Solution (2)

- ▶ In this example modal interval is located in age group of 25-30 because of the largest frequency (1054)
- ▶ Therefore, $M_0 = x_0 + h \frac{f_m - f_{m-1}}{(f_m - f_{m-1}) + (f_m - f_{m+1})} = 25 + 5 \frac{1054 - 872}{(1054 - 872) + (1054 - 781)} = 27 \text{ years}$
- ▶ Median interval is also located in the age group 25-30 because the value, which divides series into 2 equal parts ($\sum f_i / 2 = 3462 / 2 = 1731$) is inside of this interval ($1218 < 1731 < 2272$)
- ▶ Therefore, $M_e = x_0 + h \frac{\frac{\sum f_i}{2} - S_{m-1}}{f_m} = 25 + 5 \frac{\frac{3462}{2} - 1218}{1054} = 27.4 \text{ years}$



Introduction to Artificial Intelligence

Week 9

Monte Carlo Tree Search

2

Monte Carlo Tree Search

- ▶ In computer science, **Monte Carlo tree search (MCTS)** is a heuristic search algorithm for some kinds of decision processes, most notably those employed in game play
- ▶ A leading example of Monte Carlo tree search is recent computer Go programs, but it also has been used in other board games, as well as real-time video games and non-deterministic games such as poker



Innopolis University - Introduction to AI 2023 Fall



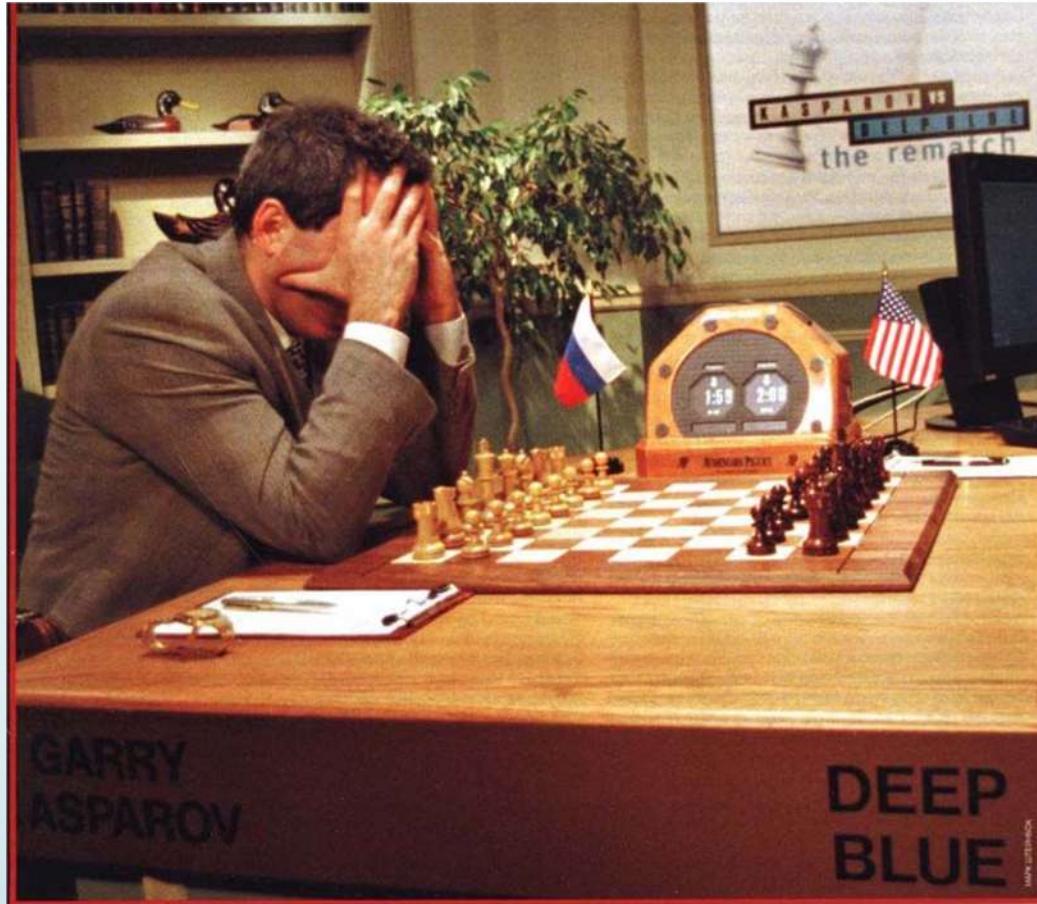
MCTS History

- ▶ In 2006, Rémi Coulom described the Monte Carlo method to game-tree search and coined the name Monte Carlo tree search
- ▶ The Monte Carlo method, which uses random sampling for deterministic problems which are difficult or impossible to solve using other approaches, dates back to the 1940s

Idea

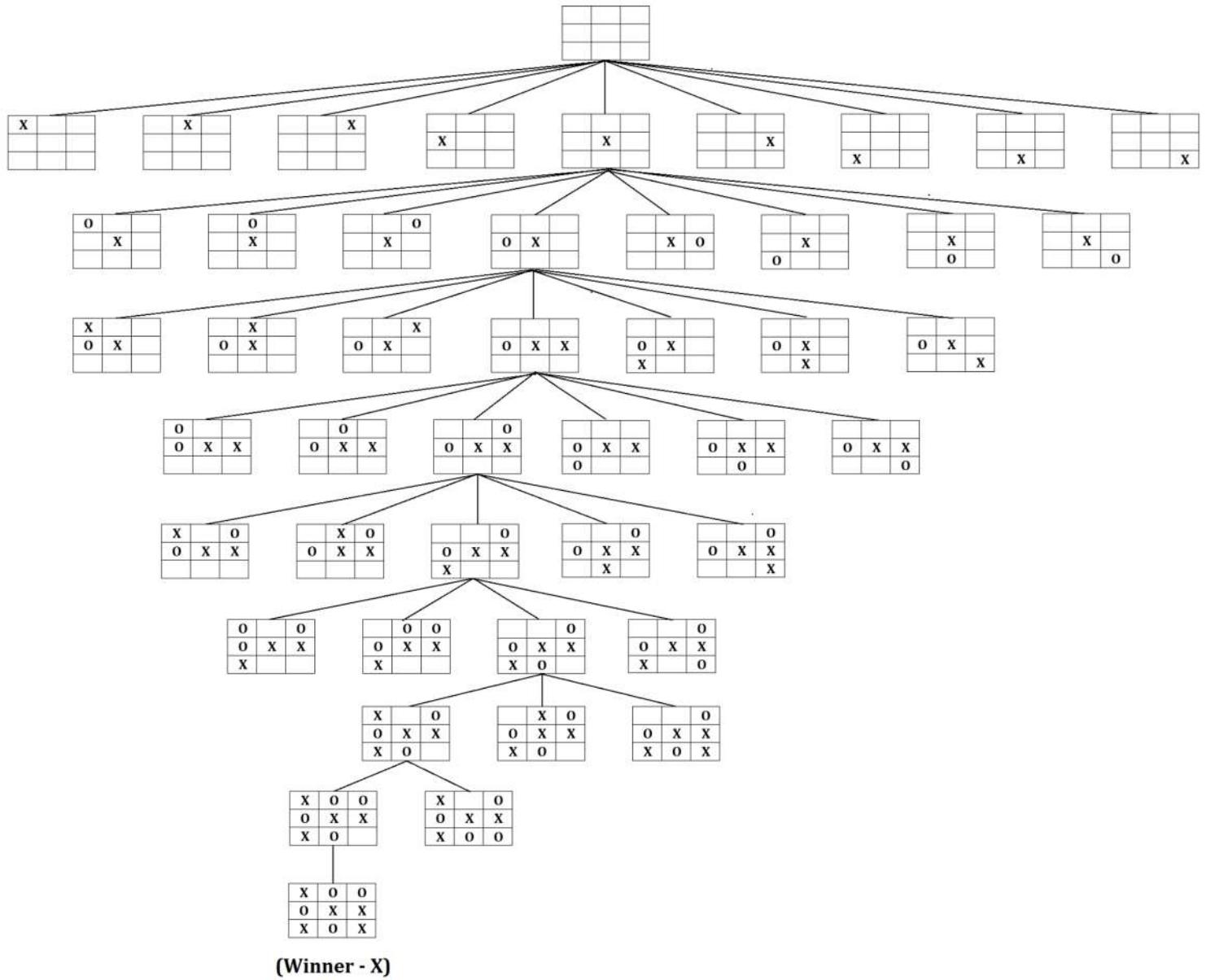
- ▶ The focus of MCTS is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space
- ▶ The application of MCTS in games is based on many *playouts*. In each playout, the game is played out to the very end by selecting moves at random
- ▶ The final game result of each playout is then used to weight the nodes in the game tree so that better nodes are more likely to be chosen in future playouts

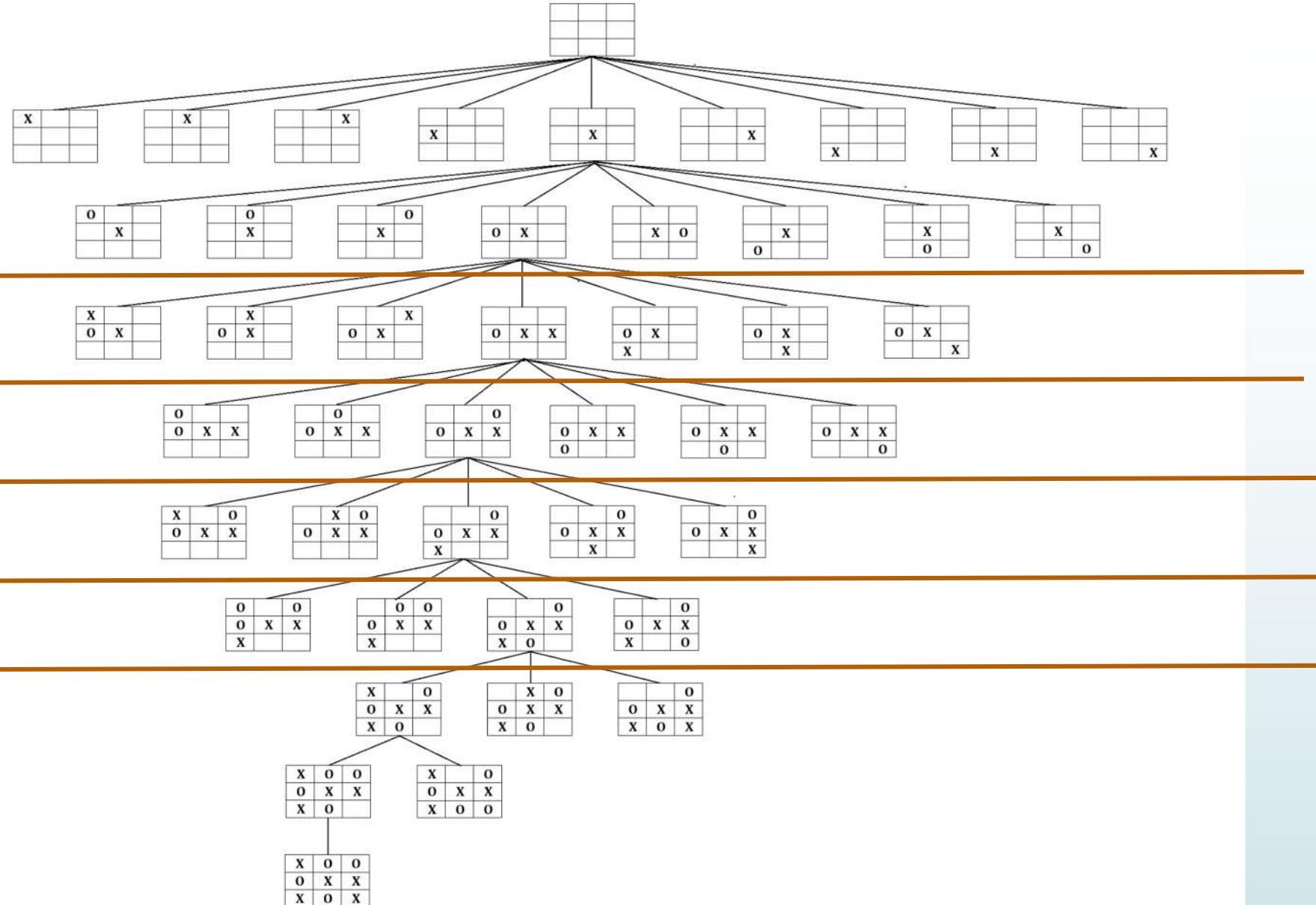
What makes MCTS different from past decision making algorithms like Minimax?



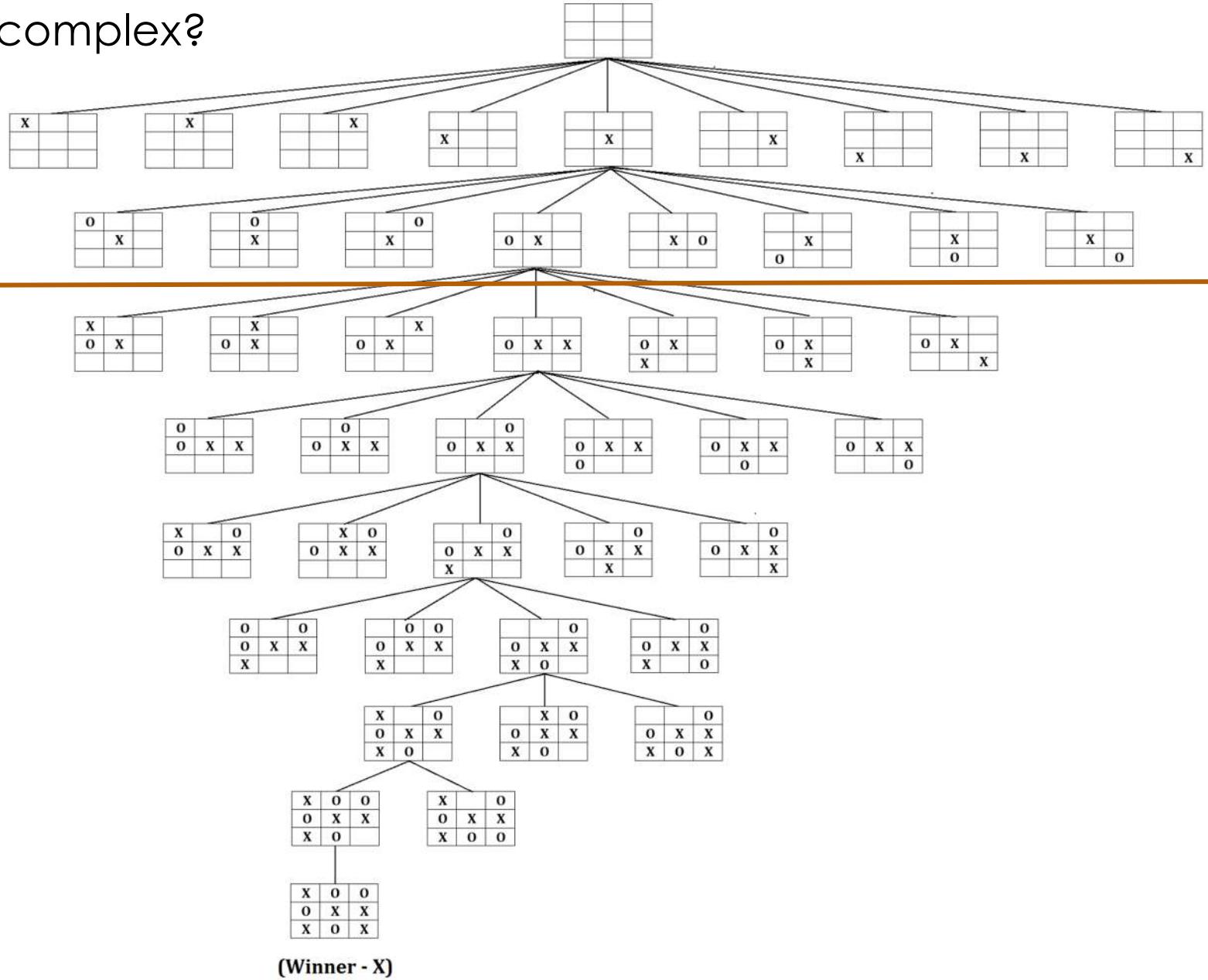
Minimax and Game

A Game tree is a directed graph whose nodes are positions in a game





What if game tree is too complex?



GO

- Go game tree complexity 10^{360} VS chess 10^{123}
- For ~20 years after Deep Blue (chess AI) no AI could master Go



2016 Alpha Go and the MCTS



Monte Carlo Tree Search

- ▶ How could it plan ahead when there are so many potential moves in Go?
- ▶ BRUTE FORCE minimax method from chess no longer applicable here

Monte Carlo Tree Search

- ▶ MCTS builds a statistics tree (detailling value of nodes) that partially maps onto the entire game tree
- ▶ Statistics tree guides the AI to look only/mostly at the most interesting nodes in the game tree
- ▶ Value of nodes determined by simulation

Monte Carlo Tree Search

- ▶ The most basic way to use playouts is to apply the same number of playouts after each legal move of the current player, then choosing the move which led to the most victories. The efficiency of this method — called *Pure Monte Carlo Game Search* — often increases with time as more playouts are assigned to the moves that have frequently resulted in the player's victory (in previous playouts)
- ▶ Full Monte Carlo tree search employs this principle recursively on many depths of the game tree. Each round of MCTS consists of four steps:
 1. Selection
 2. Expansion
 3. Simulation
 4. Backpropagation

Steps

1. Selection

- ▶ Start from root R and select successive child nodes down to a leaf node L . The section below says more about a way of choosing child nodes that lets the game tree expand towards most promising moves, which is the essence of MCTS

2. Expansion

- ▶ Unless L ends the game with a win/loss for either player, either create one or more child nodes or choose from them node C

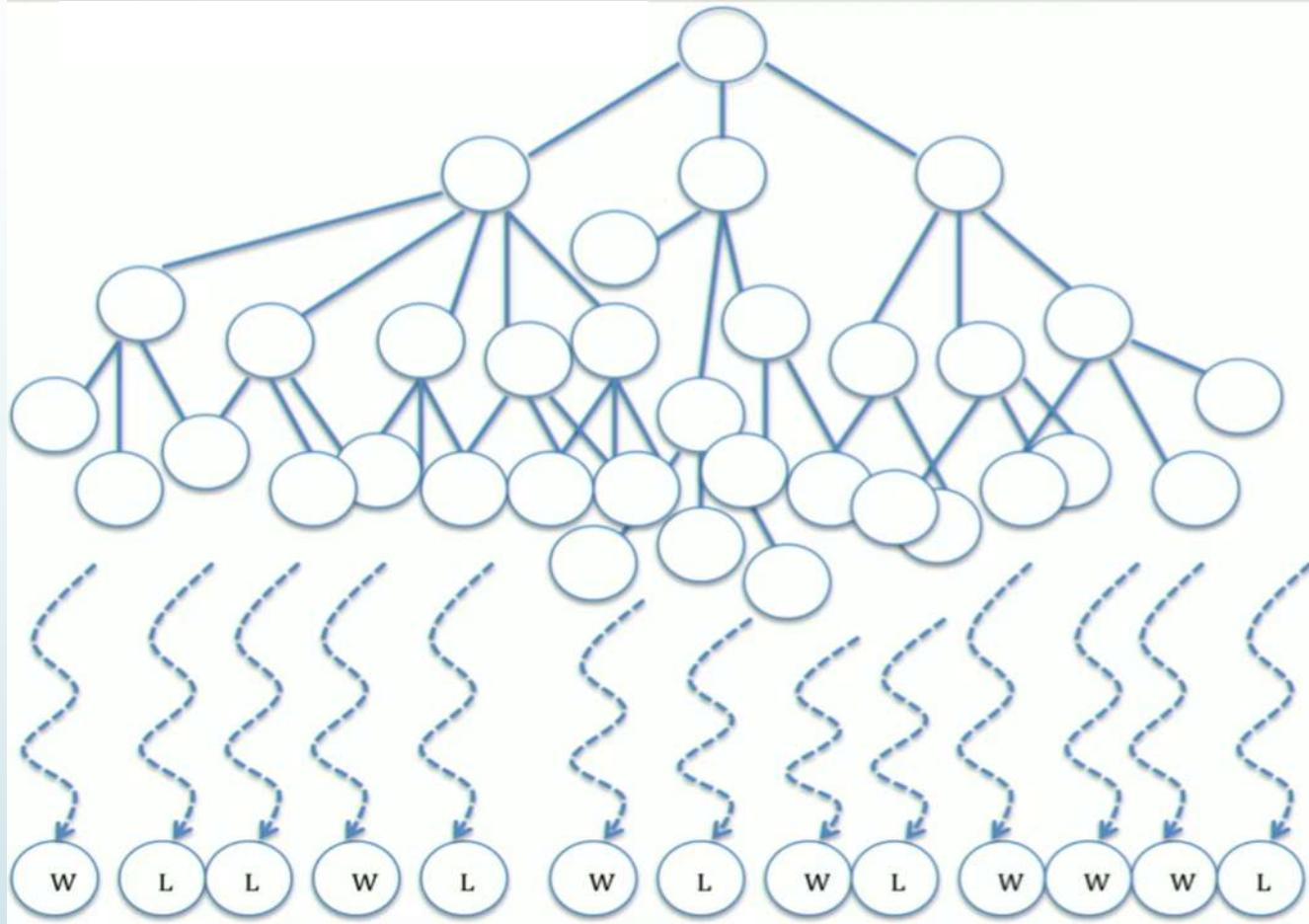
3. Simulation

- ▶ Play a random playout from node C . This step is sometimes also called playout or rollout

4. Backpropagation

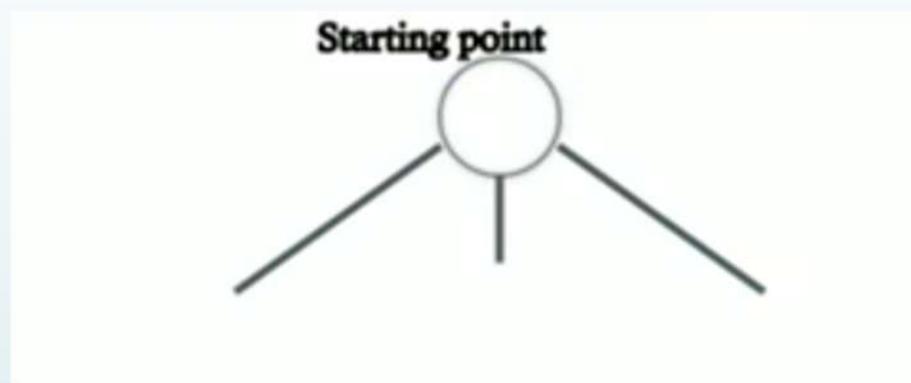
- ▶ Use the result of the playout to update information in the nodes on the path from C to R

Hypothetical Game Tree



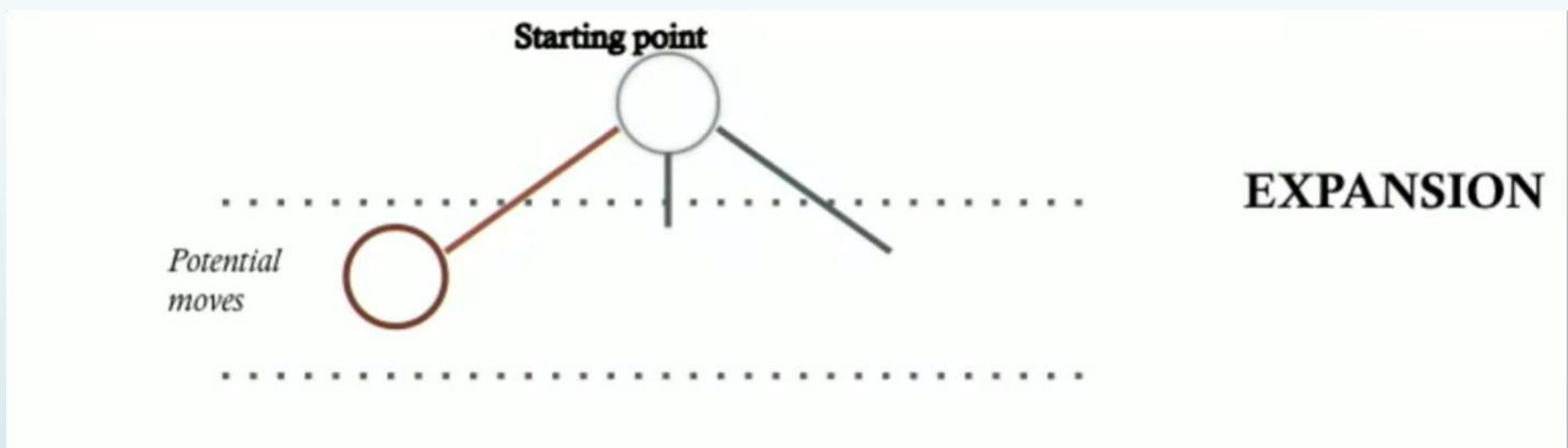
Selection, Expansion, Simulation, Backpropagation

- MCTS begins its stats tree

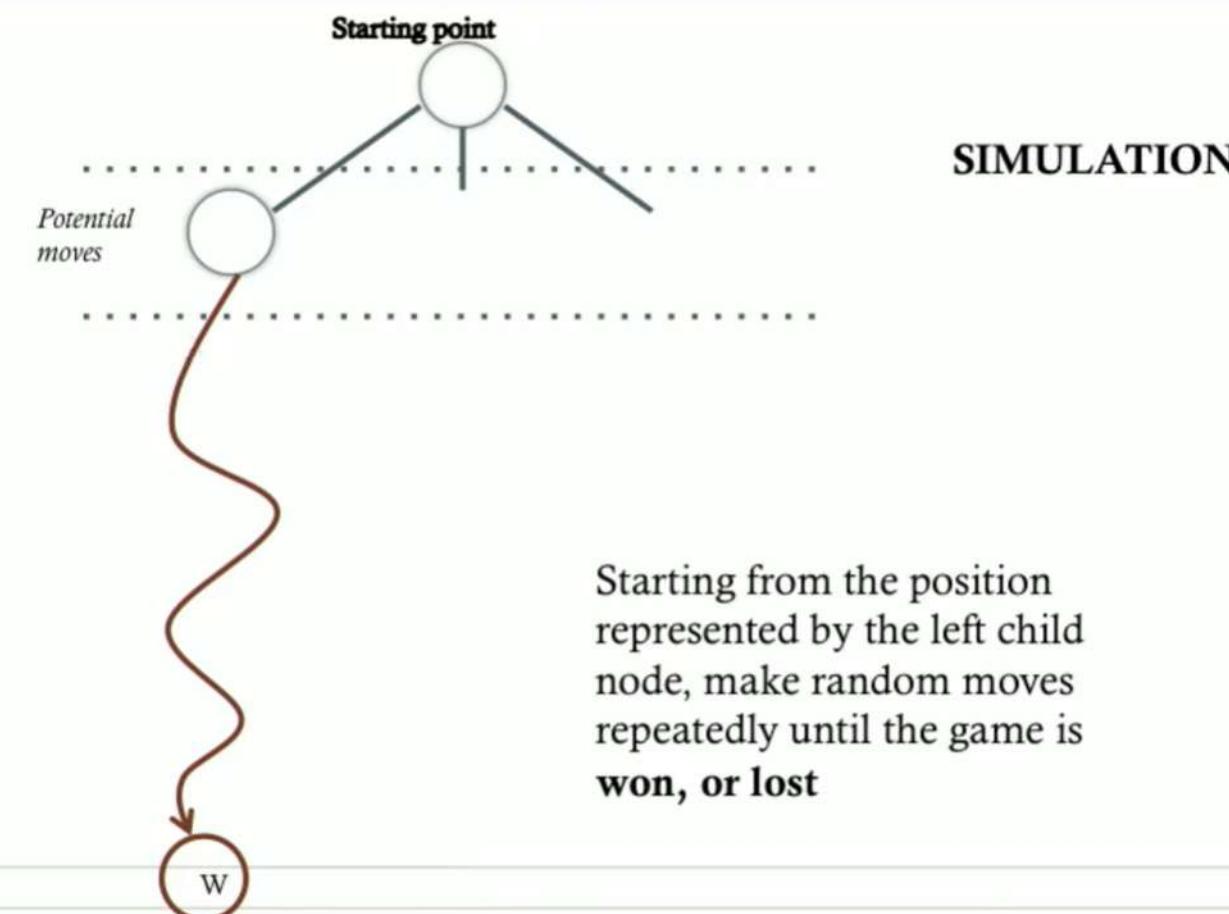


Selection, Expansion, Simulation, Backpropagation

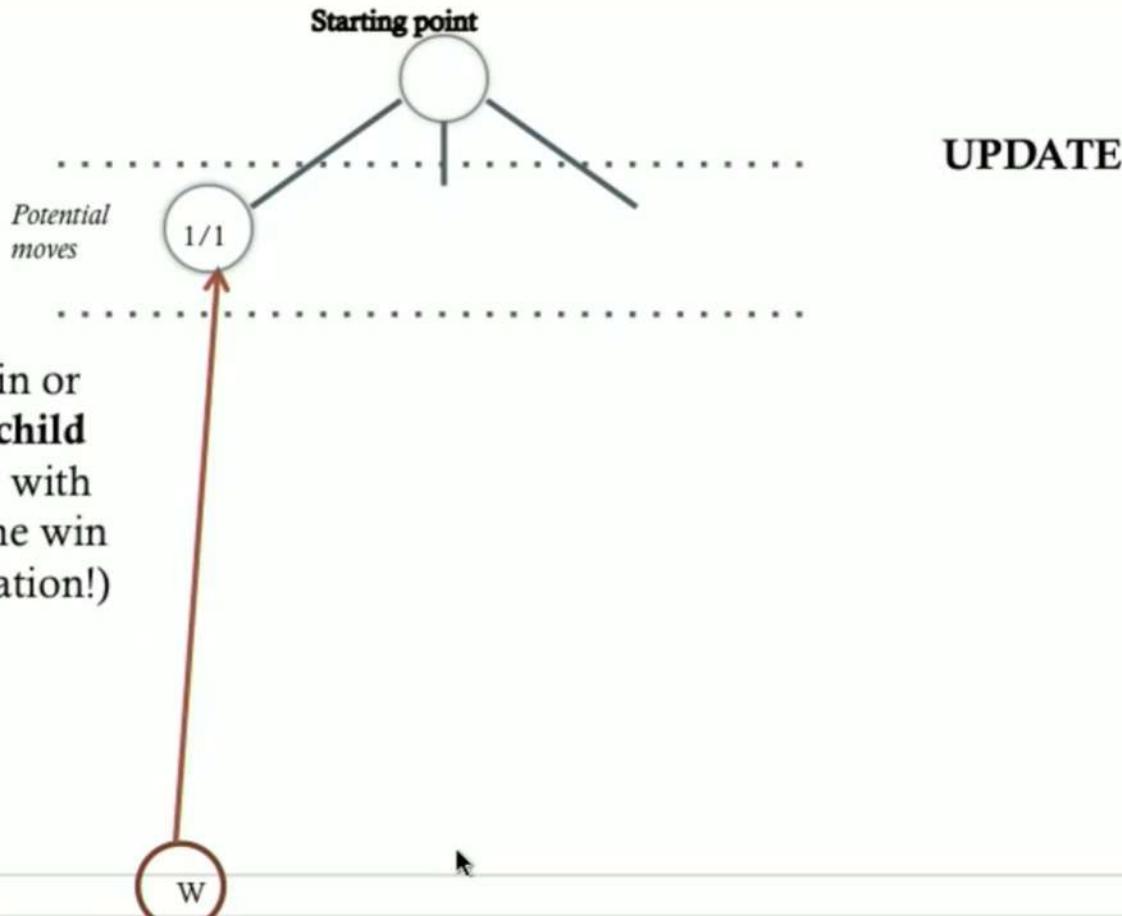
- ▶ Add a new node to the stats tree, representing a position in the game tree that the AI will investigate (how good the move is) next



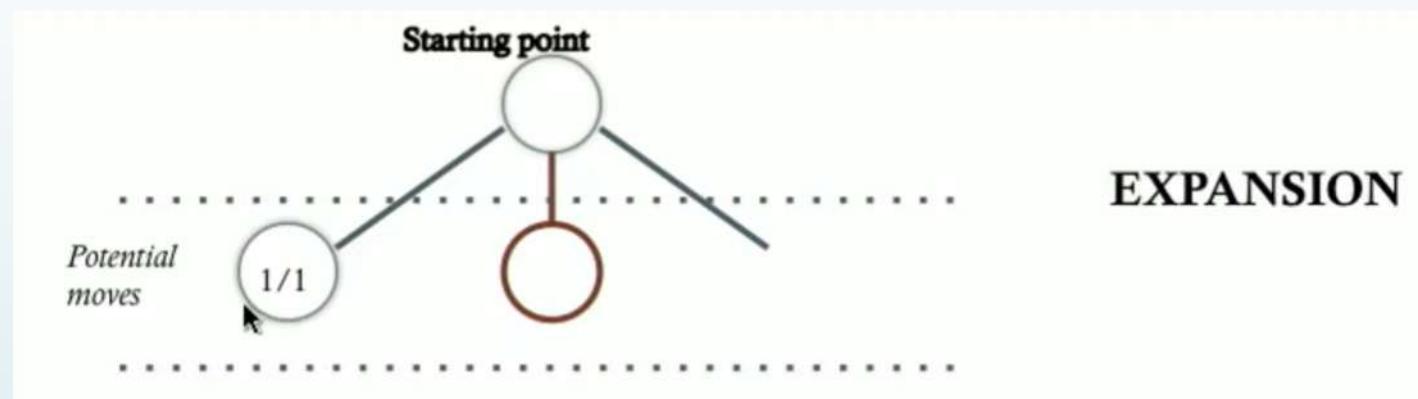
Selection, Expansion, Simulation, Backpropagation



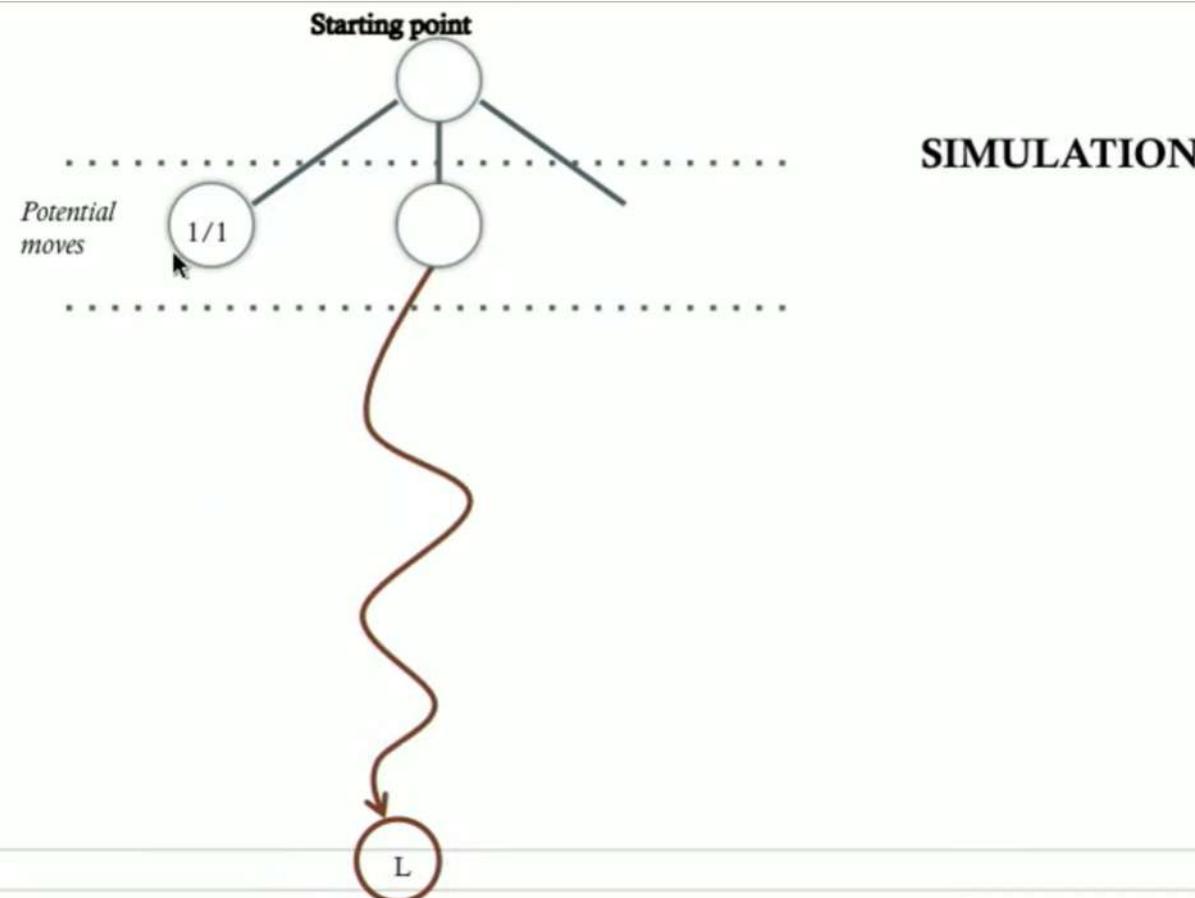
Selection, Expansion, Simulation, Backpropagation



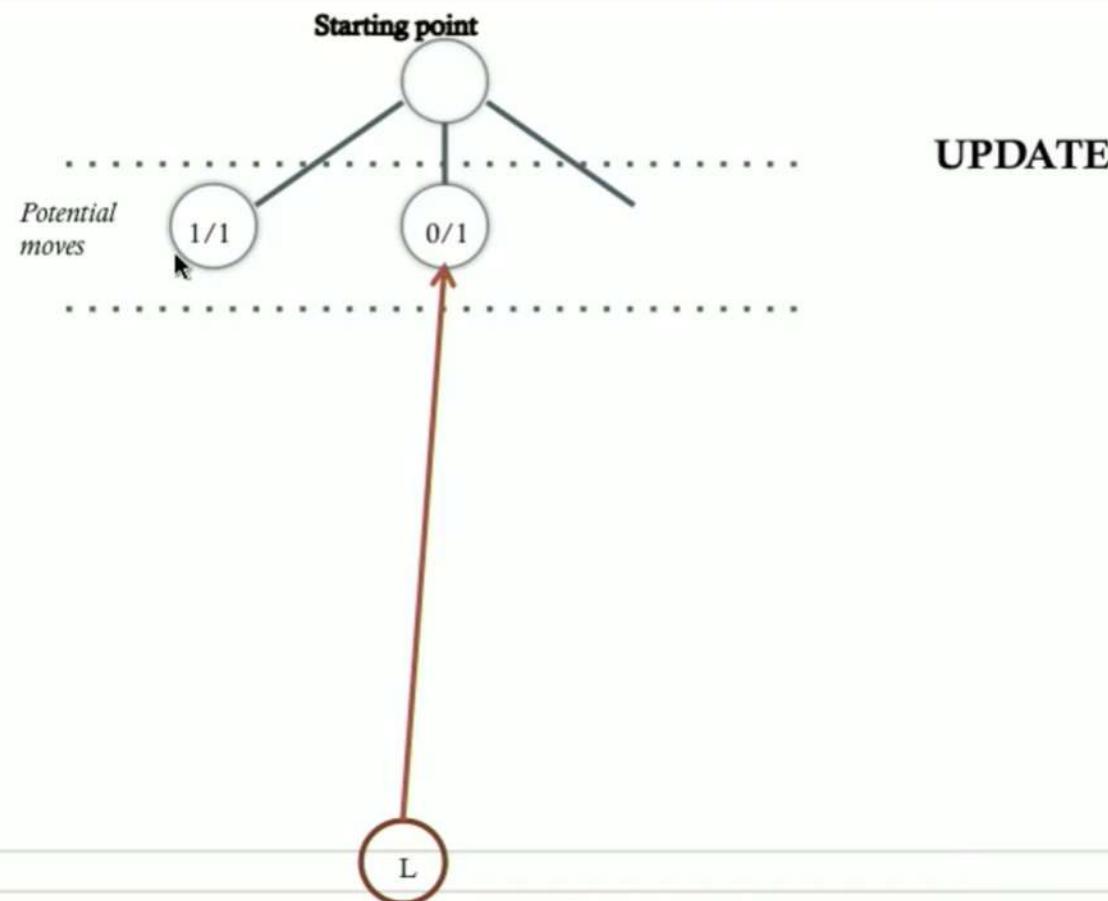
Selection, Expansion, Simulation, Backpropagation



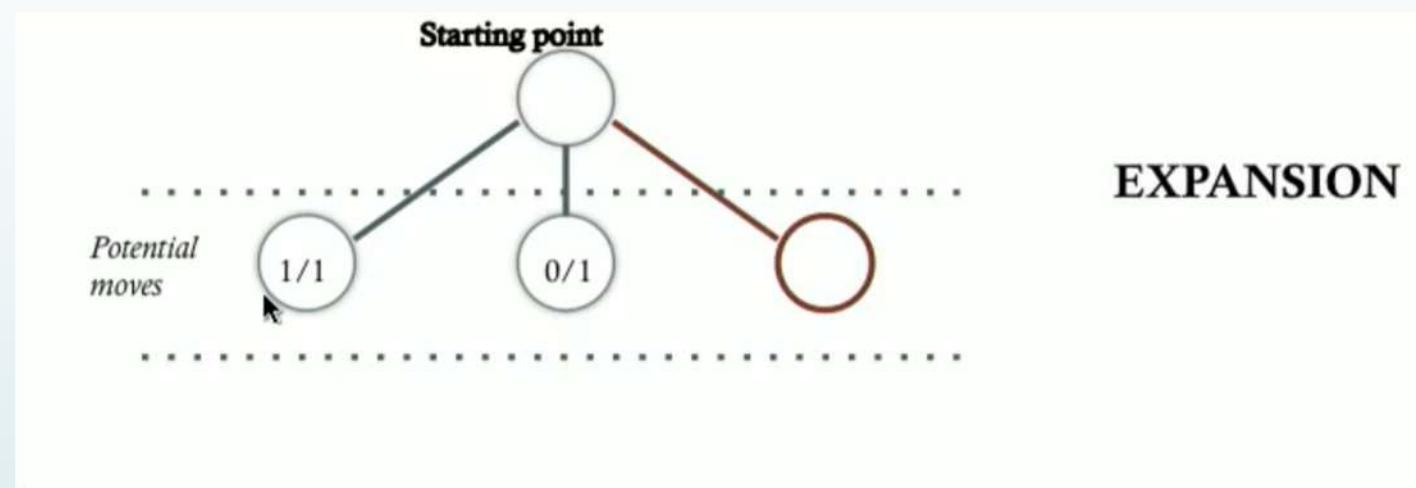
Selection, Expansion, Simulation, Backpropagation



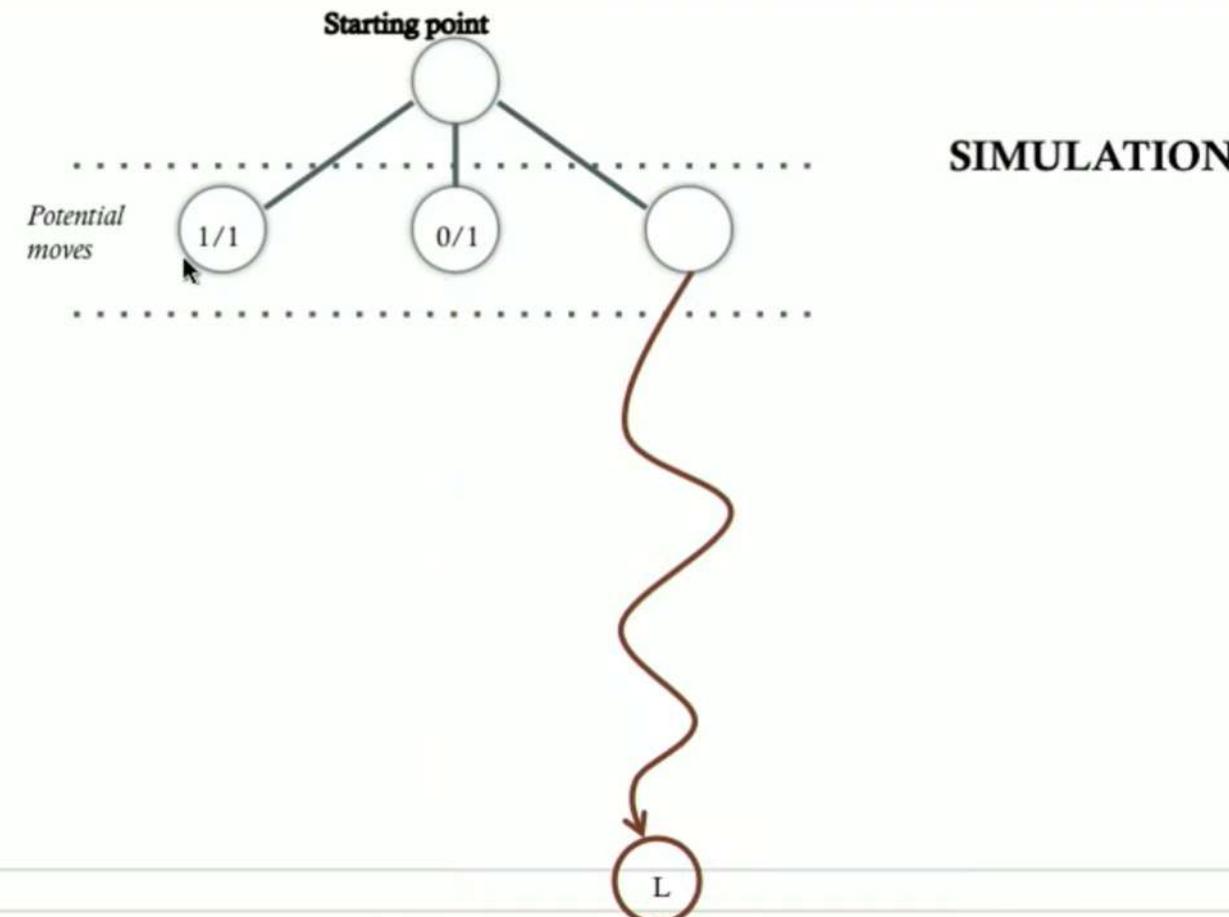
Selection, Expansion, Simulation, Backpropagation



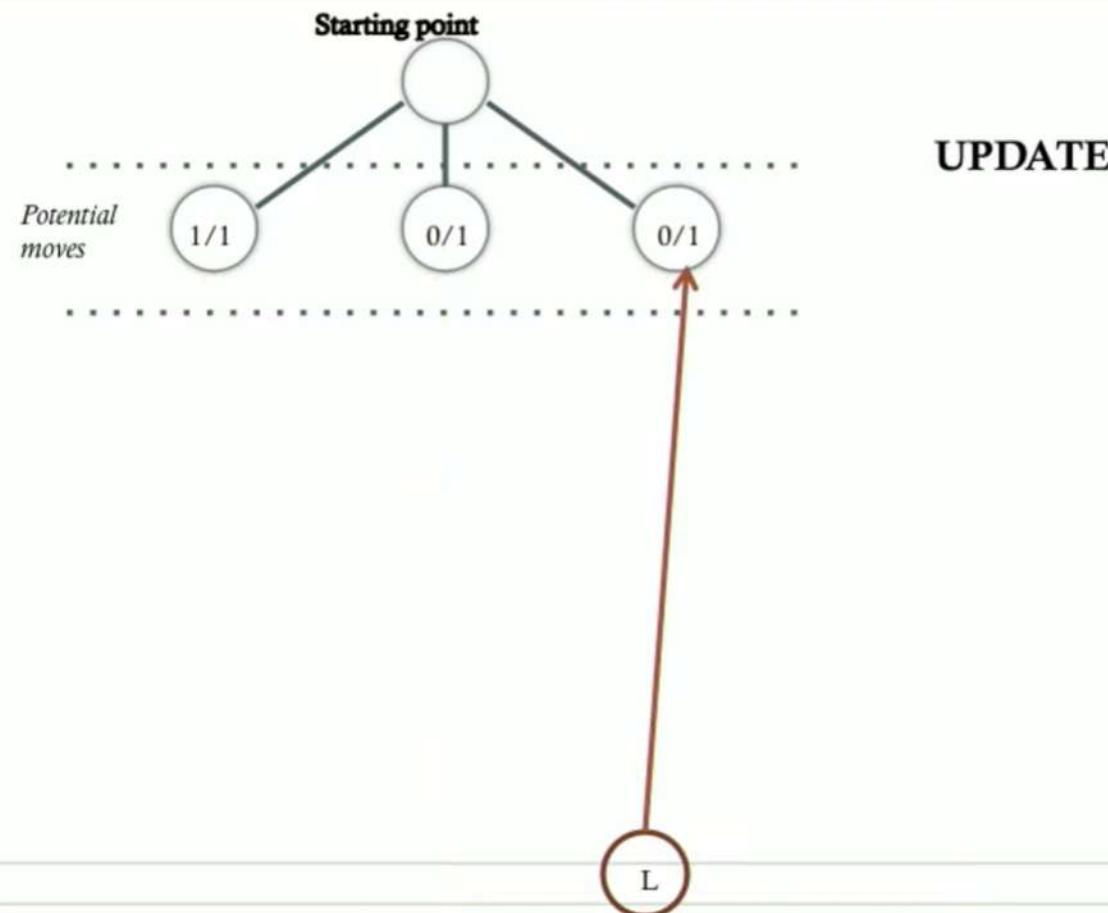
Selection, Expansion, Simulation, Backpropagation



Selection, Expansion, Simulation, Backpropagation

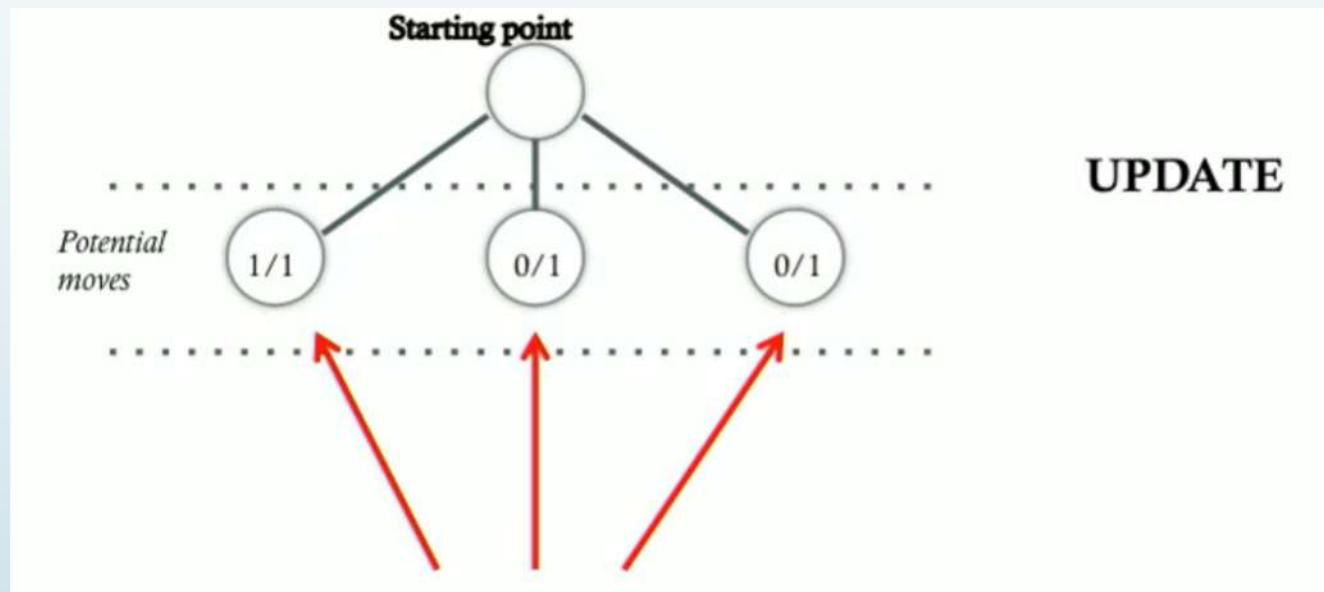


Selection, Expansion, Simulation, Backpropagation



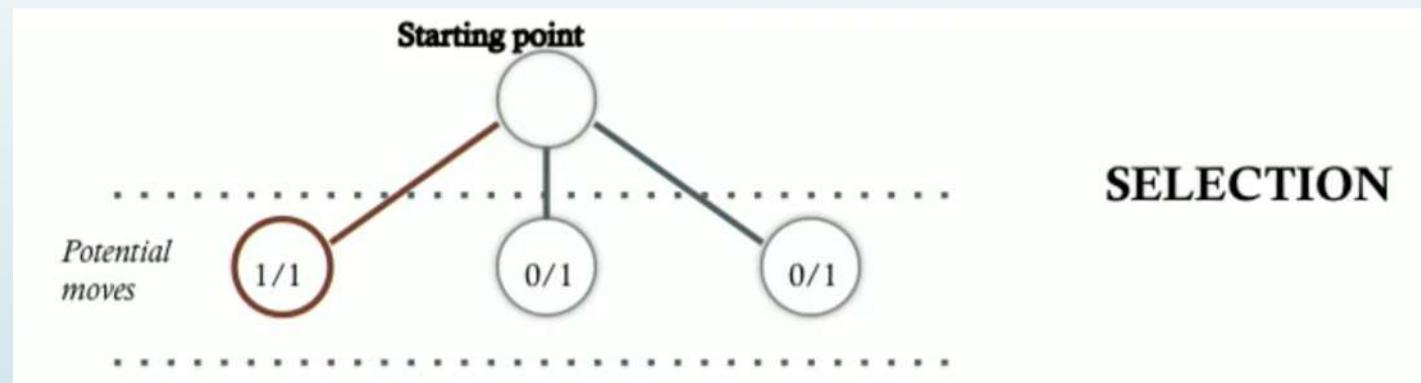
Selection, Expansion, Simulation, Backpropagation

- Left node seems to be a better move, but chances are, these numbers (produced by a single random simulation) are probably not a good indicator of how good any of the moves are
- More simulations will make them more accurate!



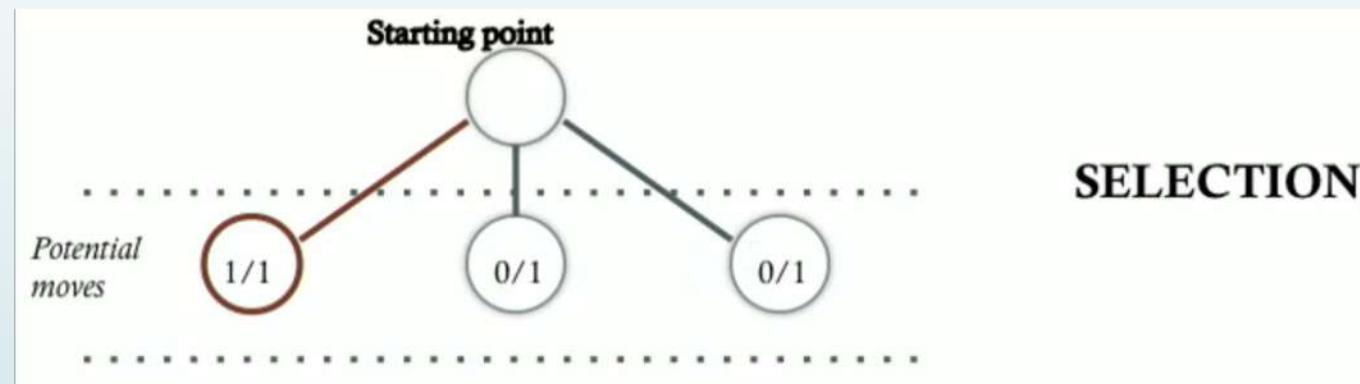
Selection, Expansion, Simulation, Backpropagation

- ▶ All child nodes have now been visited at least once. Now AI can select which child node to be investigated further
- ▶ Remember the higher the value, the better the move is (for producing a win)

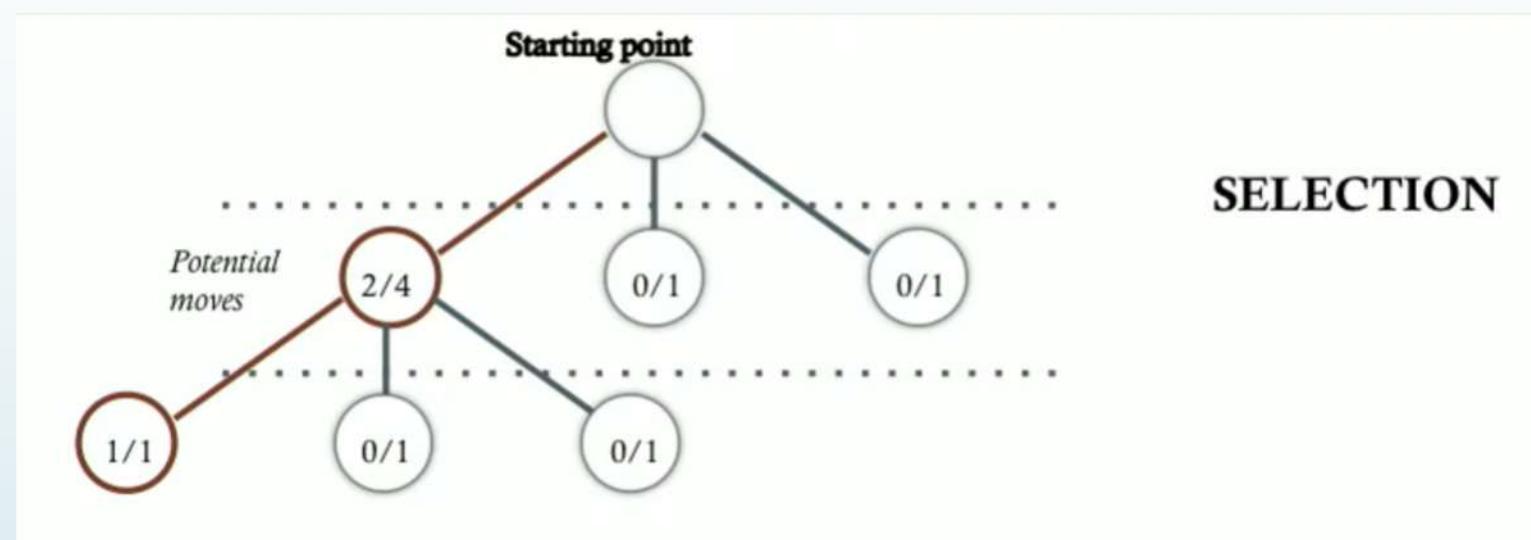


Selection, Expansion, Simulation, Backpropagation

- ▶ Selection is based on 2 things:
 - ▶ How good are the statistics?
 - ▶ How much has child node been “ignored”

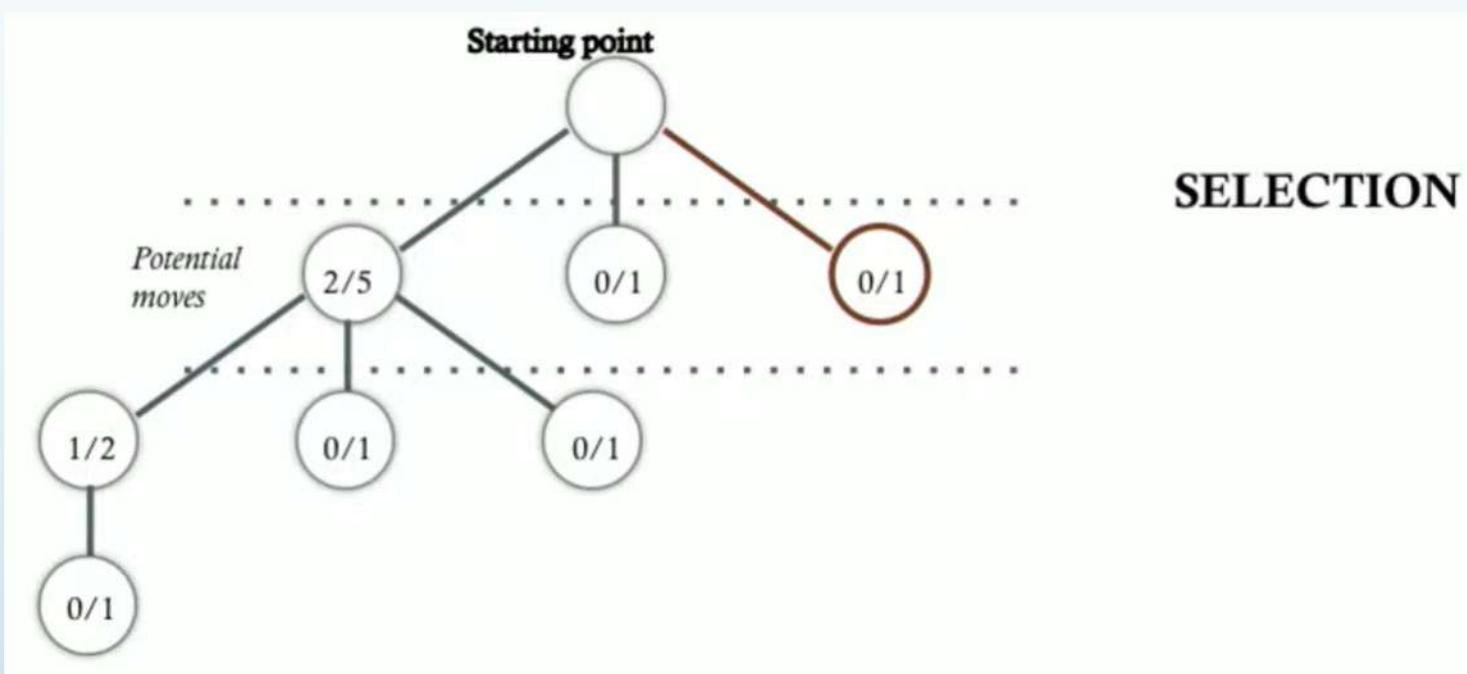


Selection, Expansion, Simulation, Backpropagation



Selection, Expansion, Simulation, Backpropagation

- ▶ These simulations can go on for as long as you want. The more simulation there are, the bigger the stats tree, and the more accurate the node values are



What is the Solution?

- ▶ Usually, the node with the highest denominator should be the solution
- ▶ In other words, the node with the biggest number of simulations
- ▶ The number of simulations should be big enough for statistical independence

Exploration vs Exploitation

- ▶ The main difficulty in selecting child nodes is maintaining some balance between the *exploitation* of deep variants after moves with high average win rate and the *exploration* of moves with few simulations
- ▶ It is recommended to select nodes with the highest value of the next formula called UCT (*Upper Confidence Bound 1 applied to trees*)

$$\frac{w_i}{n_i} + c \sqrt{\frac{\ln N_i}{n_i}}$$

- ▶ Exploitation is the first term
- ▶ Exploration is the second term

Exploration vs Exploitation

$$\frac{w_i}{n_i} + c \sqrt{\frac{\ln N_i}{n_i}}$$

- ▶ w_i stands for the number of wins for the node considered after the i -th move
- ▶ n_i stands for the number of simulations for the node considered after the i -th move
- ▶ N_i stands for the total number of simulations after the i -th move run by the parent node of the one considered
- ▶ c is the exploration parameter — theoretically equal to $\sqrt{2}$; in practice usually chosen empirically

Advantages and Disadvantages

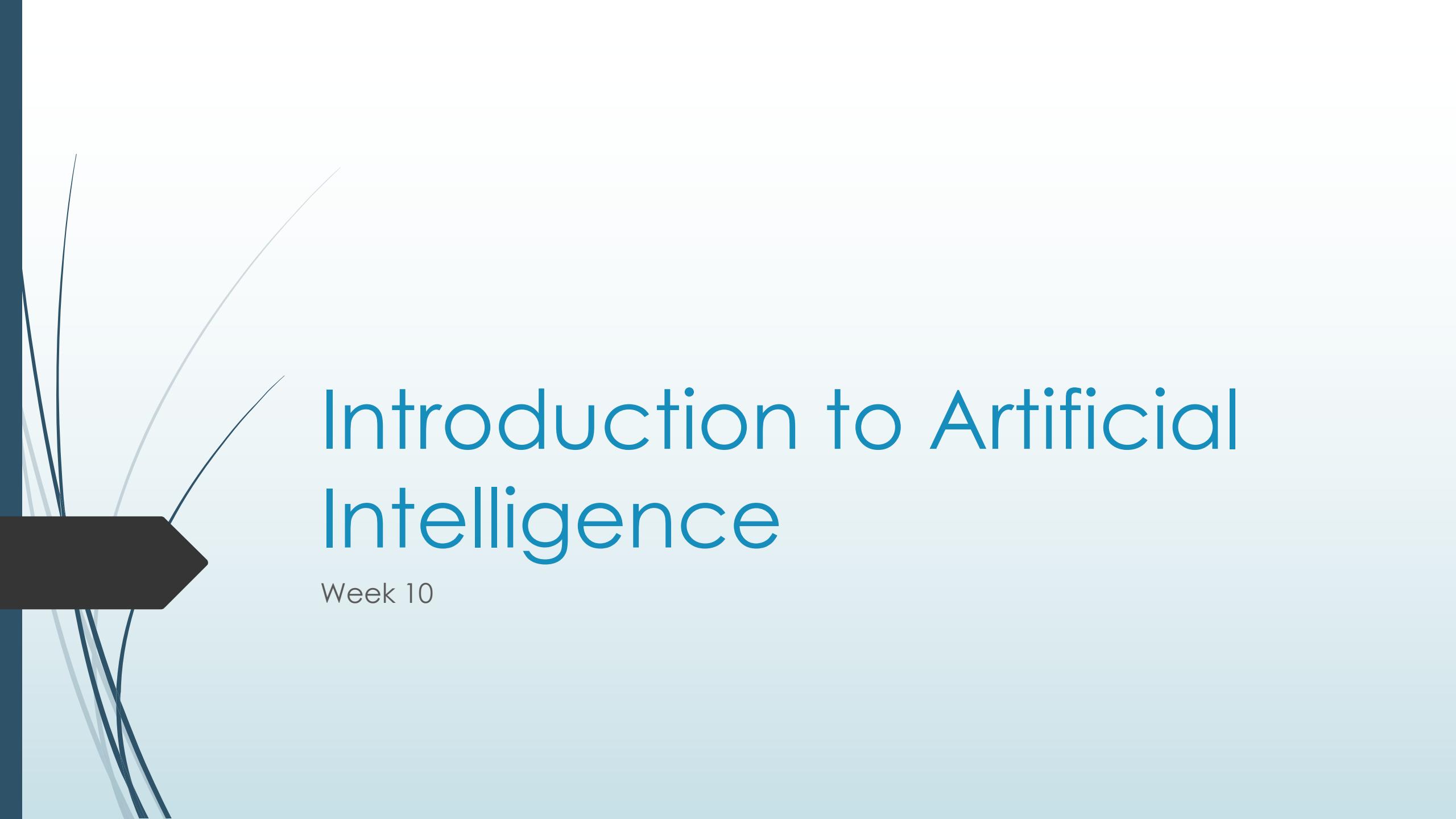
- ▶ Advantages
 - ▶ No evaluation function that makes the algorithm ultimate for different tasks
 - ▶ Achieves better results than classical algorithms in games with a high branching factor
 - ▶ Shortens the search time for large tasks
- ▶ Disadvantages
 - ▶ In certain positions, there may be moves that look superficially strong, but that actually lead to a loss via a subtle line of play

Parallelization

- ▶ Leaf parallelization
 - ▶ Parallel execution of many playouts from one leaf of the game tree
- ▶ Root parallelization
 - ▶ Building independent game trees in parallel and making the move based on the root-level branches of all these trees
- ▶ Tree parallelization
 - ▶ Parallel building of the same game tree, protecting data from simultaneous writes either with one, global mutex, with more mutexes, or with non-blocking synchronization

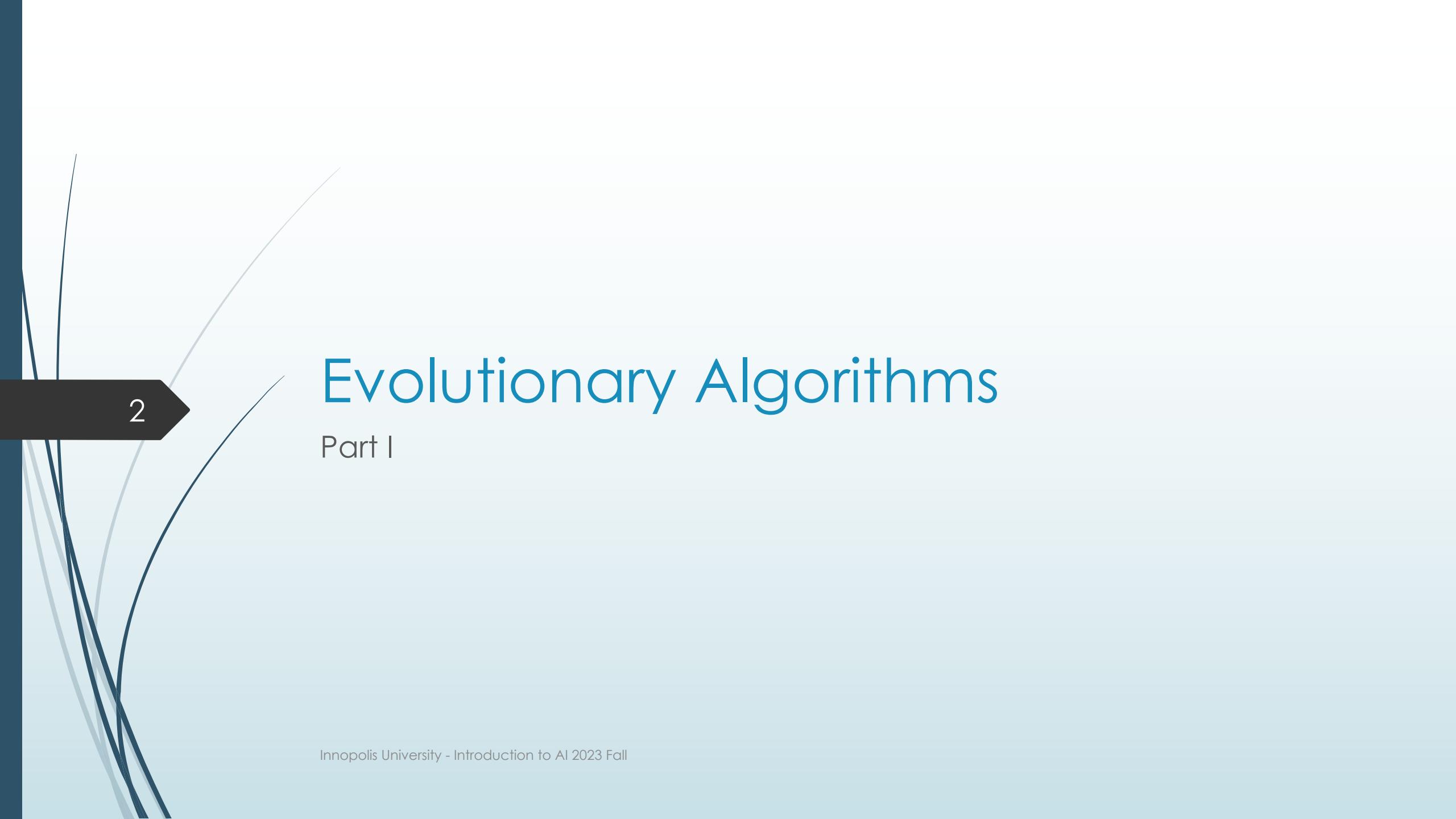
References

- ▶ [MCTS](#)
- ▶ [MCTS on Tic-Tac-Toe](#)



Introduction to Artificial Intelligence

Week 10



2

Evolutionary Algorithms

Part I

Materials

- ▶ Evolutionary Computation for Modeling and Optimization – D. Ashlock

Evolutionary Algorithms

- ▶ Use the concepts of the Neo-Darwinian Synthesis or Lamarckian Evolution
 - ▶ Natural Selection
 - ▶ Inheritable Traits
 - ▶ Fitness Biased Reproduction
 - ▶ Fitness evaluates individuals
 - ▶ Generational/Time Series
- ▶ Four major overarching techniques discovered about 1980
 - ▶ Genetic Algorithms - Holland
 - ▶ Genetic Programming - Koza
 - ▶ Evolutionary Programming - Fogel
 - ▶ Evolutionary Strategies – Rechenbreg/Schwefel
- ▶ Large arguments about priority of technique leads to a compromise on the title of Evolutionary Algorithms – schisms still fighting for dominance

EA System

- ▶ Create a randomized **population** made up of **chromosomes**, data structures which encode a potential solution
- ▶ Until <Done>, based on a **stopping criteria**
 - ▶ Find an **objective/fitness score** for each member of the population
 - ▶ **Select members** to act upon using some **variation operators**
 - ▶ Apply operations on the members
 - ▶ **Crossover**
 - ▶ **Mutations**
 - ▶ Replace some members of the population with these children from the variation operators
 - ▶ Keep some members from the previous population in the new population, i.e. **elitism/inheritance**

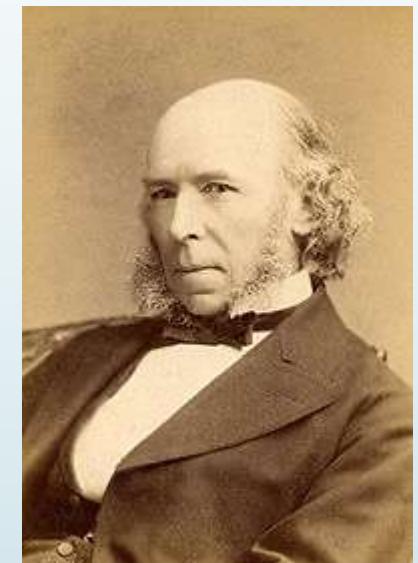
Selection

- ▶ Cartoon of the ideas of Natural Selection by Darwin
- ▶ Provides a fitness biased method of keeping good structures
 - ▶ Note: BIASED not based
 - ▶ We can still accept 'worst' choices
- ▶ Structures which have a higher fitness on the objective score are more likely to continue on in the population



Survival of the Fittest

- ▶ Major misconceptions in the application of this phase
- ▶ Darwin didn't coin it – nor was it used until the 5th edition of "The Origin of Species" (1869)
- ▶ Used by Herbert Spencer in Principles of Biology in 1864
 - ▶ "This survival of the fittest, which I have here sought to express in mechanical terms, is that which Mr. Darwin has called 'natural selection', or the preservation of favoured races in the struggle for life."
 - ▶ Darwin's use was based on the fitness of a creature to survive in a local environment

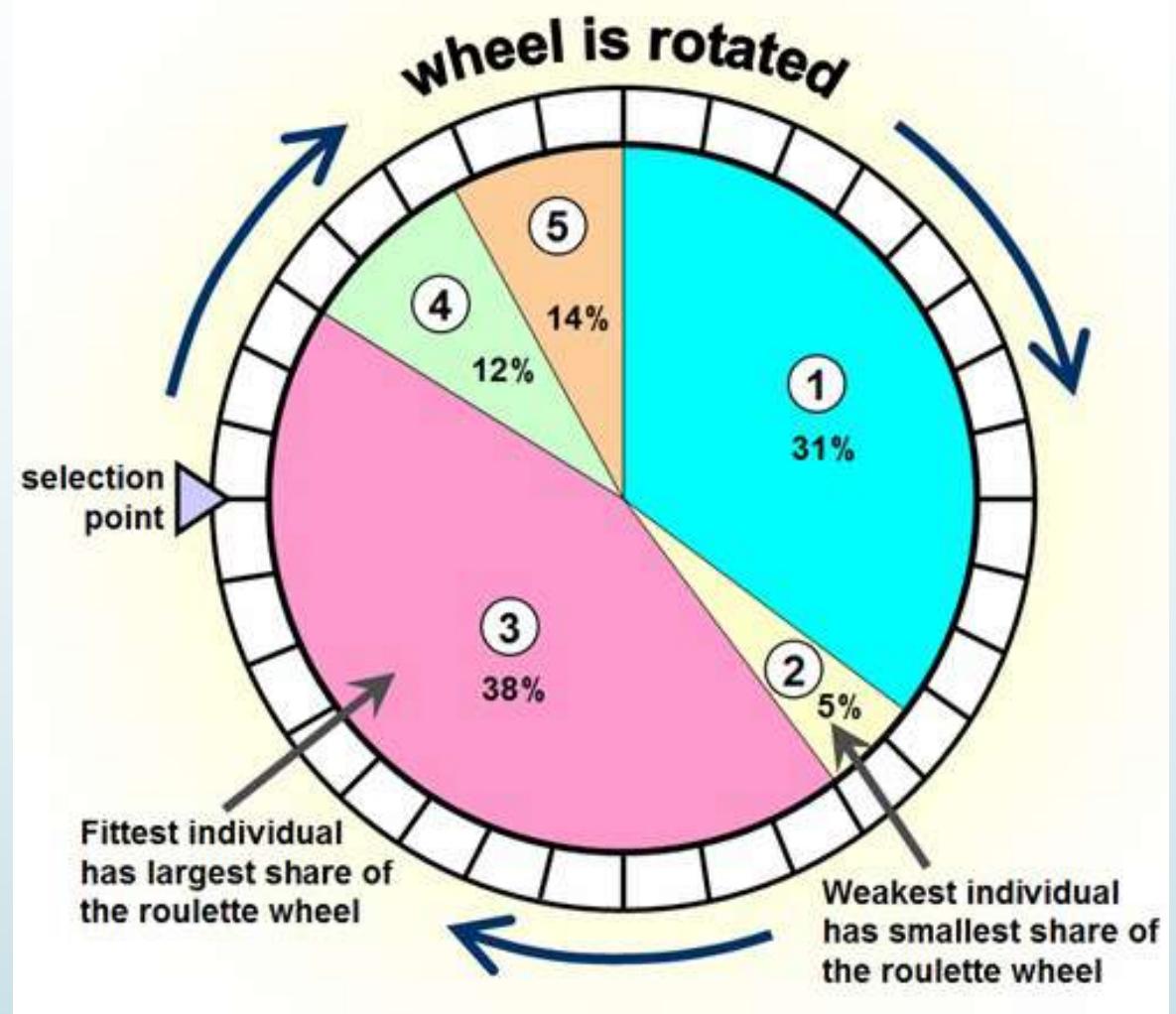


Biological Fitness

- ▶ The phrase seems to imply that there is an innate idea of what is FIT/UNFIT
- ▶ *Post Hoc Ergo Propter Hoc Fallacy* or *Post Hoc Fallacy*
 - ▶ The creature survived as it was fit
 - ▶ The creature is fit because it has survived
- ▶ Biological Fitness is defined as the number of offspring which reach sexual maturity and are able to pass along their genes
- ▶ Evolutionary Algorithms fall under this misconception – we apply fitness as a post hoc

Fitness Proportional (Roulette Wheel)

- ▶ Each member is given a section of the wheel in relation to their fitness score
- ▶ Member fitness/ Sum of all member fitness
- ▶ Wheel is spun for a number of times
- ▶ Winners Breed Together



Tournament

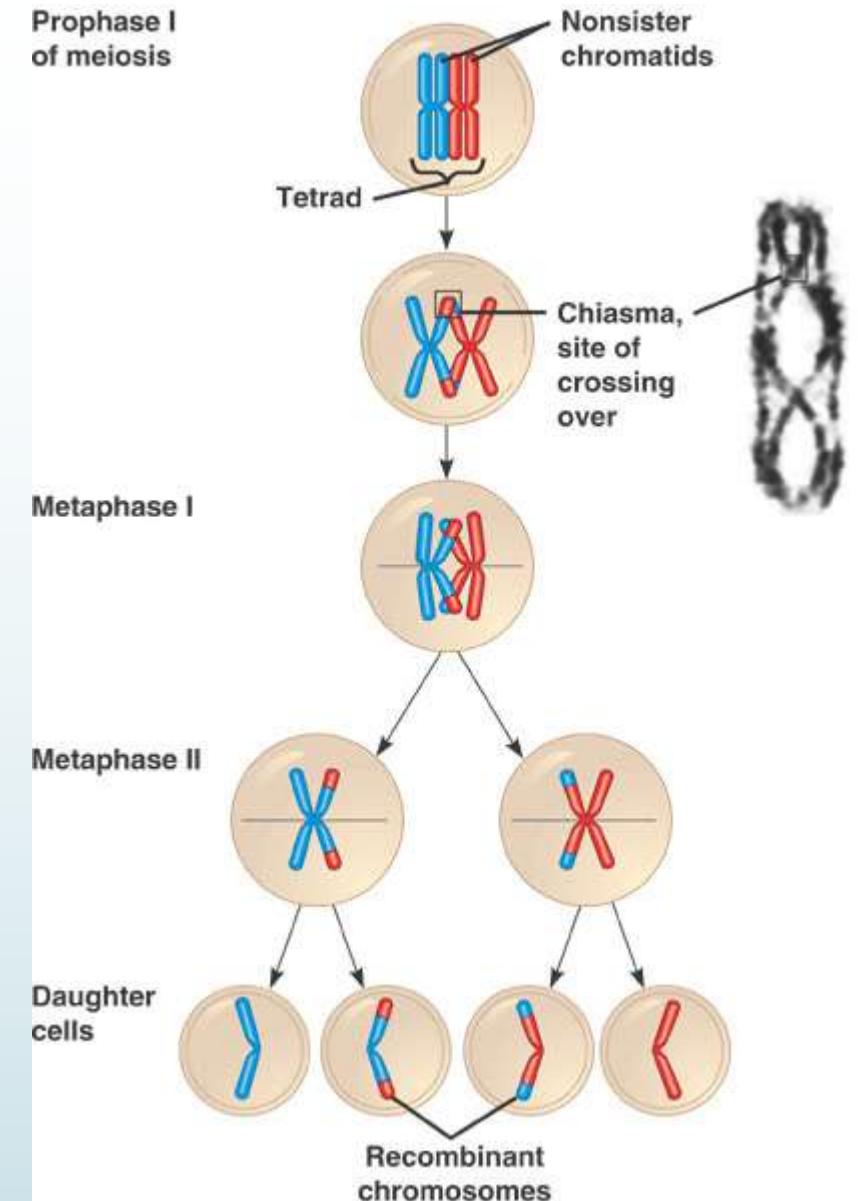
- ▶ A number of different manners are held for the construction of the challengers
 - ▶ At random (single tournament)
 - ▶ Groups of N (double tournament)
- ▶ Each of the structures in a tournament is compared and the most fit continues on to breed
- ▶ Fighting solutions
- ▶ Selection Pressure (the likelihood of only selecting from the higher fitness cohorts is a controllable feature)
 - ▶ Small Tournaments
 - ▶ Larger Tournaments

Genetic Algorithms (GA)

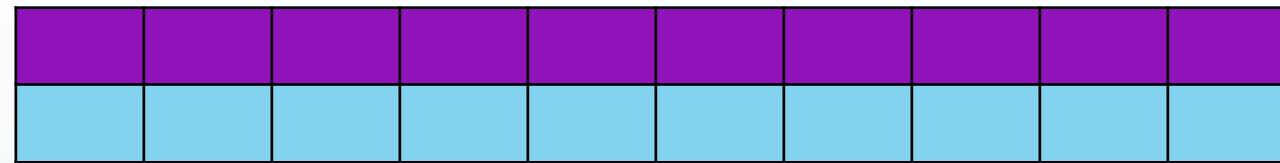
- ▶ Representation
 - ▶ Data Structure (commonly a discrete string)
- ▶ Selection
 - ▶ Fitness Proportional or Tournament
- ▶ Crossover
 - ▶ Yes. Data Structure Dependent
- ▶ Mutation
 - ▶ Yes. Data Structure Dependent, commonly a small change to a percentage of symbols in the string

Crossover in Biology

- ▶ Process of Meiosis
- ▶ Creation of **gamete** cells
 - ▶ Sex cells
 - ▶ From Greek for 'wife'
- ▶ Haploid creatures have chromosome pairs
- ▶ Biological crossover is NOT a representation of the actions happening in EAs

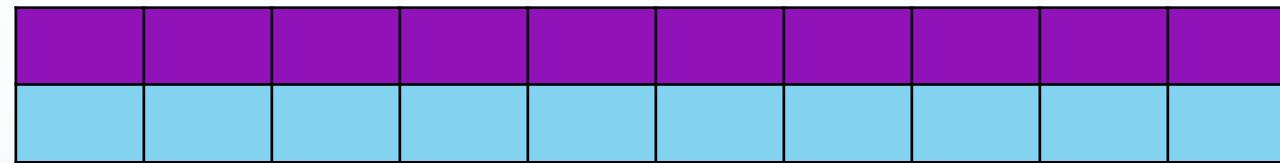


Crossover in a GA on Strings

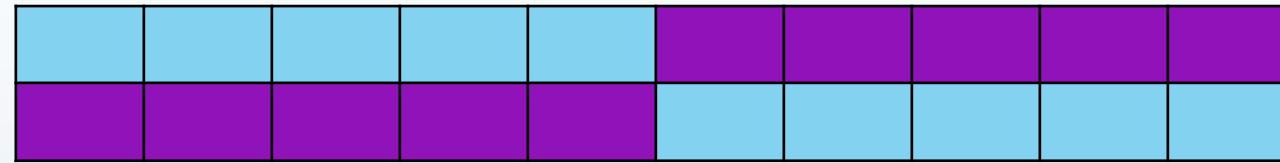


Two parent strings

Crossover in a GA on Strings

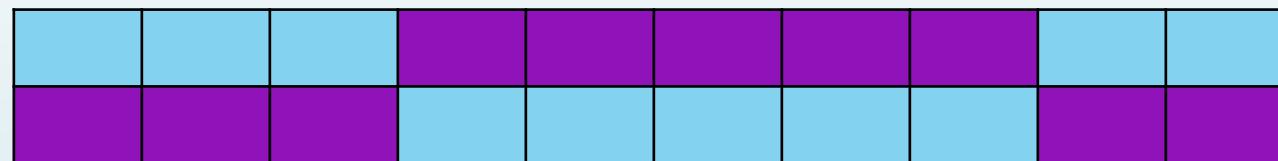
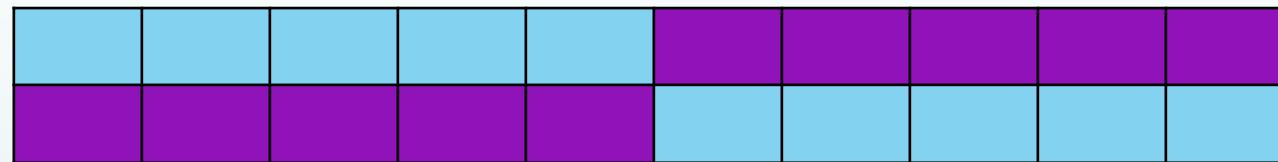
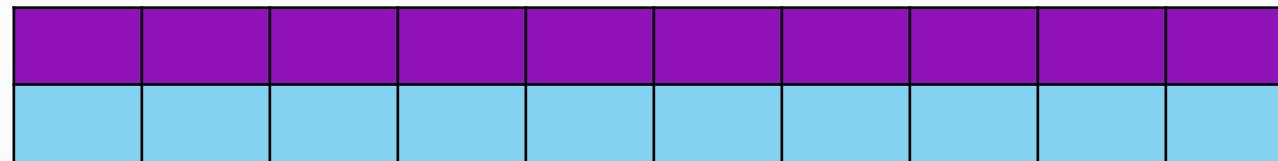


Two parent strings

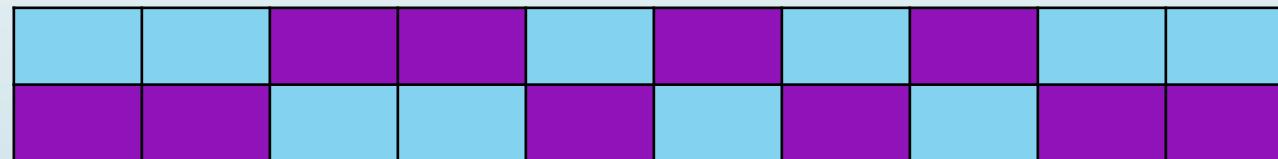
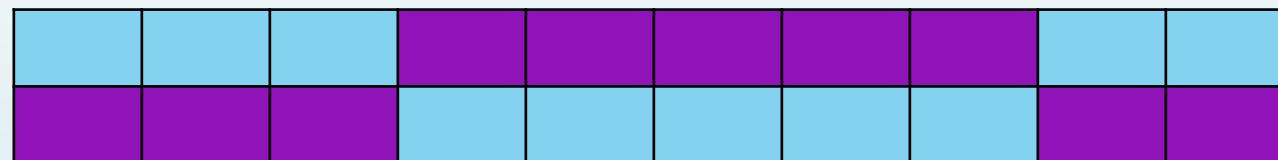
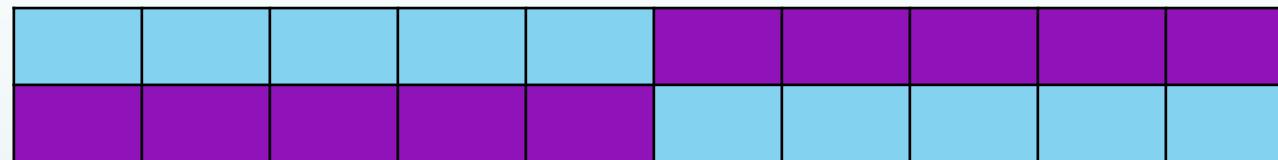
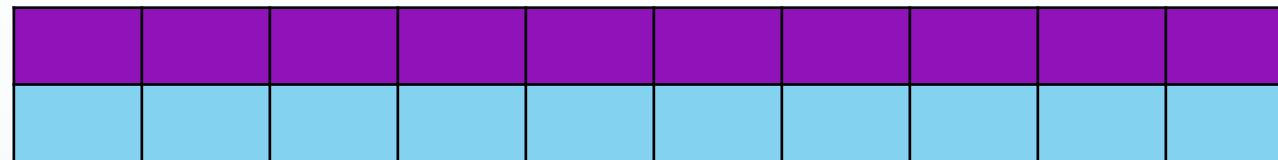


One Point – Select One Point at Random and Swap

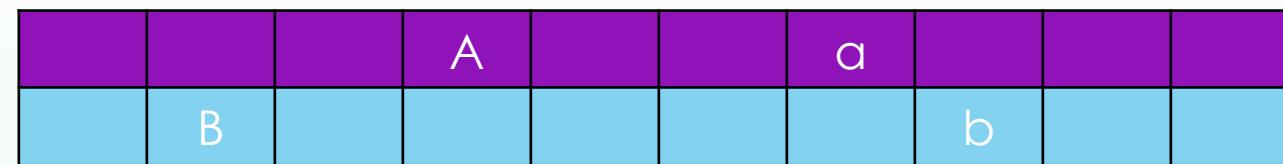
Crossover in a GA on Strings



Crossover in a GA on Strings



Mutation in a GA on Strings



Two separate strings

Mutation in a GA on Strings

			A			a			
	B						b		

Two separate strings

			a			a			
	B						B		

Point Mutation – Change the Symbol at a Loci to Some Other Symbol

Mutation in a GA on Strings

			A			a			
	B						b		

Two separate strings

			a			a			
	B						B		

Point Mutation – Change the Symbol at a Loci to Some Other Symbol

			a			A			
	b						B		

Swap Mutation – Swap Two Loci in the String

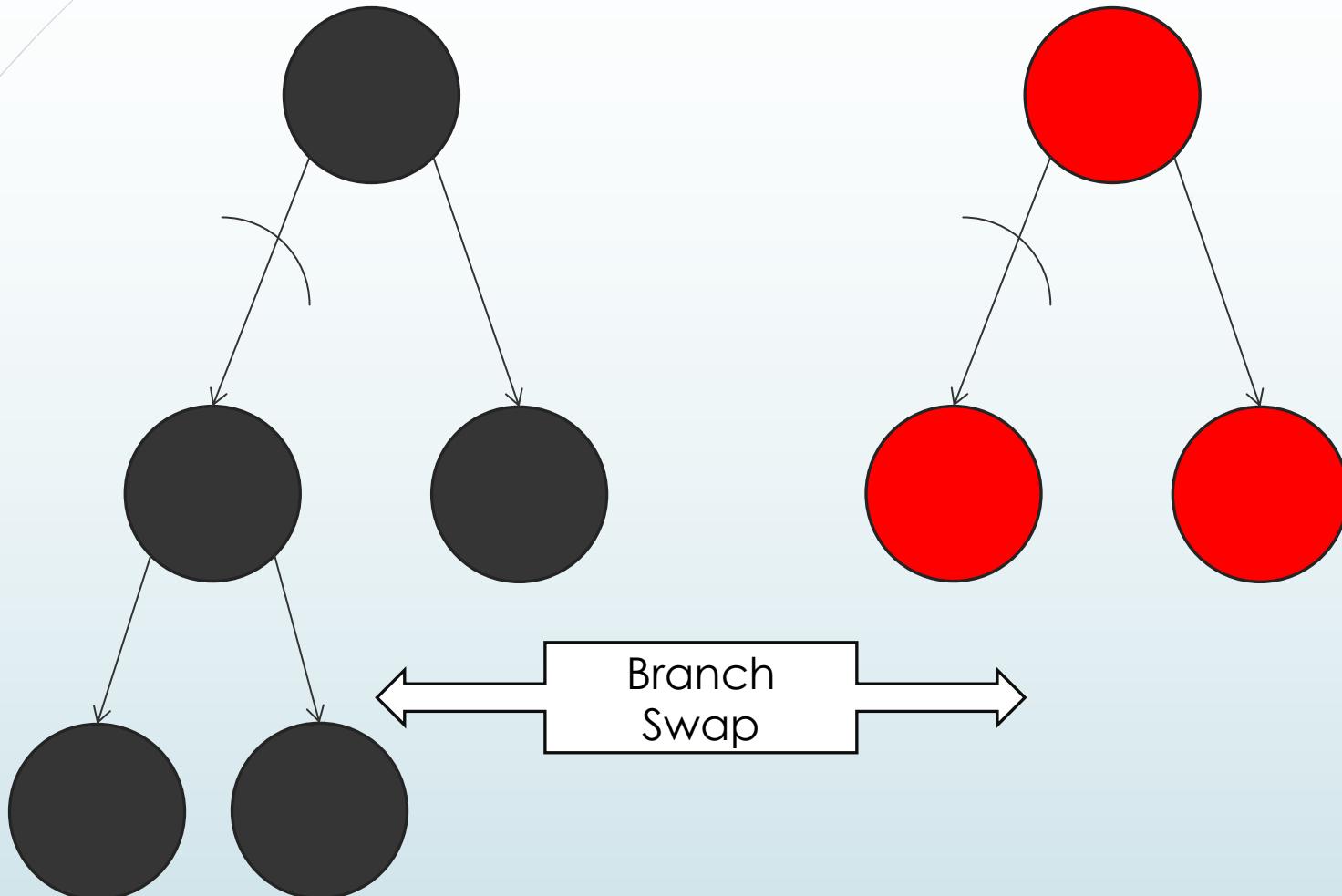
Genetic Programming (GP)

- ▶ Representation
 - ▶ Tree-based
- ▶ Selection
 - ▶ Fitness Proportional or Tournament
- ▶ Crossover
 - ▶ Yes. Branches of the Trees are Exchanged
- ▶ Mutation
 - ▶ Yes. Leaf value/Symbol Change or Operator Change
- ▶ Special Operations
 - ▶ Yes. Removal of Extra Symbols called bloat. Functions may be defined as shorter symbols (ADF - Automatically Defined Functions)

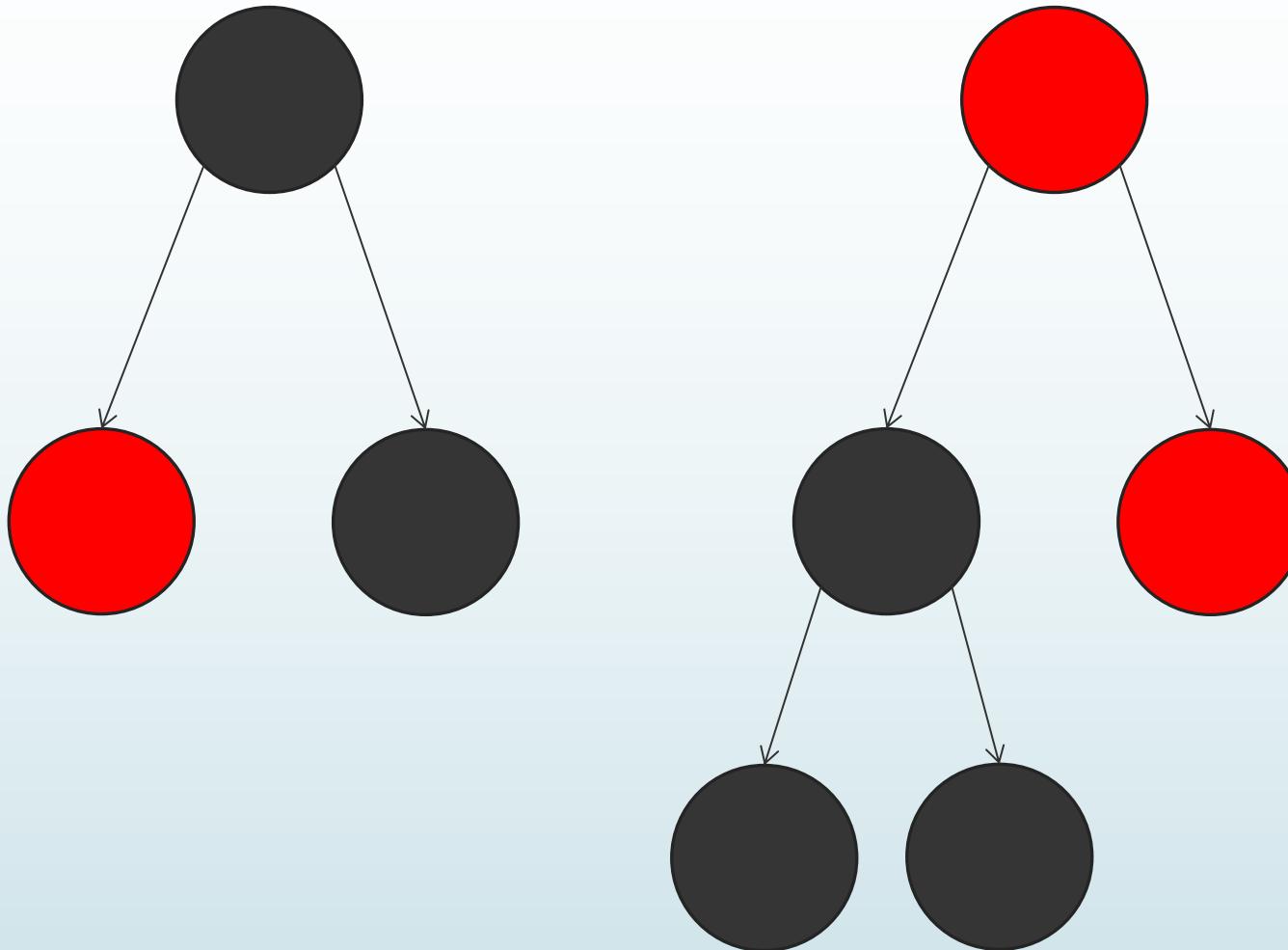
GP Parse Trees and LISP

- ▶ The idea comes from the programming language of LISP
 - ▶ $(function, arg1, arg2, \dots, argN)$
- ▶ Arguments are functions or terminals
- ▶ Terminals are literals (1, `x`) or variables (x, count)
- ▶ LISP allows for programs which manipulate code and run that code
- ▶ Other languages need to create a simulator
- ▶ Prefix notation e.g. $(+ 1 (* 7 x))$ is $1 + 7x$
- ▶ No need for order of operations – all operations are explicitly ordered by brackets

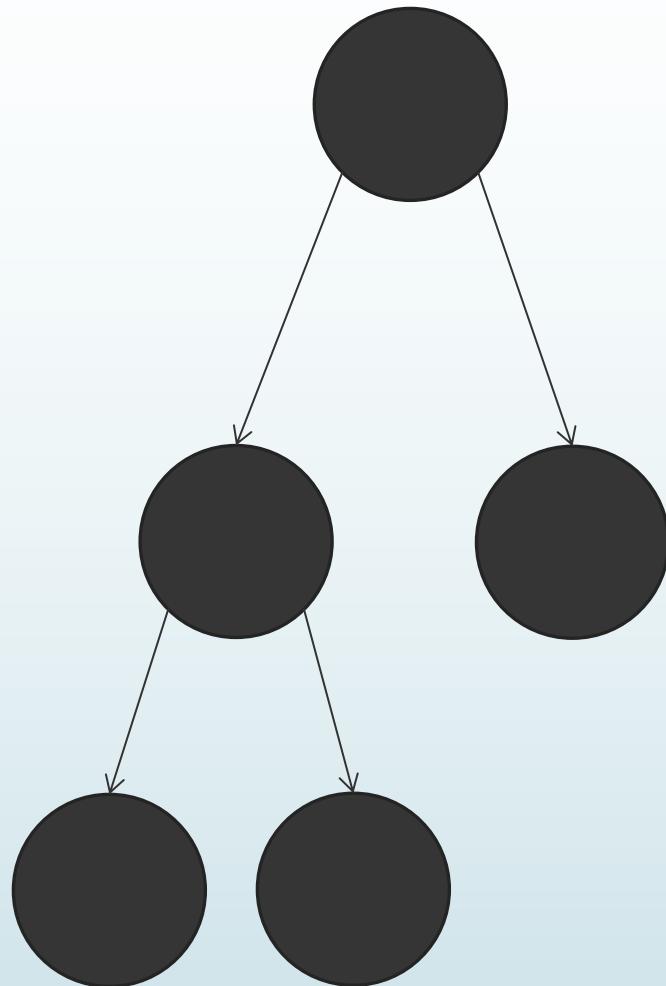
Crossover in a GP Tree



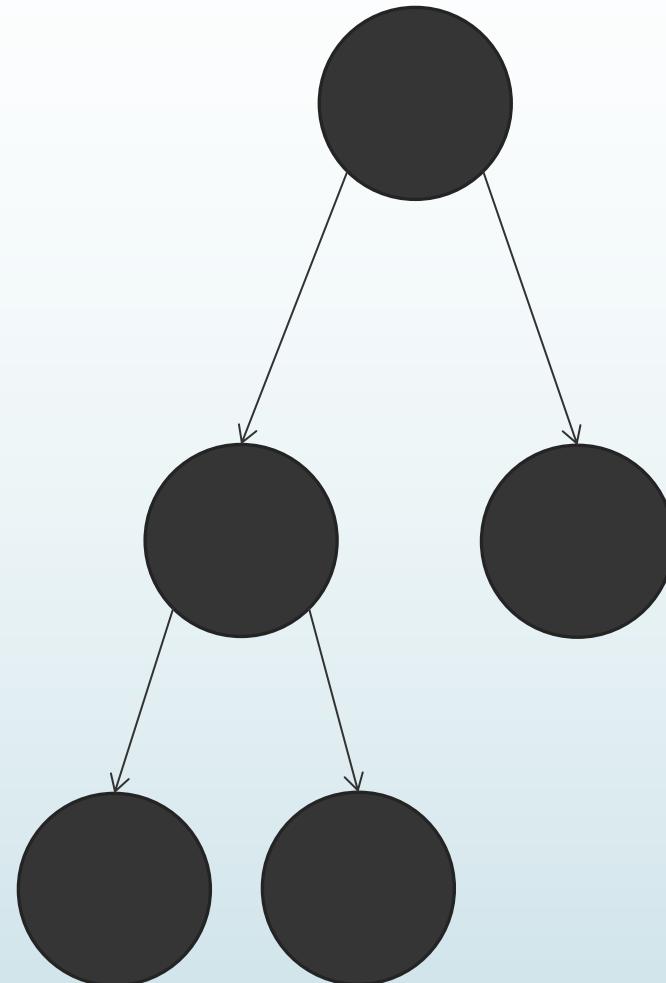
Crossover in a GP Tree



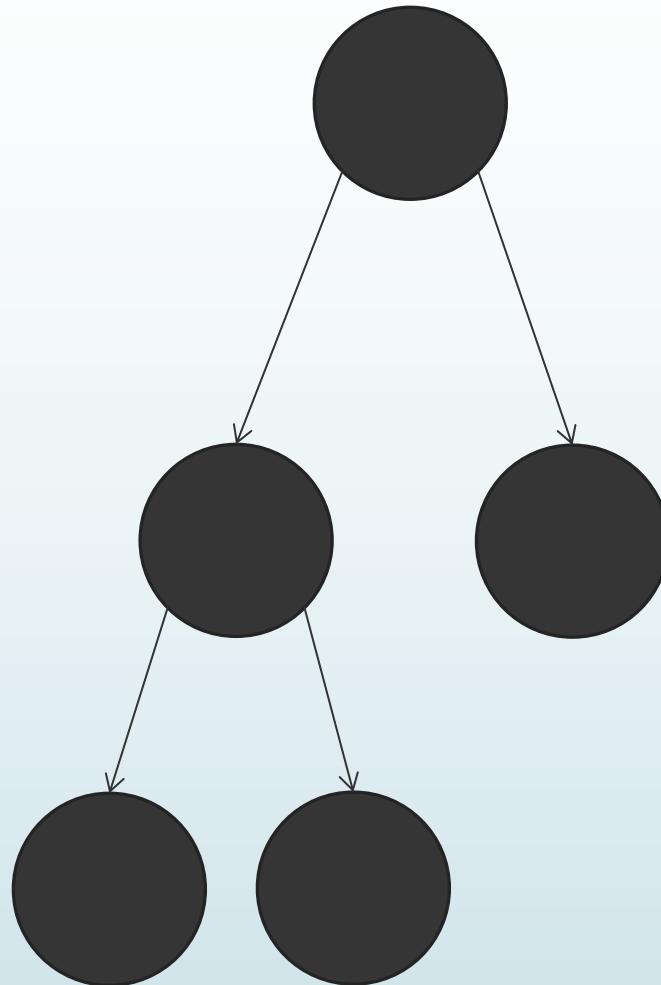
Mutation of a Terminal in a GP Tree



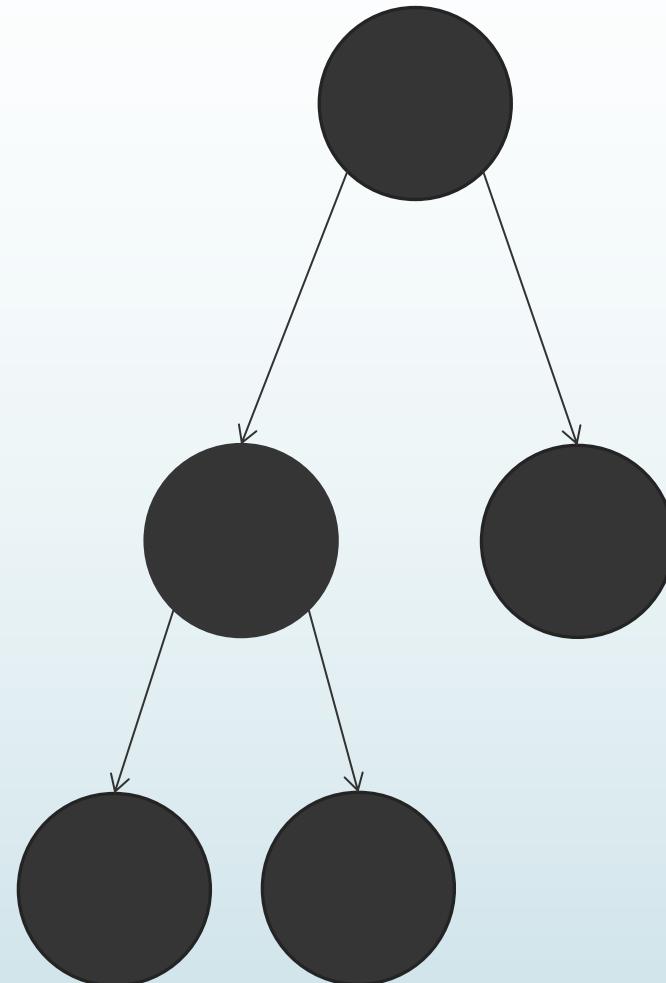
Mutation of a Operation in a GP Tree



Growing Operation in a GP Tree

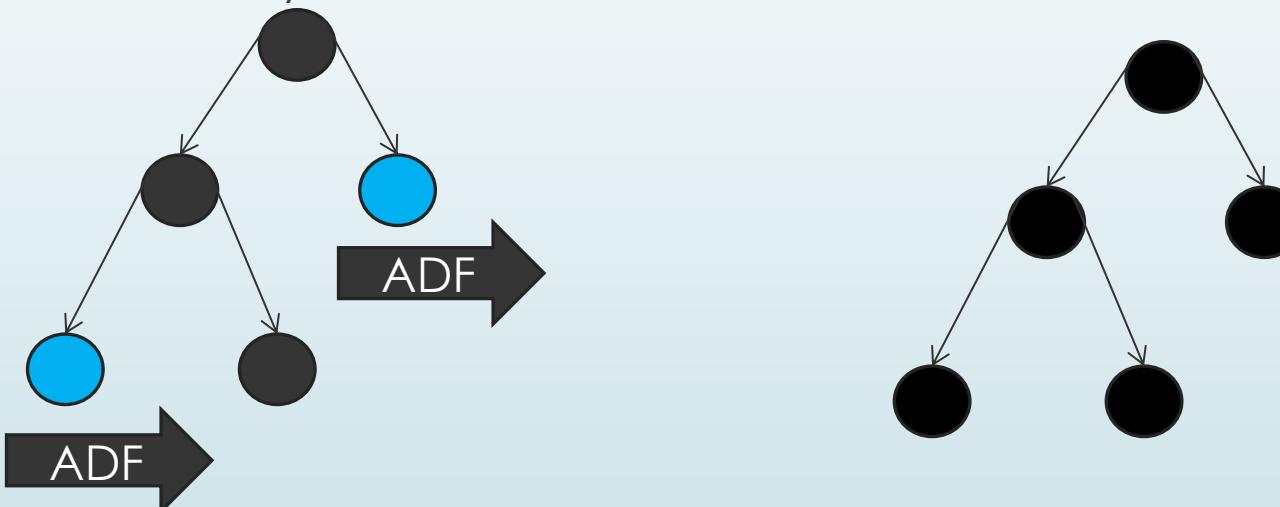


Cut Operation in a GP Tree



ADF Trees

- ▶ ADF – Automatically Defined Functions
- ▶ Many Times we have a tree computed again and again – repetition is costly
- ▶ Allow for the construction of GPs with smaller GP trees – construct a hierarchy



Rules on Functions in Trees

- ▶ All trees should produce ‘legal’ programs
- ▶ Operations which produce common errors – such as divide by zero – should have a protected version that explicitly maps those errors to a legal input value – such as 0

Bloat

- ▶ A number of operations provide no change in the result
 - ▶ Anything multiplied by 1
 - ▶ Anything added to 0
- ▶ A number of operations cancel out parts of the tree
 - ▶ Anything multiplied by 0
 - ▶ An operation followed by its inverse
- ▶ Leads to larger trees which are equally as fit as shorter ones

Why does Bloat exist?

- ▶ Imagine two trees which both add 5 to 6:
 1. 3 nodes
 2. 10 nodes which add a value multiplied by 0
- ▶ You require a minimum number of 3 nodes to implement (+ 5 6)
 - ▶ One for each of the arguments
 - ▶ One for the operand
- ▶ 7 nodes in the second tree are bloat
- ▶ What is the probability that a mutation operation (change operand/argument) will affect the solution to the problem?

Bloat Saves Solutions

- ▶ In the first tree the changing of an operation or argument will completely change the result, 100% of the time it will change the outcome
- ▶ In the bloated tree, 3 nodes are part of our solution, one to add, and two to multiply by 0. Changing these nodes will lead to a different answer
- ▶ Yet 4 nodes are inconsequential to the answer – 40% of the time there will be no change in fitness based on a mutation
- ▶ Heritability – A solution with more of these null mutations is likely to have its children survive as they have the same fitness

Bloat in Biology

- ▶ Repetition of genes
- ▶ Repetition of genes
- ▶ Duplication of genes
- ▶ Transposon Elements
- ▶ Repetition of genes
- ▶ Transposon Elements
- ▶ Not to be confused with redundant systems – Example Weight Loss Pill Trials

Fat Blocking Pill

- ▶ Idea – We want to create a diet pill
- ▶ Block the regulatory system in the human body which makes you gain weight
- ▶ Step 1 – find system
- ▶ Step 2 – create blocking drug
- ▶ Step 3 – clinical Trials on Mice



Mice Got Fatter

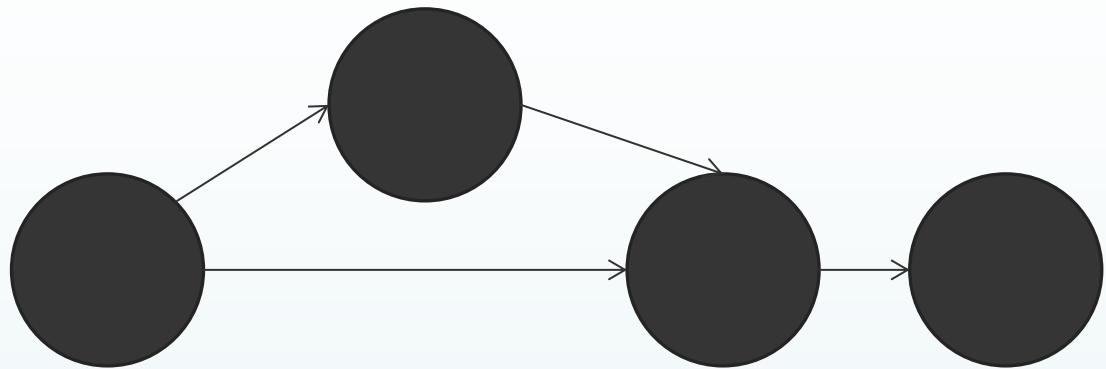
- ▶ The clinical trial showed the mice not only gained weight – they gained more weight than the control on the same diet!
- ▶ But we Blocked the Signals
- ▶ Ah – but did you block all the signals?
- ▶ Mammals have a secondary fat producing system which will come into effect when our primary system is compromised
- ▶ Issue – this secondary system is not as refined

Parsimony

- ▶ We like things simple in design of solutions
 - ▶ It seems that a perfect design is not one which one looks for things to add, but is one where there is nothing left to remove
- ▶ Let the trees grow but trim them at the result
- ▶ Penalize Larger Trees!
 - ▶ Reduction in fitness score
 - ▶ Less chance to Breed
- ▶ Find a method which does not use a tree-based model for the representation

Other Representations

- ▶ Directed Acyclic Graphs (DAG)
 - ▶ Cartesian Genetic Programming
 - ▶ Function Stacks
- ▶ Instead of Evolving Trees – Representation is graph
- ▶ Repeated input branches are passed down the DAG representation
 - ▶ Removes the need to recompute
 - ▶ Expansion and Bloat is limited – fixed size data structure
 - ▶ Operated upon as if it was a linear chromosome in a GA

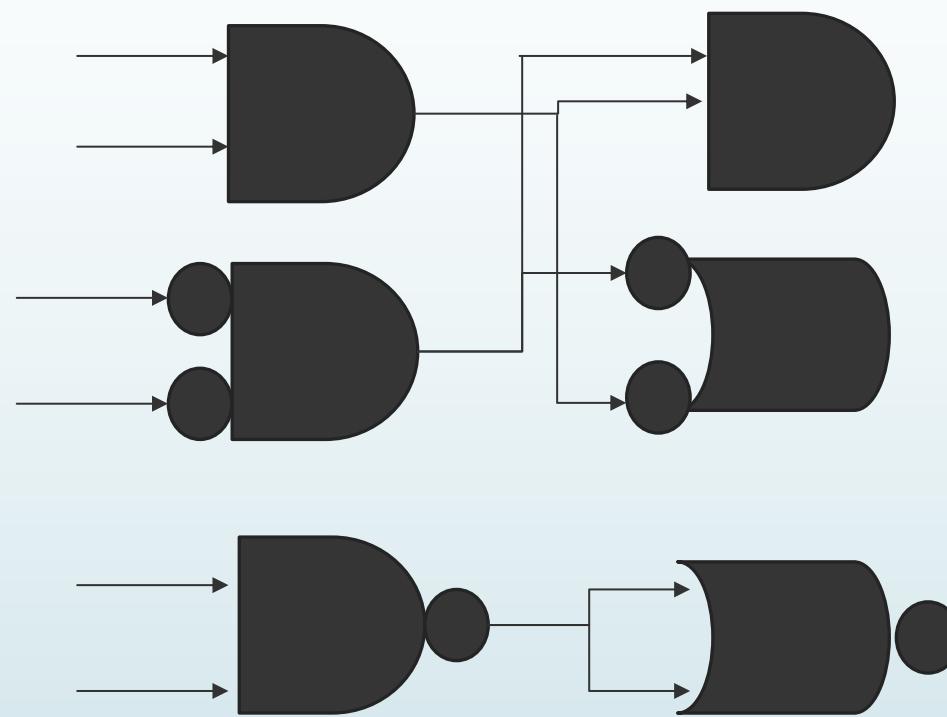


Cartesian Genetic Programming

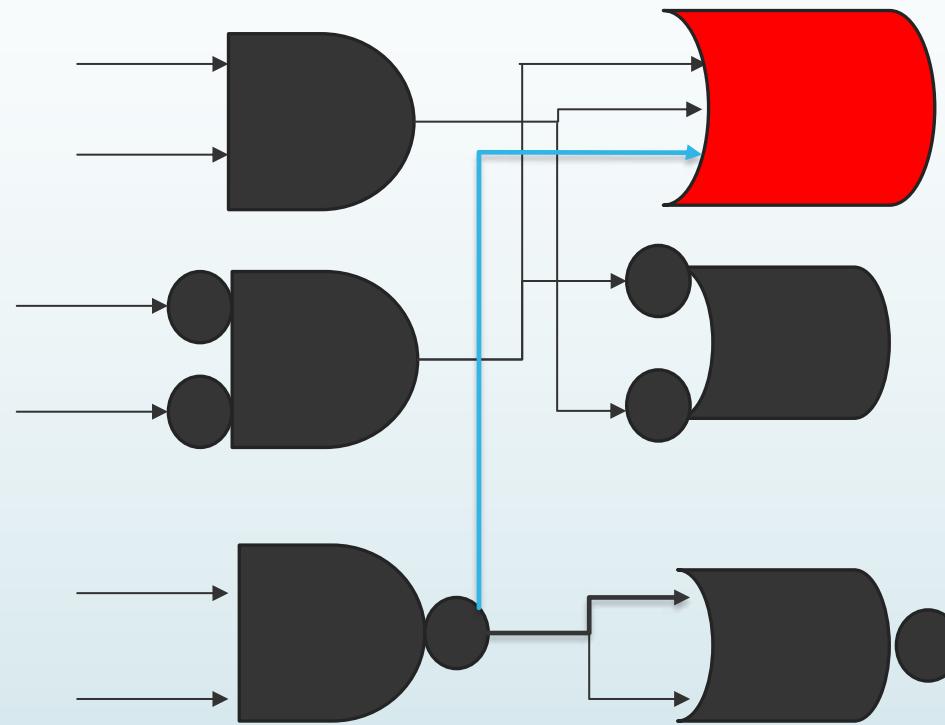
- NxM grid of Operations connected by wires
- Think Printed Circuit Boards



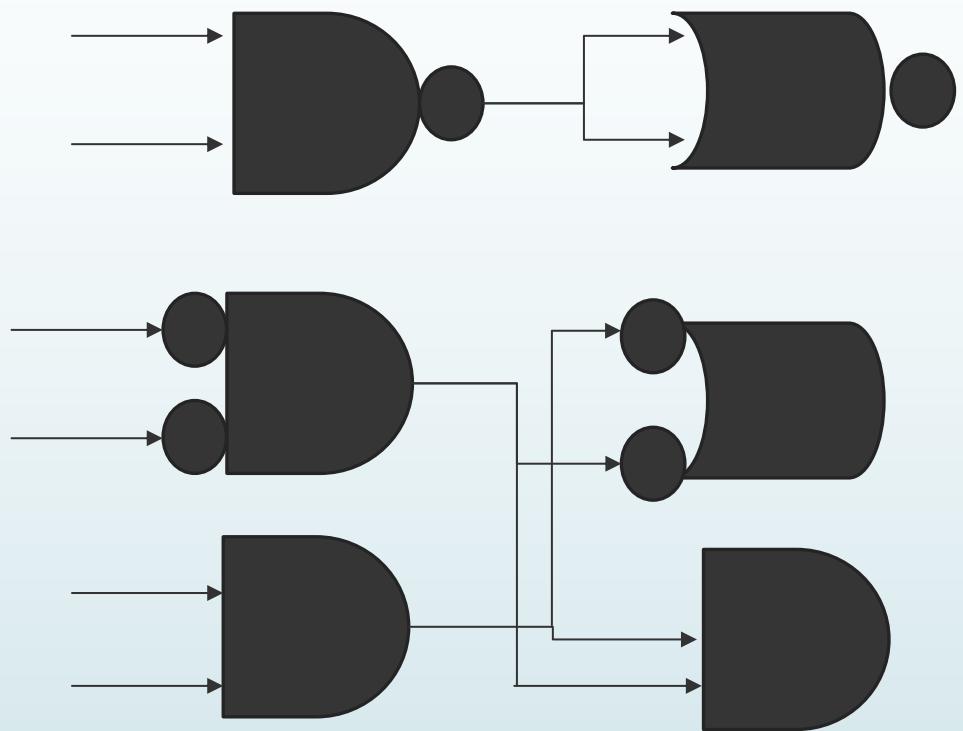
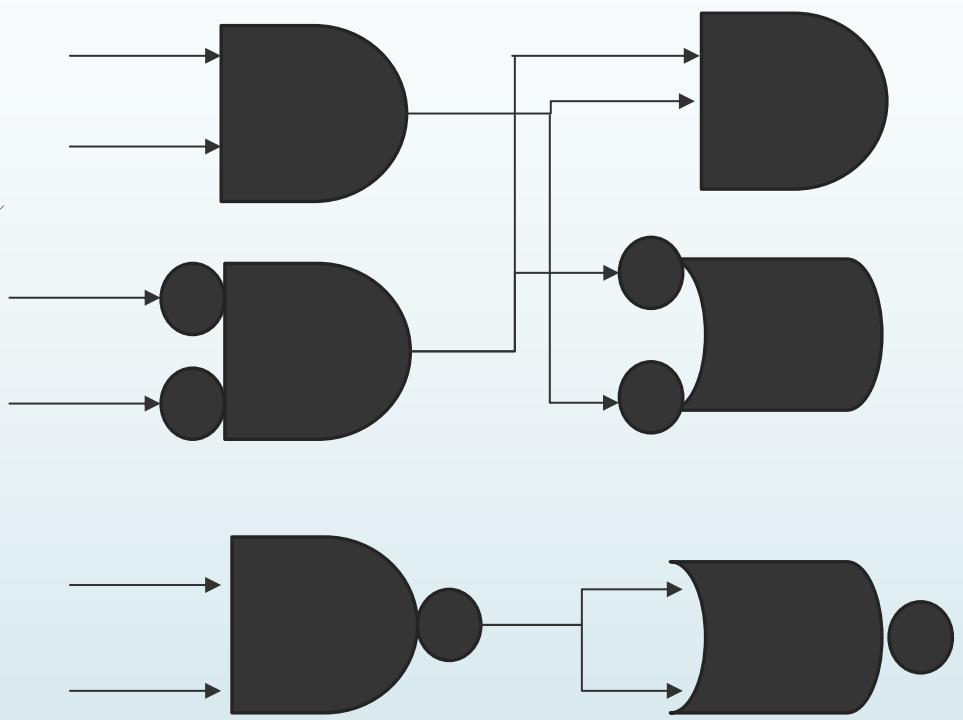
Data Structure



Mutations Can Affect Nodes and Edges



Flip Operations



Function Stack Representation

- ▶ Function Stacks have a linear chromosome consisting of nodes
- ▶ Node Contains
 - ▶ Function of 0..N inputs
 - ▶ Inputs – Either Previous Nodes in Order of the Chromosome or an input value
 - ▶ An Ephemeral Constant
- ▶ Crossover as per a linear string in a GA
- ▶ Mutations change the operation or constants

Evolutionary Programming

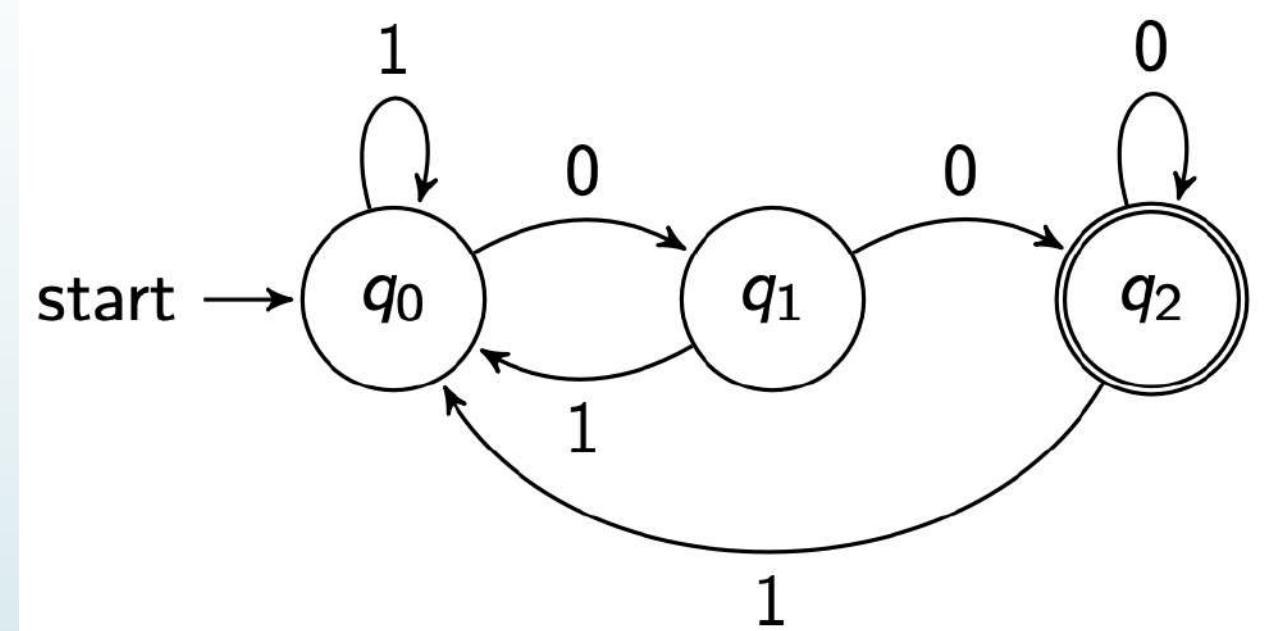
- ▶ Representation
 - ▶ Finite State Machine
- ▶ Selection
 - ▶ Replace with a member of a sample of mutants if better than parent
- ▶ Crossover
 - ▶ No*
- ▶ Mutation
 - ▶ Yes. Add or Remove a node, or Change transition, output, or starting node
- ▶ Note: Designed for use in an online setting for controller

Finite State Machine

- ▶ A deterministic finite state machine is defined by a tuple $\langle Q, \Sigma, O, \delta, \omega, q_0 \rangle$ $\langle Q, \Sigma, \delta, q_0, F \rangle$ where:
 - ▶ Q – finite set of states
 - ▶ Σ – a finite set of input symbols called the alphabet
 - ▶ δ – transition function $\delta: Q \times \Sigma \rightarrow Q$
 - ▶ q_0 – initial starting state where $q_0 \in Q$
 - ▶ F – a set of accepting states $F \subseteq Q$
- ▶ You can also define it via a state transition diagram

Representations of a FSM

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2^*	q_2	q_0



Mutations in EP FSM

- ▶ Mutations are insertions, deletions, changes to a transition, changes to an output, change starting node
 - ▶ Insertions – add a node and its connectors, find some set of random transitions to place into it (do not want it isolated)
 - ▶ Deletions – select a random node, all incoming transitions sent to other nodes at random
 - ▶ Change transition, change output, change initial state are self-explanatory

Evolutionary Strategies

- ▶ Representation
 - ▶ Vector of Real Values
- ▶ Selection
 - ▶ Replace with a member of a sample of mutations if better than parent
- ▶ Crossover
 - ▶ No*
- ▶ Mutation
 - ▶ Yes. Add small normally distributed parameter to a value

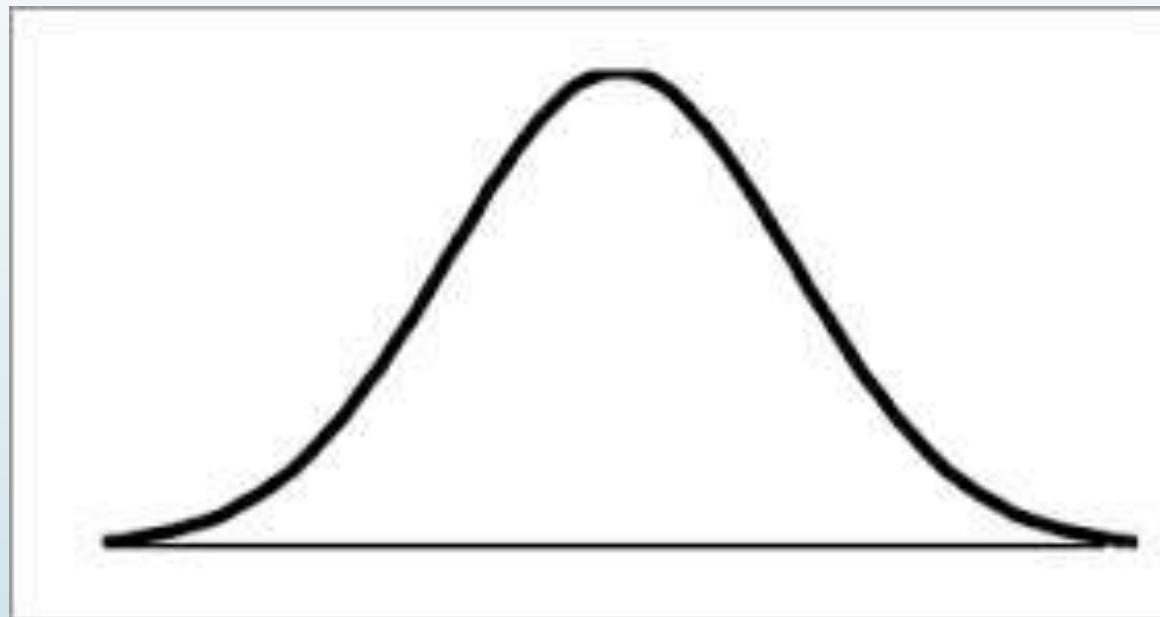
*Has been added in some variants

ES Bracketed Notation

- ▶ Normally Distributed Function of mutation is applied to the string of real numbers – some use log normal.
- ▶ Variants:
 1. $(1+1)$ -ES – a mutant is tested against its parent and the fittest is retained
 2. $(1+\lambda)$ -ES – λ mutants are tested against their parent with the fittest remaining, the parent retained if the best
 3. $(1,\lambda)$ -ES – λ mutants are tested against their parent, the parent is never retained, only one of mutants will continue on
 4. $(\mu/\rho+, \lambda)$ -ES – A population is used where a group of mutants is made for each and compete with the set of parents, this may also have a crossover operation

Small Mutations

- ▶ Pull from the Gaussian/Normal Distribution
- ▶ Many Mutations will make small changes in parameters, few will make large changes



Which Should I Use?

- ▶ No Free Lunches Here
- ▶ Note the similarities between Genetic Algorithms and Genetic Programming – Key Difference is the type of representation
- ▶ Similarly Evolutionary Programming and Evolutionary Strategies differ based on representation
- ▶ How you can represent your problem has a big effect on which of these methods are available (more on this next time in the representation lecture)
- ▶ Speed and cost
 - ▶ Speed of evaluation becomes a factor
 - ▶ Crossover is more expensive
 - ▶ Fitness evaluation is ALWAYS more expensive

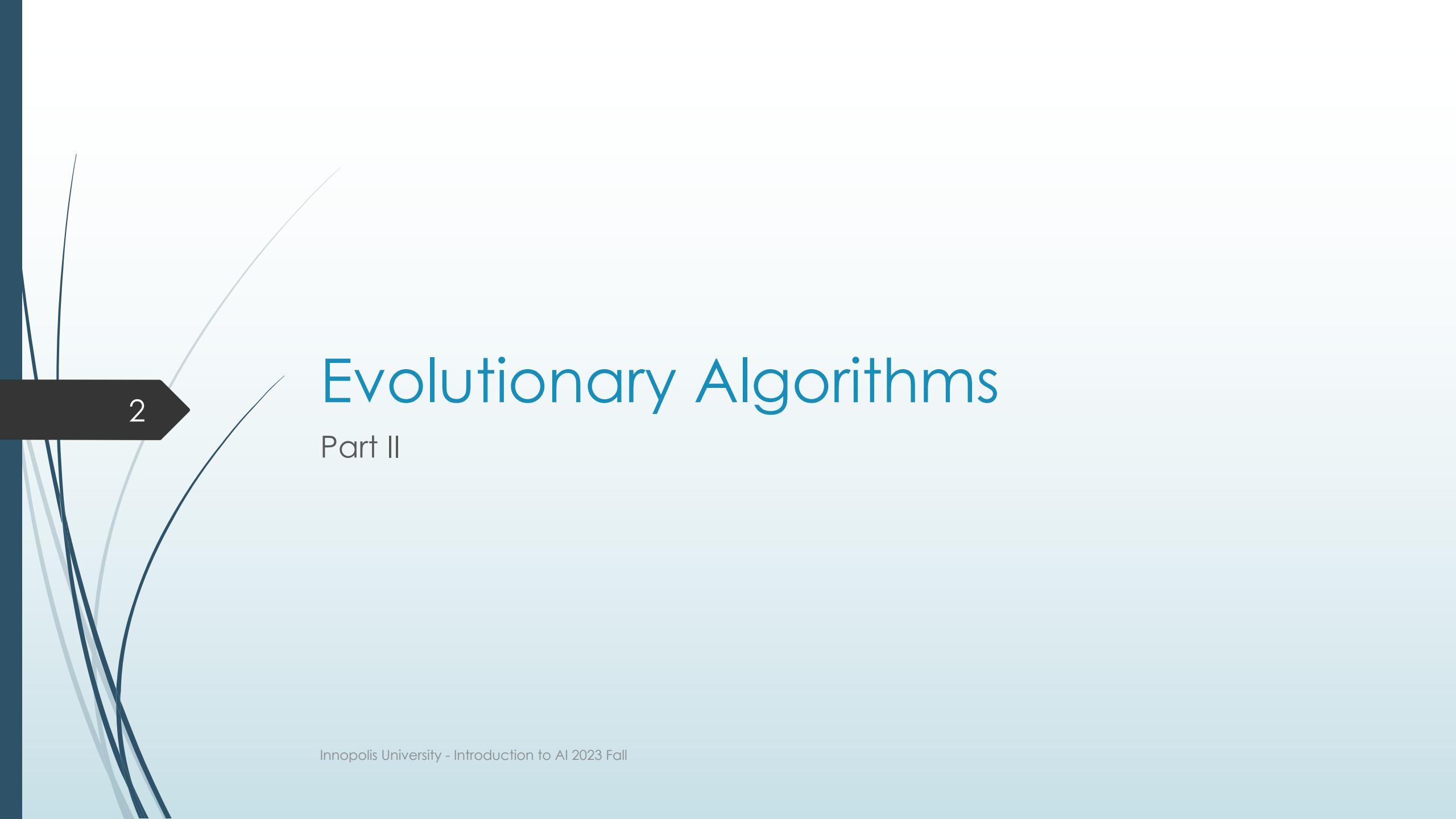
Summary

- ▶ Genetic Algorithms
- ▶ Genetic Programming
- ▶ Evolutionary Programming
- ▶ Evolutionary Strategies



Introduction to Artificial Intelligence

Week 11

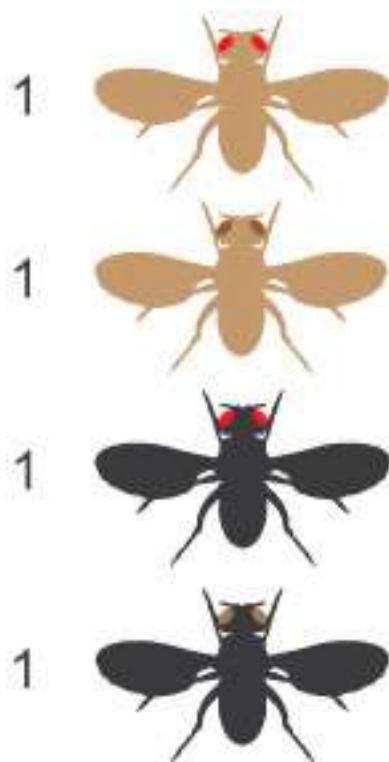
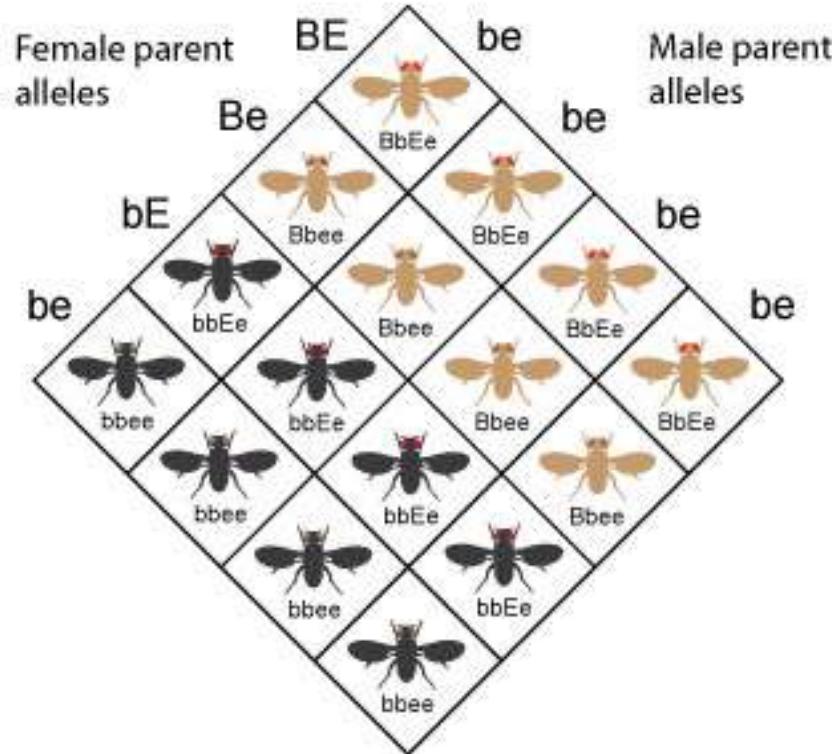


2

Evolutionary Algorithms

Part II

Direct v. Indirect Representations



- Genetics – Genotype v. Phenotype
- We may have a translation step between our internal representation in the system v. the traits perceived by someone
- The genotype of an organism is its complete set of genetic material
- Genotype contributes to phenotype, the observable traits and characteristics in an individual or organism

Direct Representation in Problems

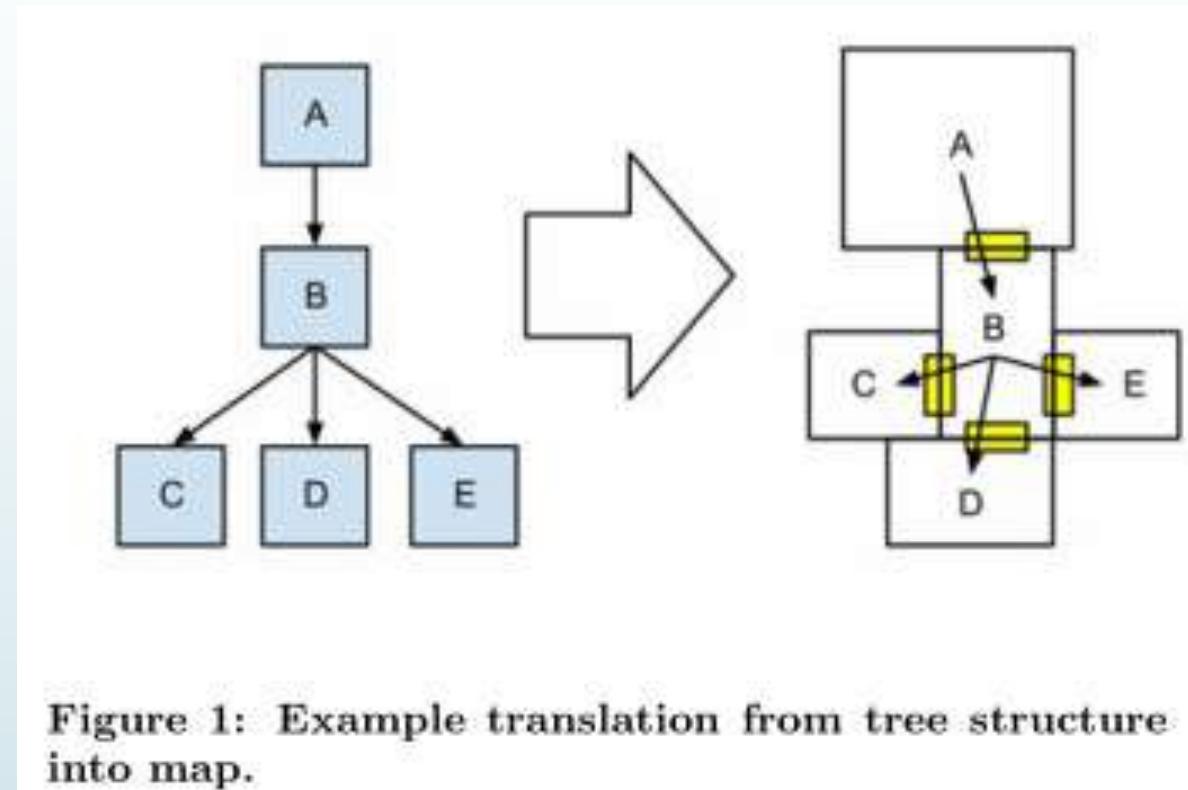
- ▶ Direct Representation
 - ▶ When the genotype and phenotype are the same
- ▶ Examples
 - ▶ Settings on a machine
 - ▶ Creation of a FSM or other controller
- ▶ Majority of optimizations have a direct search for the outcome

Indirect Representation in Problems

- ▶ Indirect Representation
 - ▶ When the evolved values allow for the construction of a solution to the problem
- ▶ Examples
 - ▶ Evolving a program for the solution
- ▶ This is an interesting emerging method for looking at evolution of structures which leads to solutions or for the entire system

Dungeon Levels

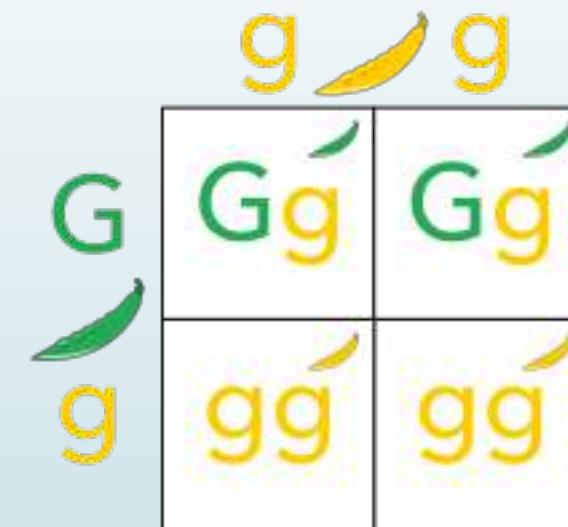
- ▶ Genotype – a tree of tuples
- ▶ Phenotype – a board with placement of rooms
- ▶ Translation step



Mendelian Genetics

- Discovered by **Gregor Johann Mendel** in 1865, but not popularized until Thomas Hunt Morgan in 1915
- Ratios of Pea Pods always in divisions of 1/4s
- Dominant and recessive traits
- Punnett Squares:

	R	R
r	Rr	Rr
r	Rr	Rr



What doesn't kill you makes for an interesting feature

- ▶ Sickle-cell diseases lead to a resistance to Malaria (Nature vol. 515 no. 7526)
- ▶ Серповидноклеточная анемия приводит к устойчивости к малярии

A moving target

Migration has led to an increase in the occurrence of sickle-cell disease in countries with previously low incidence of the disorder.

BY SIMON PLEASANTS

Sickle-cell disease is a heritable blood disorder that can have devastating effects. It is the result of a minor genetic change — a single error in a single gene — yet it is one of the most common genetic causes of sickness and death¹. It is also on the rise. According to a study published last year, approximately 305,800 children were born with sickle-cell disease in 2010, about two thirds of them in Africa². By 2050, the study authors predict, those numbers will increase by more than 25%, to more than 400,000 babies born annually with the disorder.

The disease-causing mutation occurs in a gene responsible for the production of haemoglobin, the protein in red blood cells that carries oxygen to tissues throughout the body. Individuals who inherit only one mutant gene typically experience no ill effects, but those who have two faulty copies develop the full-blown disease. The mutation itself is tiny; it replaces one amino acid (called glutamic acid) with another (valine) at one location in the haemoglobin protein. This small change has a major consequence — it turns the flexible, disc-shaped red blood cells rigid, giving them a distorted, sickle-like appearance.

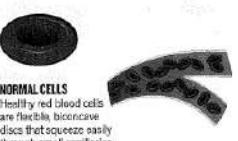
Sickled cells are not only less pliable than healthy red blood cells, they're also stickier. This causes them to adhere to each other and plug up narrow blood vessels, reducing blood flow and preventing adequate oxygen from being delivered to the tissues that need it. This in turn leads to many complications, some of which are life threatening. People with sickle-cell disease commonly experience episodes of acute pain, known as crises, that can last from a few hours to a few days. Other complications include anaemia, leg ulcers, jaundice, kidney damage, high blood pressure, gallstones, increased susceptibility to infection and stroke. These problems arise from a variety of complex mechanisms, such as altered pain receptors (see page S8) and the decreased lifespan of red blood cells³.

In high-income countries, improved management of the disease and medication, including strong painkillers, has meant that patients with sickle-cell disease can expect a better quality of life. In low-income nations, however, those with sickle cell are not as fortunate: in Africa, at least 50% of children born with the condition die before their fifth birthday⁴.

Sickle-cell disease is most prevalent in areas with hot climates — Africa, the Indian

subcontinent and the Caribbean are all affected. There is evidence to suggest that the geographic concentration of people with sickle-cell disease arose because people with sickle-cell trait — those who carry the sickle-cell gene but are generally healthy — have enhanced resistance to malaria⁵, resulting in preferential selection of the gene.

The malaria-causing parasite typically invades red blood cells and multiplies inside them. But although sickle-cell carriers can be infected, the presence of sickle haemoglobin confers resistance to malaria. There are a number of different theories, one of which is that subtle biological changes in the blood — pos-



NORMAL CELLS

Healthy red blood cells are flexible biconcave discs that squeeze easily through small capillaries. They contain multiple haemoglobin molecules that carry oxygen from the lungs to tissues throughout the body. Red blood cells are made in bone marrow and have a lifespan of 120 days.



SICKLE CELLS

Sickle cells contain a defective form of haemoglobin that, when not bound to oxygen, causes them to bend into a crescent shape. The sticky, rigid cells block small capillaries, which results in pain and other serious complications. Sickled cells have a lifespan of 10–20 days.

sibly higher carbon monoxide levels⁶ or the interaction between oxygen and sickle haemoglobin⁷ — makes people with the sickle-cell trait less of a hospitable host to the parasite and usually prevents the disease from taking hold.

About 70% of global cases of sickle-cell disease are in Africa. The World Health Organization estimates that in some African countries, including the Democratic Republic of Congo, Ghana and Nigeria, prevalence of sickle-cell trait is between 10% and 30%, resulting in an estimated disease prevalence of at least 2% in most

of Africa — a crippling health burden in a region that is ill-equipped to bear it, and one that lacks the financial and health-care resources present in wealthy countries.

ON THE MOVE

Increased population movement associated with globalization is bringing sickle cell to regions that previously had negligibly low incidences of the disease. Migration of people with the sickle-cell trait or sickle-cell disease caused the faulty gene to flow from high-frequency sickle-cell gene areas, such as Africa and India, to western Europe, the eastern coast of South America, and North America⁸. These migrations have increased disease prevalence in areas not previously associated with the disorder.

The Sickle Cell Disease Association of America estimates that there are about 100,000 people living with the disorder in the country. The US Department of Health and Human Service's National Heart, Lung and Blood Institute suggests that sickle-cell disease affects about 1 in 500 African Americans, a number about ten times greater than the general population.

Ironically, improved medical treatment leading to longer life expectancy is contributing to the increased prevalence of sickle-cell disease. Penicillin, taken daily, helps to prevent and treat infection in young children; folic acid stimulates the production of red blood cells; the drug hydroxyurea (known as hydroxyurea in the United States) reduces the number and severity of painful crises and blood transfusions help to temper acute cases of the disease. With so many interventions, more people will survive to pass on sickle-cell genes to their children. ■

Simon Pleasants is a copy editor at *Nature*.

- Makani, A. et al. *Sci. World J.* <http://dx.doi.org/10.1155/2013/93252> (2013).
- Piel, F. B. et al. *PLoS Med.* <http://dx.doi.org/10.1371/journal.pmed.1001484> (2013).
- Grossi, S. D. et al. *Am. J. Hum. Genet.* 91, 539–540 (2011).
- Allison, A. C. *Br. Med. J.* 1, 290–294 (1954).
- Ferreira, A. et al. *Cell* 145, 398–409 (2011).
- Cyrklaft, M. et al. *Science* 334, 1293–1296 (2011).
- Piel, F. B. et al. *Lancet* 381, 142–151 (2013).

CARRIED IN THE BLOOD

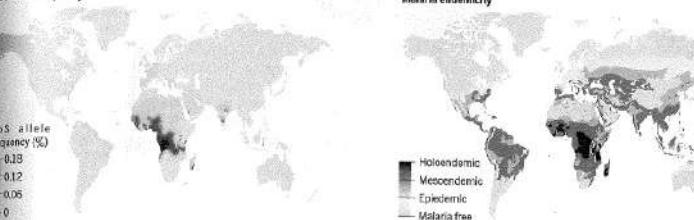
Sickle-cell disease is often fatal in low-income nations. But researchers are beginning to understand more about the disease and how to treat it. By Simon Pleasants.

FACTS LINK

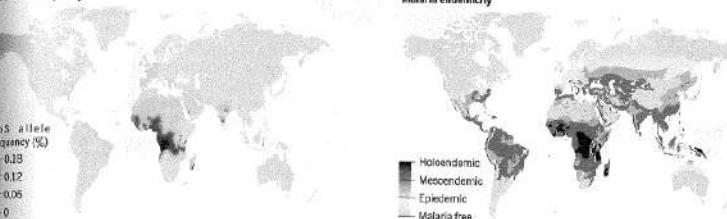
It is coincidence that sickle-cell disease and malaria share similar global distributions. The gene responsible for sickle haemoglobin confers increased resistance to the malaria parasite when inherited from only one parent. Such a trait allowed carriers of the gene (*HbS*) distribution, on the map below (red), to survive and reproduce in areas where malaria is present (green).

Although malaria is indigenous in many areas, the sickle haemoglobin gene has a more narrow distribution than malaria because the mutation arose in Africa. Limited migration from Africa to certain regions (for example, China) has restricted its spread.

HbS allele frequency

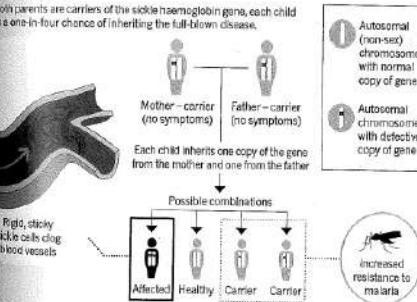


Malaria endemicity



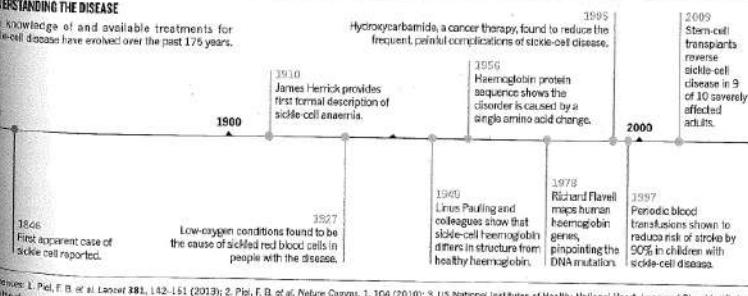
GENETIC ROULETTE

If both parents are carriers of the sickle haemoglobin gene, each child has a one-in-four chance of inheriting the full-blown disease.



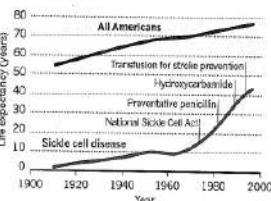
UNDERSTANDING THE DISEASE

The knowledge of available treatments for sickle-cell disease have evolved over the past 175 years.



HEALTH IMPROVEMENT

As a result of more effective treatments, the life expectancy of people with sickle-cell disease in the United States has increased dramatically. Most developing nations, however, have yet to see such a change.



References: 1. Piel, F. B. et al. *Lancet* 381, 142–151 (2013); 2. Piel, F. B. et al. *Nature Commun.* 1, 104 (2010); 3. US National Institutes of Health: National Heart, Lung and Blood Institute, <http://www.ncbi.nlm.nih.gov/4sugv>.

EA Representation

- ▶ A data structure and its defined variation operators
- ▶ Choice of representation defines the search space
 - ▶ Defines the possible solutions
 - ▶ Data structure choice
 - ▶ Defines the connection
 - ▶ Variation Operation
- ▶ How we represent a problem is a key to a good EA
- ▶ Good representations allow smaller search spaces and faster searches through the space

Self-avoiding Walks

- Take an $N \times N$ grid, a Self-avoiding walk of size $N-1$ is a walk with $N \times N$ steps visiting every square once

V	>	V	>	V	X
V	^	V	^	V	^
V	^	V	^	V	^
V	^	V	^	V	^
>	^	>	^	>	^

How Can We Represent an Agent's Movement?

- ▶ Directionally – we could put each of the movement directions into a string
 - ▶ *UP, DOWN, LEFT, RIGHT*
- ▶ The previous example would be:
 - ▶ Down Down Down Down Down Right Up Up Up Up Right Down Down Down
Down Down Right Up Up Up Up Right Down Down Down Down Right Up
Up Up Up Up
 - ▶ OR
 - ▶ *Down - X5, Right, Up - X5, Right, Down - X5, Right, Up - X5, Right, Down - X5, Right, Up - X5*
- ▶ The size of the search space is 4 directions raised to N-1 steps

Can We Do Better?

D	R	D	R	D	X
D	U	D	U	D	U
D	U	D	U	D	U
D	U	D	U	D	U
D	U	D	U	D	U
R	U	R	U	R	U

Agent with Direction Saved

- ▶ Lets have an agent which saves its current direction it faces in
- ▶ Now movements are:
 1. Forward (*F*)
 2. Turn Left + Forward (*L*)
 3. Turn Right + Forward (*R*)
- ▶ The solution string from before is:
 - ▶ *FFFFFFRRFFFFRRFFFFRRFFFFRRFFFFRRFFFF*
- ▶ Only 3 symbols raised to the $N-1$
- ▶ Large reduction in the search space

What Do We Lose?

F	R	R	R	R	X
F	F	F	F	F	F
F	F	F	F	F	F
F	F	F	F	F	F
F	F	F	F	F	F
R	R	R	R	R	R

Search Space has been Reduced

- ▶ All legal walks still retained in the Saved Direction
- ▶ Walks where we backtrack are removed – remember this is not a movement we would be interested in
- ▶ The only walks we have reduced from the solution space where those who would never provide a correct solution – reduction of the space without consequence

Epistasis

- ▶ **Epistatic gene** in genetics: a gene that determines whether or not a trait will be expressed
- ▶ **Epistatic gene** in EAs: the actions made in earlier parts of the chromosome having larger impacts on the fitness value of the chromosome
- ▶ **Epistatic gene** in Self-avoiding Walk: actions made early in the string will have large impacts on the fitness
- ▶ For example, changing the first move to a R will run the walk off the stage, giving it 0 fitness rather than 35, whereas changing the last symbol to a R will only reduce the fitness by 1. Assume that each correct movement gives 1 points
- ▶ $FFFFFRRFFFFRRFFFFRRFFFFRRFFFFR R FFFF \rightarrow f(x) = 35$
- ▶ $(F \rightarrow R) FFFFRRFFFFRRFFFFRRFFFFRRFFFFR R FFFF \rightarrow f(x) = 0$
- ▶ $FFFFFRRFFFFRRFFFFRRFFFFRRFFFFR R FFFF (F \rightarrow R) \rightarrow f(x) = 34$

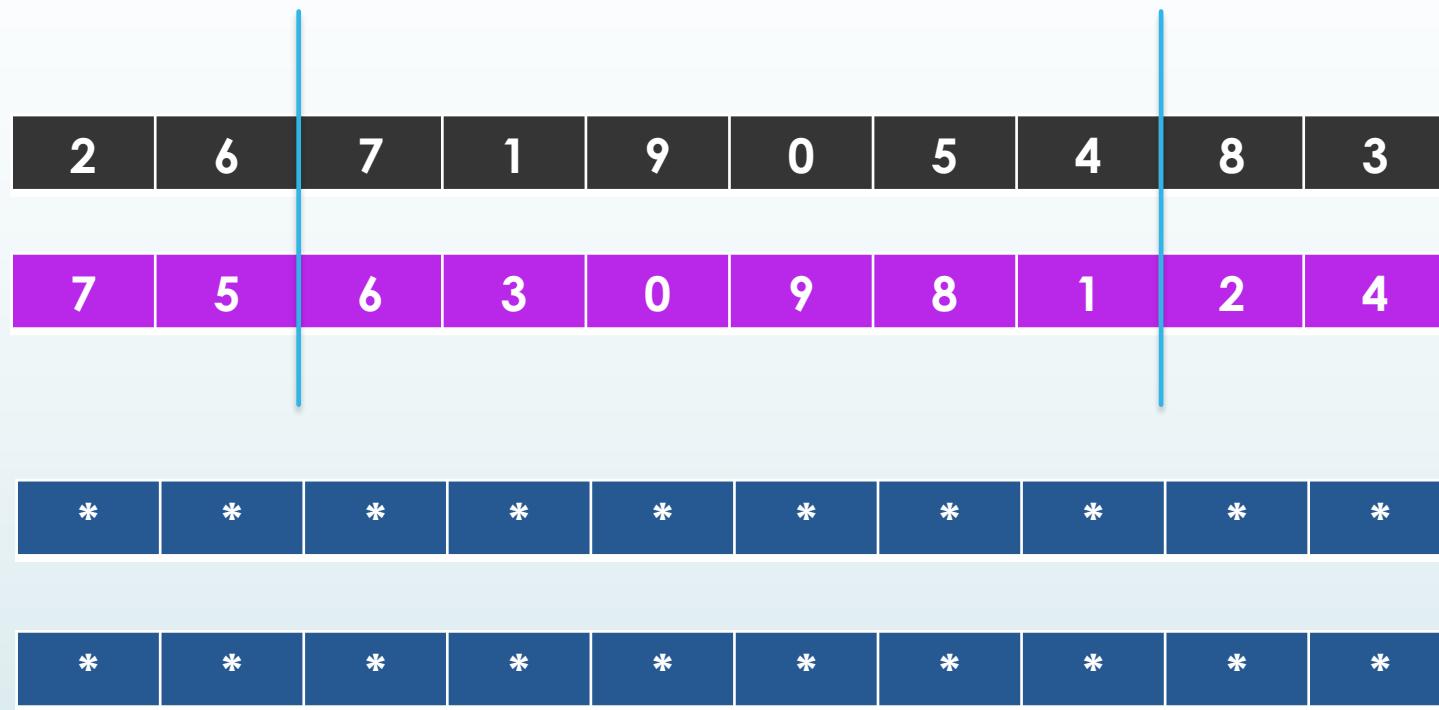
TSP-Ordered Gene

- ▶ GAs representation is the tour of cities
- ▶ Natural idea of ordering in gene

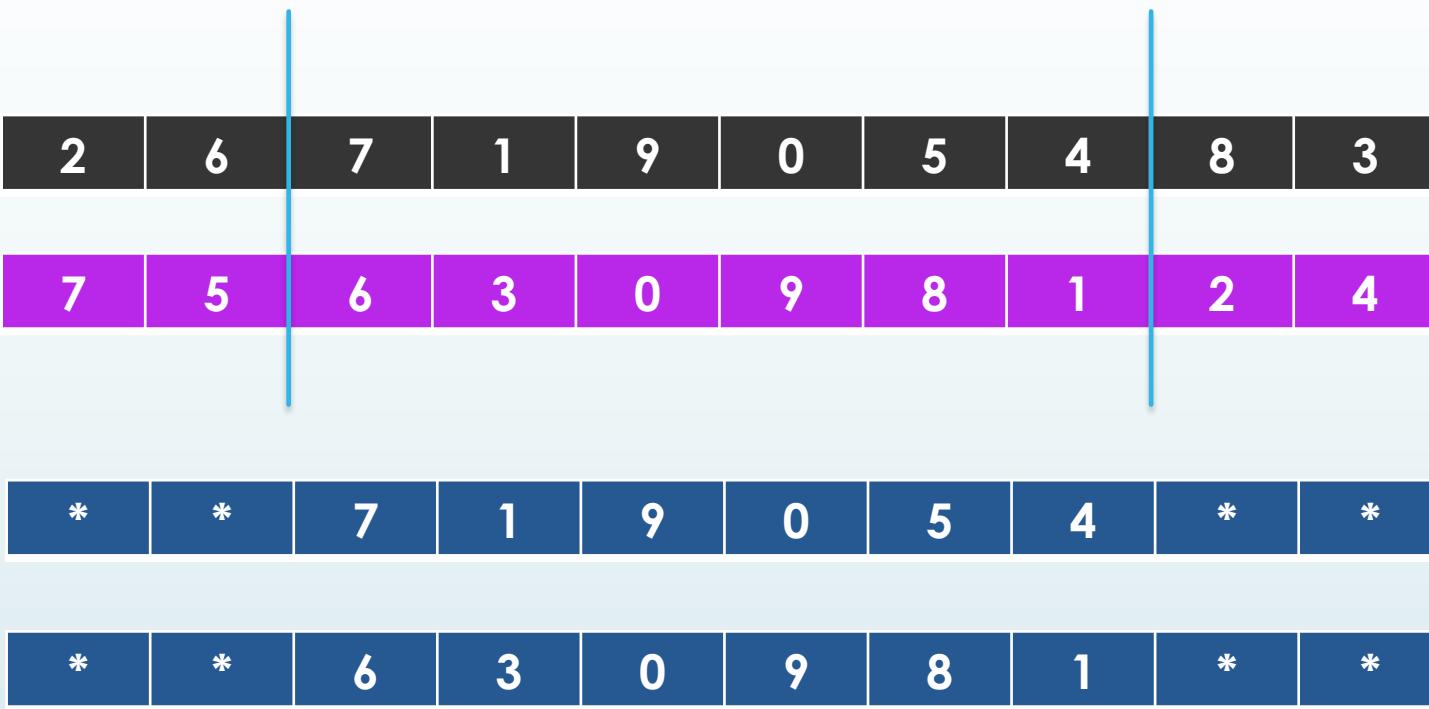
Two Point Crossover



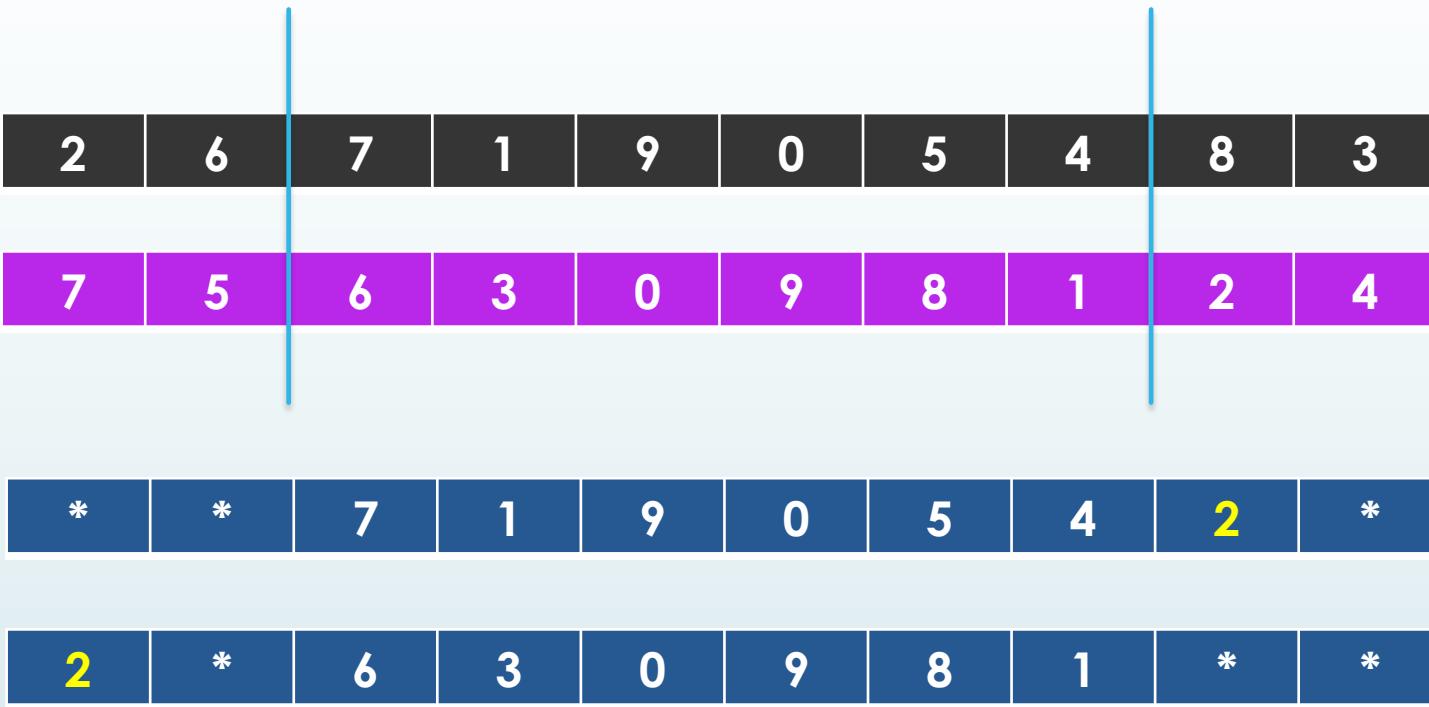
Partial Mapped Crossover



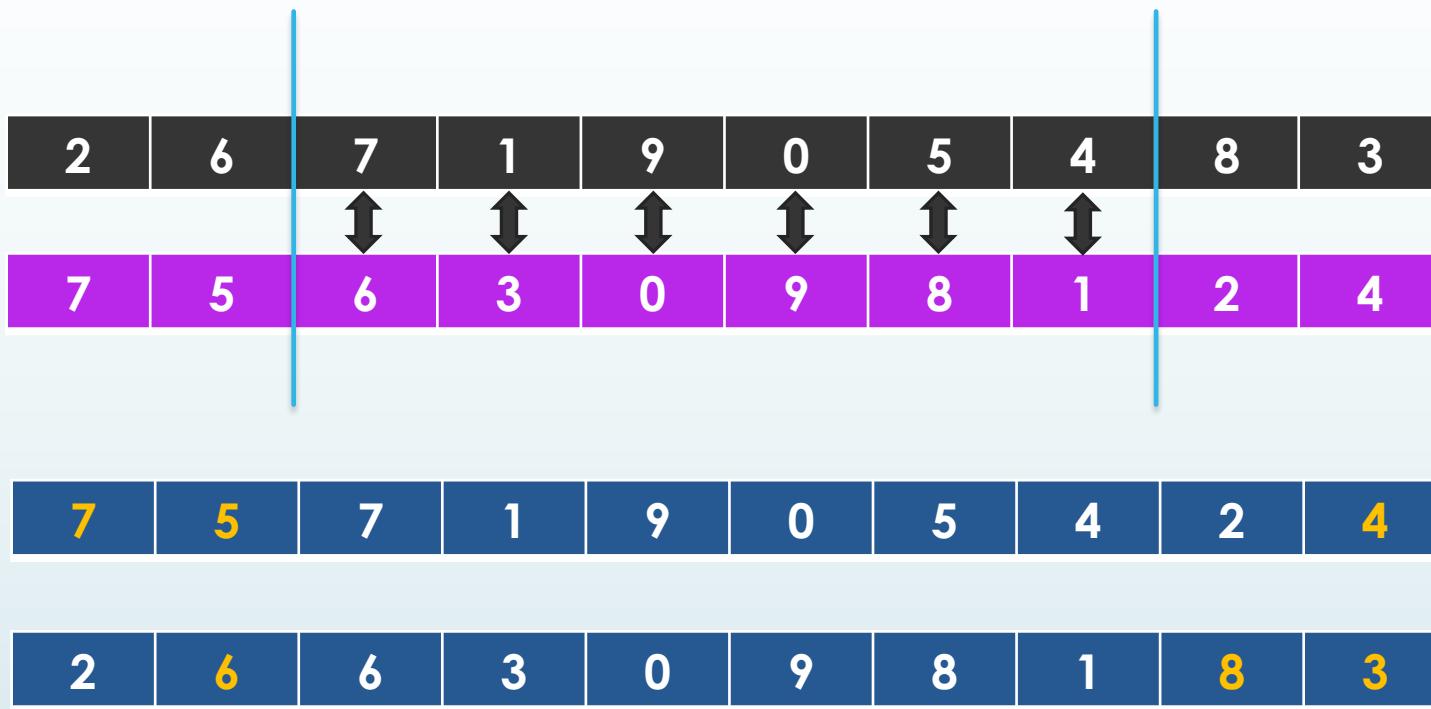
Partial Mapped Crossover



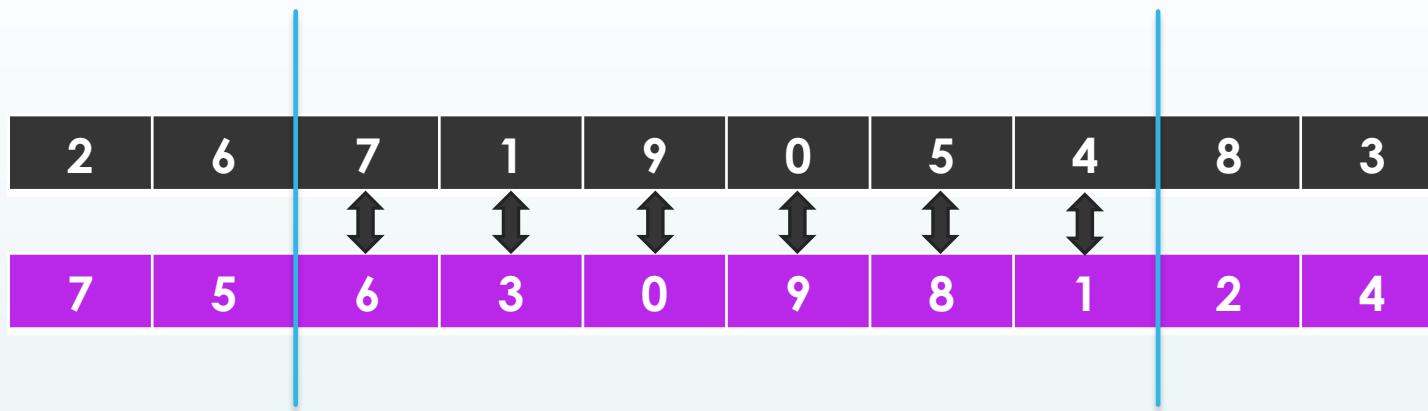
Partial Mapped Crossover



Partial Order Crossover



Partial Order Crossover



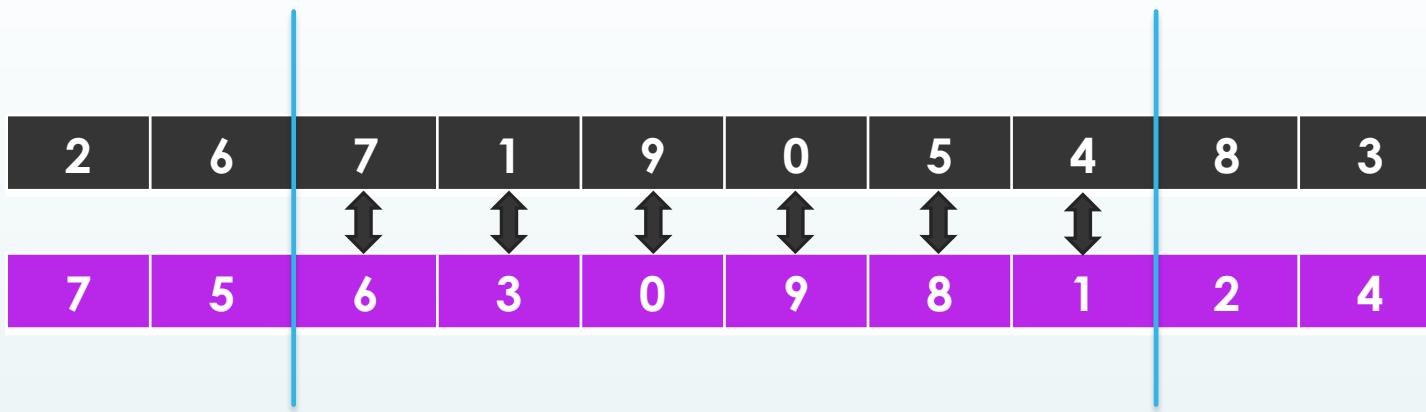
Two parent strings

7→6	5→8	7	1	9	0	5	4	2	4→1
-----	-----	---	---	---	---	---	---	---	-----

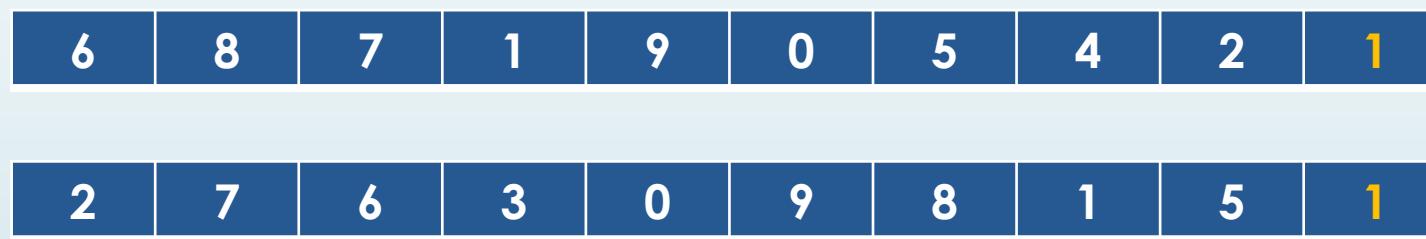
Two children strings

2	6→7	6	3	0	9	8	1	8→5	3→1
---	-----	---	---	---	---	---	---	-----	-----

Partial Order Crossover

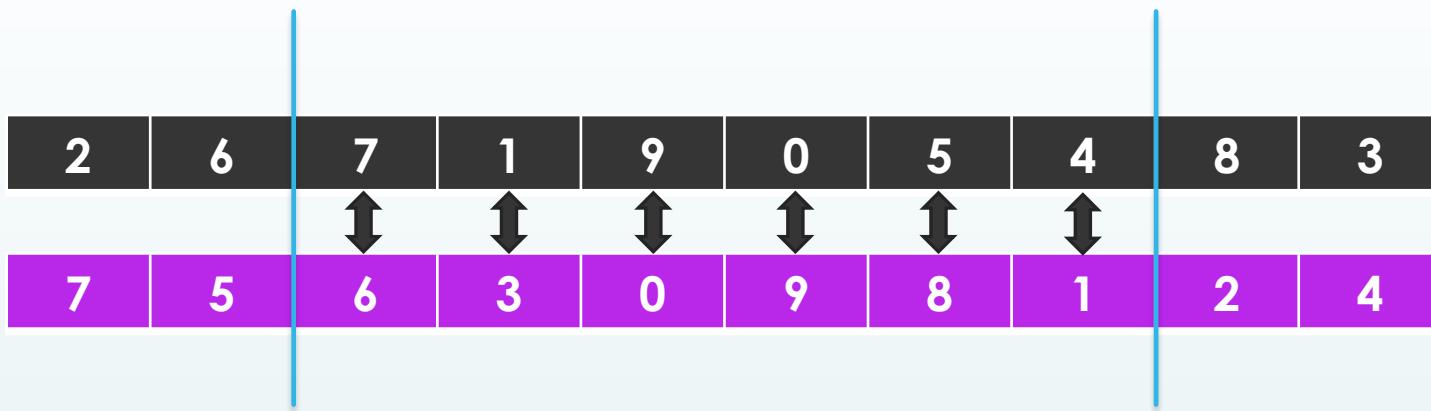


Two parent strings



Two children strings

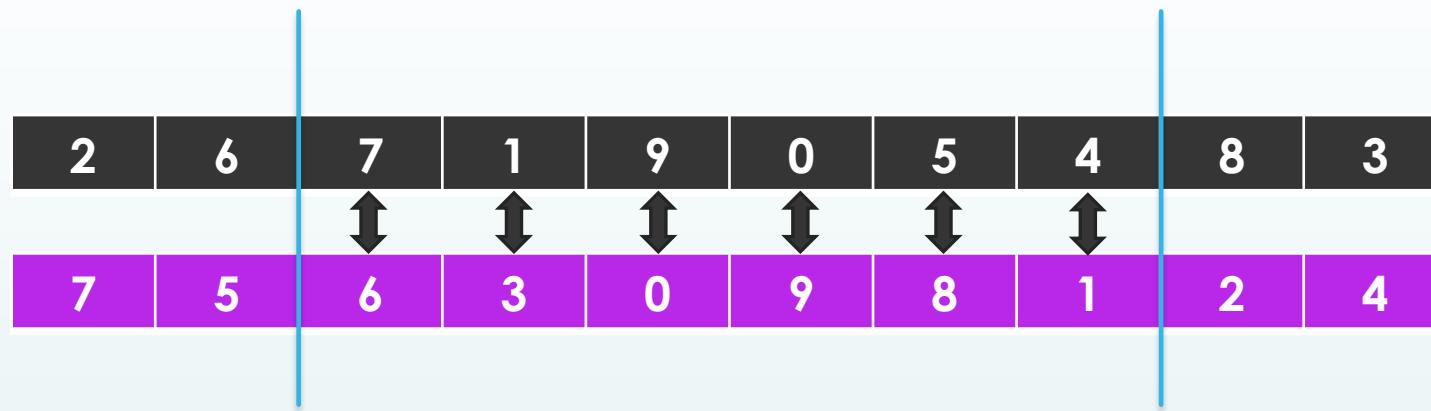
Partial Order Crossover



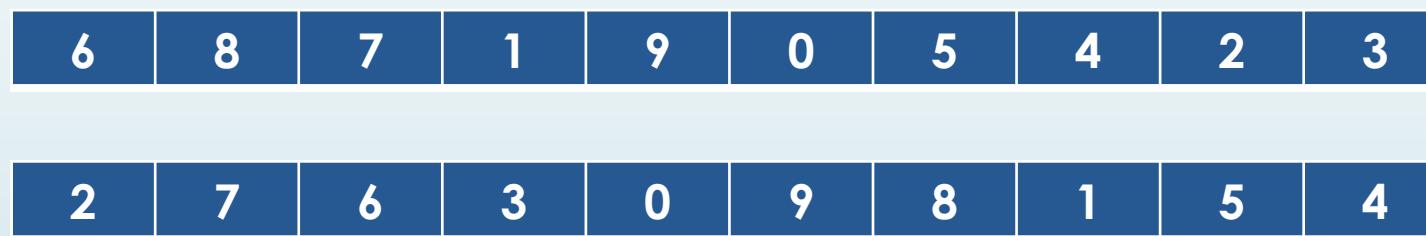
Two parent strings

Two children strings

Partial Order Crossover



Two parent strings



Two children strings

Indirect Real Numbered Representation

- ▶ Move from direct to indirect representation
- ▶ Real-valued string
- ▶ Crossover as per one and two point
- ▶ Mutation is now as per ES – add a random number from a distribution
- ▶ Mapping from
- ▶ Example

0	1	2	3	4	5	6	7	8	9
34.3	53.1	22.1	65.5	34.5	12.4	0.7	66.3	96.2	0.12

- ▶ 9, 6, 5, 2, 0, 4, 1, 3, 7, 8.

Indirect Real Numbered Representation

- ▶ Direct to Indirect method
- ▶ No longer need to worry about ordering
 - ▶ Ordering is now taken care of in the list
 - ▶ Sorting algorithms are fast
- ▶ Evolutionary Changes are less likely to numbers which are closer together
 - ▶ Evolution also has a direct control on the heritability

Selection of Representation

- ▶ Has a large impact on the outcome of the evolution
- ▶ Defines how your space will look – the search space in terms of the connectivity on the space
- ▶ Genetic operations are defined by how they act upon the representation of the problem

Iterated Prisoner's Dilemma (ITP)

- ▶ Two player game
- ▶ Story of two convicts who are placed in interrogation and played against each other
- ▶ Exploitable player allows a score >3 against it

	Player 1	
Player 2	C	D
	C	(3,3) (5,0)
D	(0,5)	(1,1)

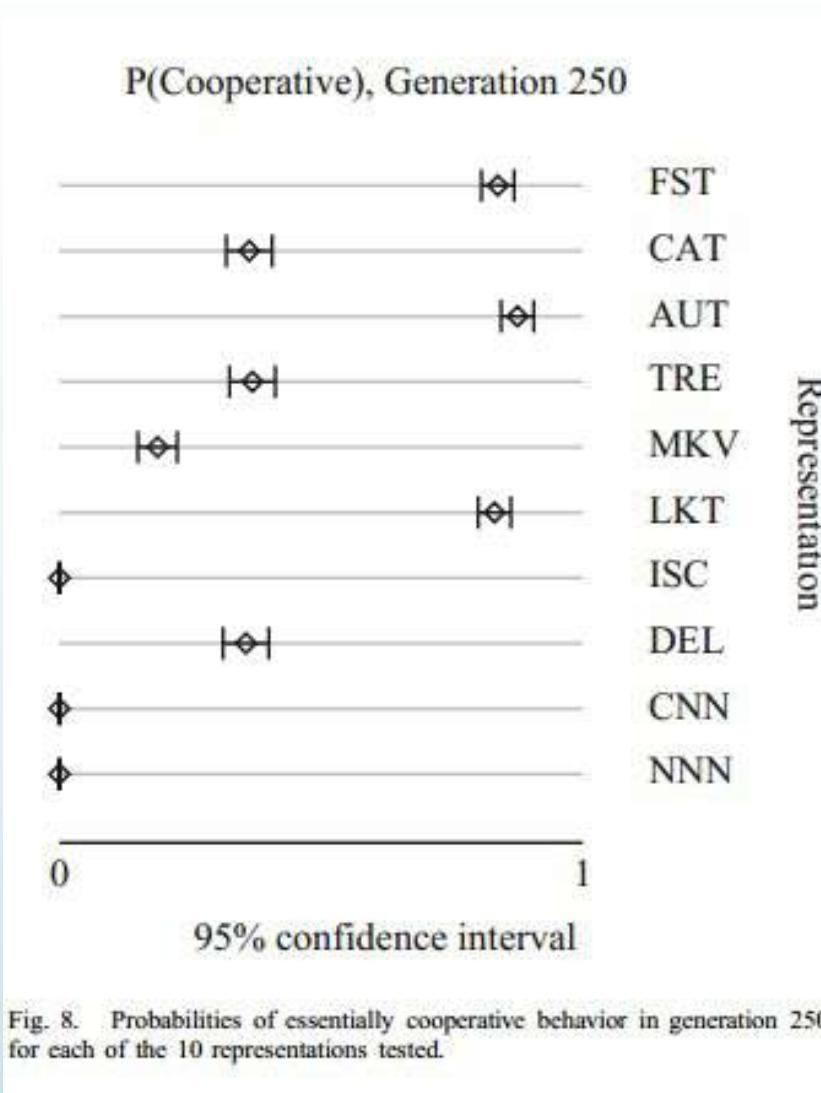
Iterated Prisoner's Dilemma

- ▶ Game playing agents
- ▶ Representations
 - ▶ Function Stack Machines – FST
 - ▶ Cellular Encoded FSM – CAT
 - ▶ Finite State Machines – AUT
 - ▶ Boolean Formulas – TRE
 - ▶ Markov Chains – MKV
 - ▶ Look-up Tables – LKT
 - ▶ IF-ACTION-SKIP Lists - ISC
 - ▶ Boolean Formula with Delays - DEL

Co-evolution

- ▶ The fitness of the population is based not upon the target, but the other members within the population
- ▶ Used to create game player agents
- ▶ For IPD
 - ▶ Play each of the other players in the population once
 - ▶ Score is what you make off of all the other members in the population

Cooperative Ability



Play in the Population v. Random Player

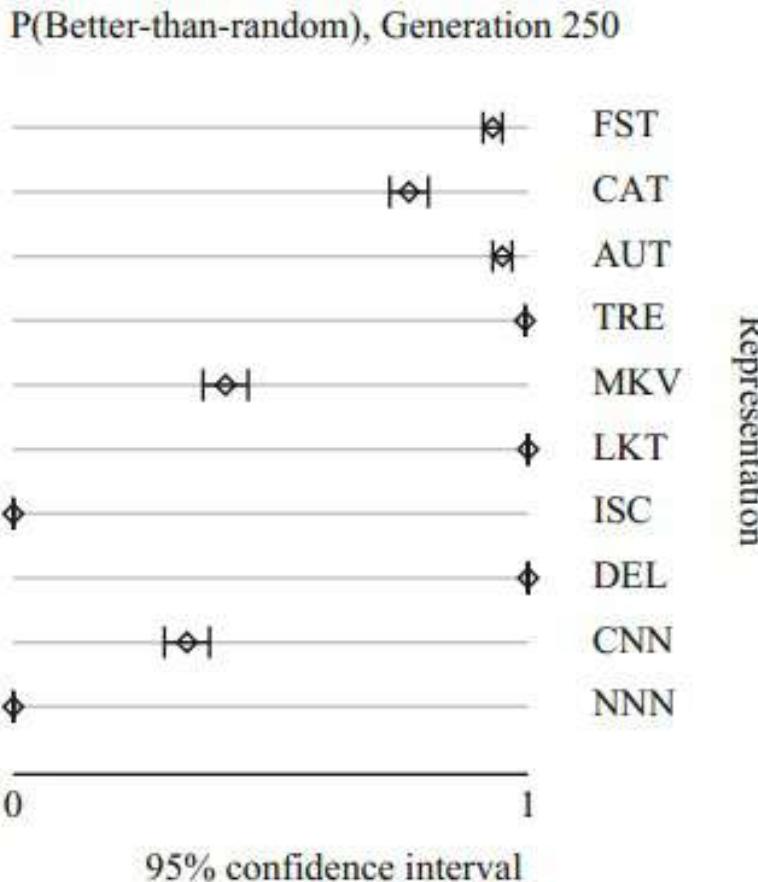


Fig. 9. Probabilities of better-than-random behavior in generation 250 for each of the 10 representations tested.

Why?

1. Initial Population

Random initialization of all of these techniques has some differences in the ability to create an early cooperator

2. Variation Operators with Different Probabilities of Creating “Stupid” Offspring

May allow for offspring which are easy to exploit

3. Fidelity of Reproduction

Differences in the ability to represent ones parents. Some reproduction rules lead to drastically different children than parents

4. Granularity of Reproduction

Cooperation must be valued to be sustained, if newer models are allowed in which can be exploited it allows for exploiters to overrun the population

String Players

- ▶ String of actions
- ▶ Actions made in a step-wise order or are representations of the player
- ▶ Have been used for a number of simulation ideas:
 - ▶ Sunburn – Space Ships
 - ▶ VIPs – Virtual Politician Simulator
 - ▶ Lessons to a young man courting – Dating Simulator
 - ▶ Dawn of the Dead - Government Responses to Zombies
- ▶ Co-evolution and Evolution to a target
 - ▶ Sunburn – co-evolved
 - ▶ VIPs – co-evolved
 - ▶ Lessons – Based on a model of the potential date
 - ▶ Dawn of the Dead – based on an epidemic model

Sunburn

- ▶ Space Warfare Simulation
- ▶ Student Project by John Walker
- ▶ Ships are represented by a string which is an indirect method of building a starships
- ▶ Ships compete in a gladiatorial tournament selection method
- ▶ Ships have a distance measure from each other – 1 dimension

Sunburn Ship

- ▶ Length N string of characters
- ▶ (S)hields – Take a hit of damage before dropping
- ▶ Weapons
 - ▶ (G)uns – Short Range Physical Projectiles which drop in effectiveness as distance increases. Slow moving and able to dodge
 - ▶ (L)asers – Can be equally effective at any range
 - ▶ (M)issiles – Guided missiles which take some time in order to ensure a lock. They have a better long range effectiveness as they track their targets
- ▶ (D)rives – allow a ship to move one space
- ▶ String kept in order – Shields are moved to front of string and tripled. Each system is worth one point of damage
- ▶ Ships have a preferred distance

Weapon Effectiveness

- ▶ A weapon hitting does one system worth of damage
- ▶ A damaged system cannot be used
- ▶ Weapons are probabilistic in their effect
- ▶ Hit based upon the weapon effectiveness curve – a function based on the current range
 - ▶ $PG(r) = 0.6 - (r - 1)/38$
 - ▶ $PL(r) = 0.35$
 - ▶ $PM(r) = 0.1 + (r - 1)/38$

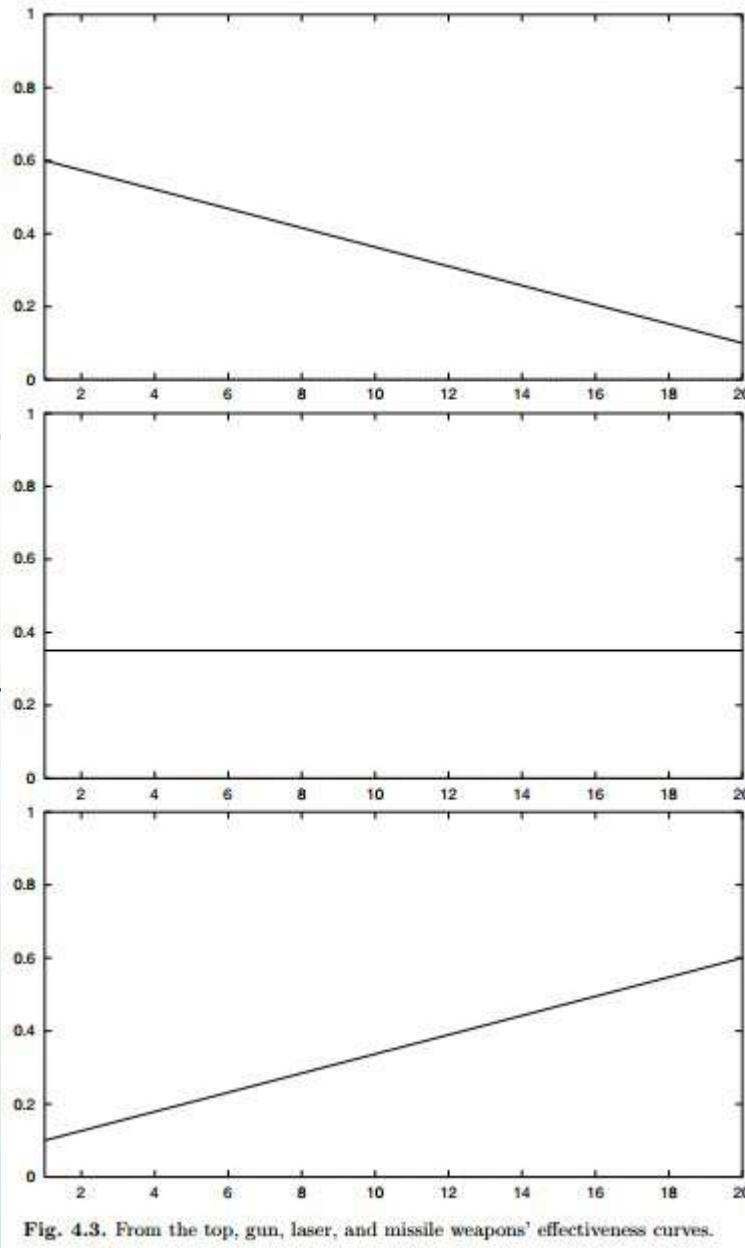


Fig. 4.3. From the top, gun, laser, and missile weapons' effectiveness curves.



Example Ship Translation

- ▶ SGLMSDSMGS,16
- ▶ Systems with shields in the chromosome
- ▶ Translation moves these to the front and triples them
- ▶ 4 shield nodes buys you 12 shields
- ▶ SSSS SSSS SSSS GLMDMGLD,16

Ship Controllers

- ▶ Ships activate their drives in unison one at a time with the goal of moving to their key range
- ▶ Ships fire weapons one at a time in unison, based-on their order in the string

LLGML

FIRE! - LASER, LASER, GUN, MISSILE, LASER

Gladiatorial Selection



Gladiatorial Fights

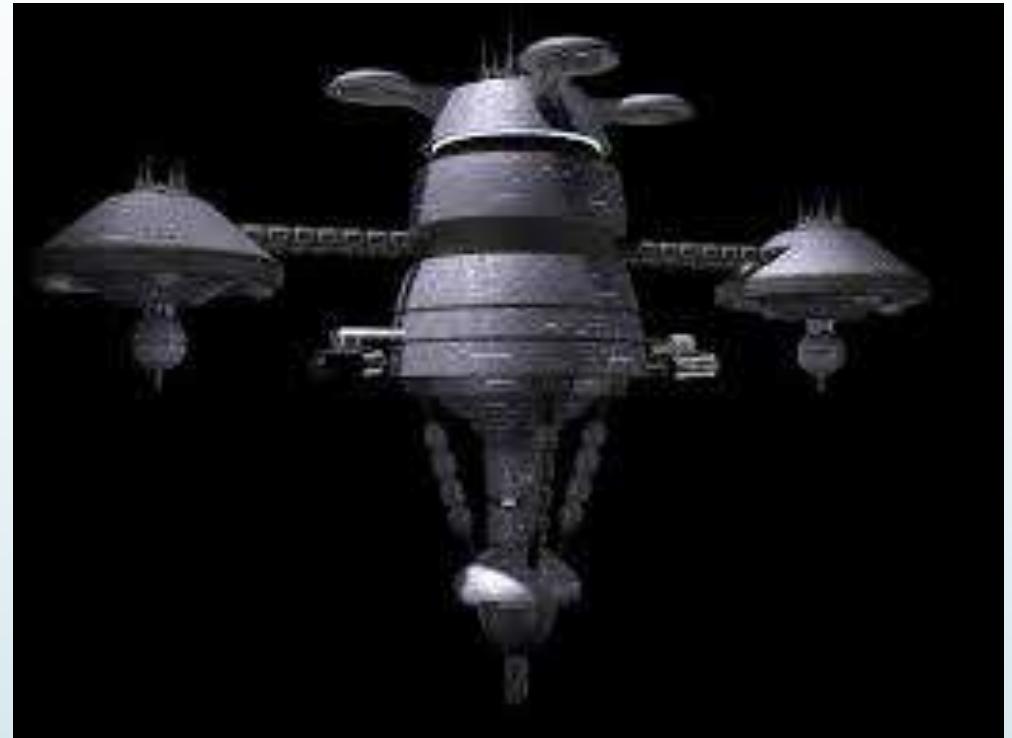
- ▶ Steady State Population
 1. Pull out two ships from the population
 2. Simulate their battle
 3. If there is a winner – keep for breeding
 4. Else return to step 1
 5. Pull out two ships from the population
 6. Simulate their battle
 7. If there is a winner – keep for breeding
 8. Else return to step 5
 9. Breeding Step

Win Conditions

- ▶ A ship without any points remaining is destroyed
- ▶ 100 Turns without damage - Draw
- ▶ Are there any others?

Starbases

- ▶ Strip down to one drive in the final position
- ▶ Load up on Missiles and Shields



Gunrunners

- ▶ Lots of drives
- ▶ MORE GUN



Representational Issues

- ▶ Position within the chromosome has a large effect on the range and the damage
- ▶ Systems ‘go down’ in order
 - ▶ Expendable systems at front
 - ▶ Important systems near end
 - ▶ Win condition is a disabling of drives – ensure at least one drive is at the end of the string
- ▶ Epistatic – with importance on the end of the string

Sunburn

- ▶ What crossovers would make sense?
- ▶ What crossovers would disrupt structure?
- ▶ What about the preferred distance?

Evolutionary techniques (Part III)

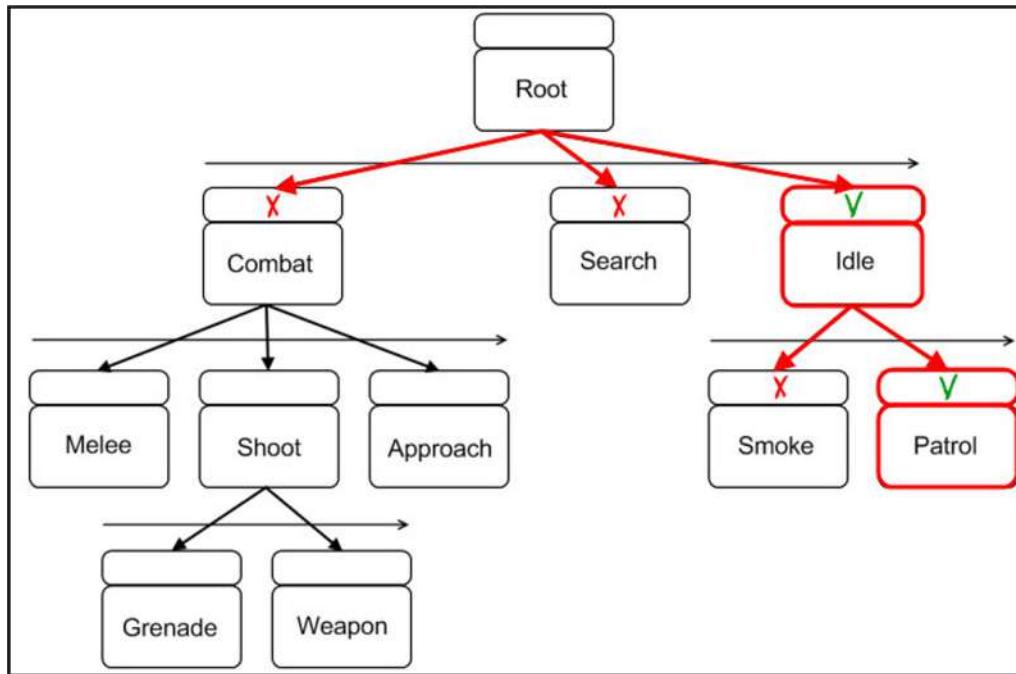
**Darko Bozhinoski,
Ph.D. in Computer Science**

Agenda

- Behavior Trees (BT)

BT: Game Industry

- Originated from Game Industry, as a powerful way to describe AI for “NPC”
 - Halo, Bioshock, Spore



Behaviour Tree Architecture

- ❖ Mathematical Model of Plan Execution – describe switching between a finite set of behaviours in a *modular* fashion
- ❖ Properties
 - ❑ Expressiveness
 - ❑ Accuracy
 - ❑ one-way vs. two-way control transfer
(BTs are more modular than FSMs is that they use a two-way control transfer)
 - ❑ Implementation (memory-less vs. memory-full nodes)

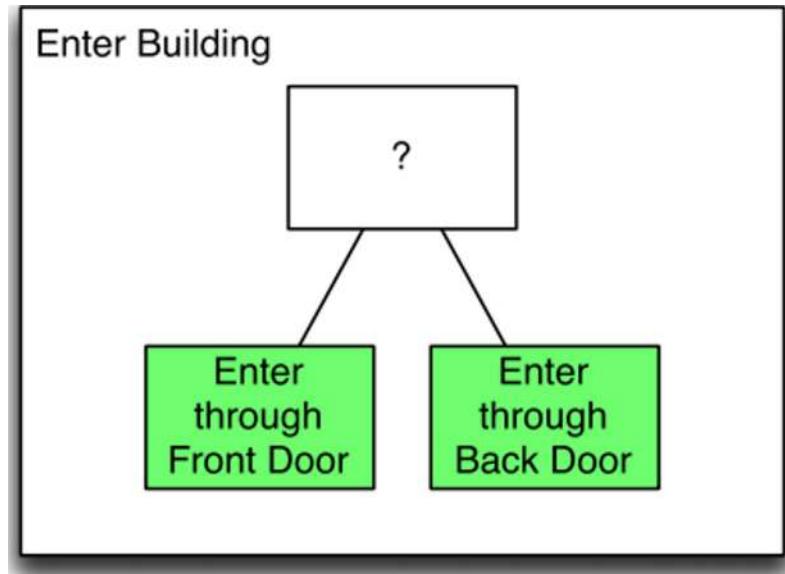
Behaviour Tree Architecture (2)

BT is directed tree in which the nodes are classified as:

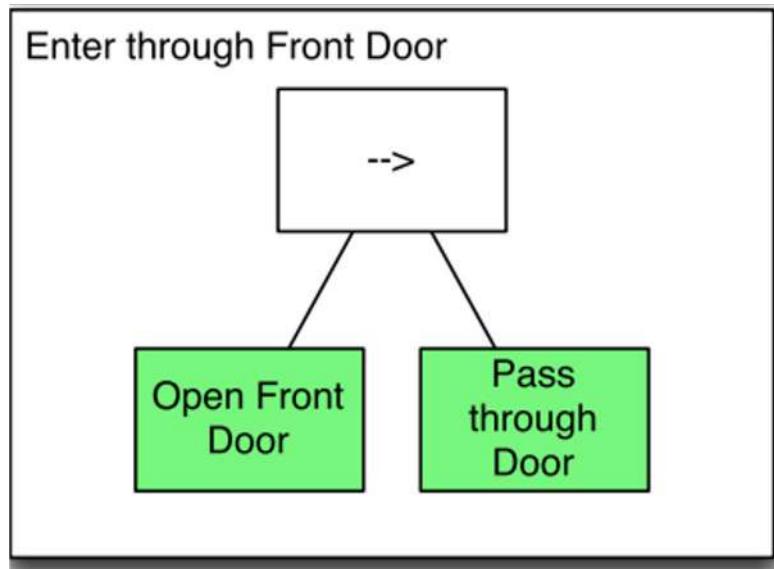
- ❖ Root, control flow nodes and execution nodes
- ❖ The execution of a BT starts from the root which sends ticks with a certain frequency to its child.

Node Type	Symb.	Succeeds if	Fails if	Runs if
Root	\emptyset	tree S	tree F	tree R
Selector	?	1 Ch S	N Ch F	1 Ch R
Sequence	\rightarrow	N Ch S	1 Ch F	1 Ch R
Parallel	\Rightarrow	$\geq S$ Ch S	$\geq F$ Ch F	otherwise
Decorator	\diamond	varies	varies	varies
Action n	\square	$X_n(t) \in S_n$	$X_n(t) \in F_n$	$X_n(t) \in R_n$
Condition n	\circ	$X_n(t) \in S_n$	$X_n(t) \in F_n$	never

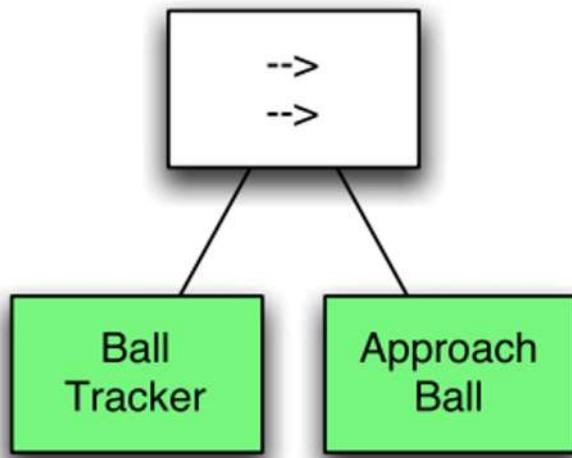
Selector



Sequence



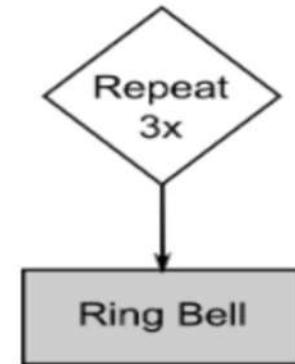
Parallel



Decorator

Decorators are special nodes that can have only a single child. The goal of the decorator is to change the behavior of the child by manipulating the returning value or changing its ticking frequency

- Repeater
- Inverter

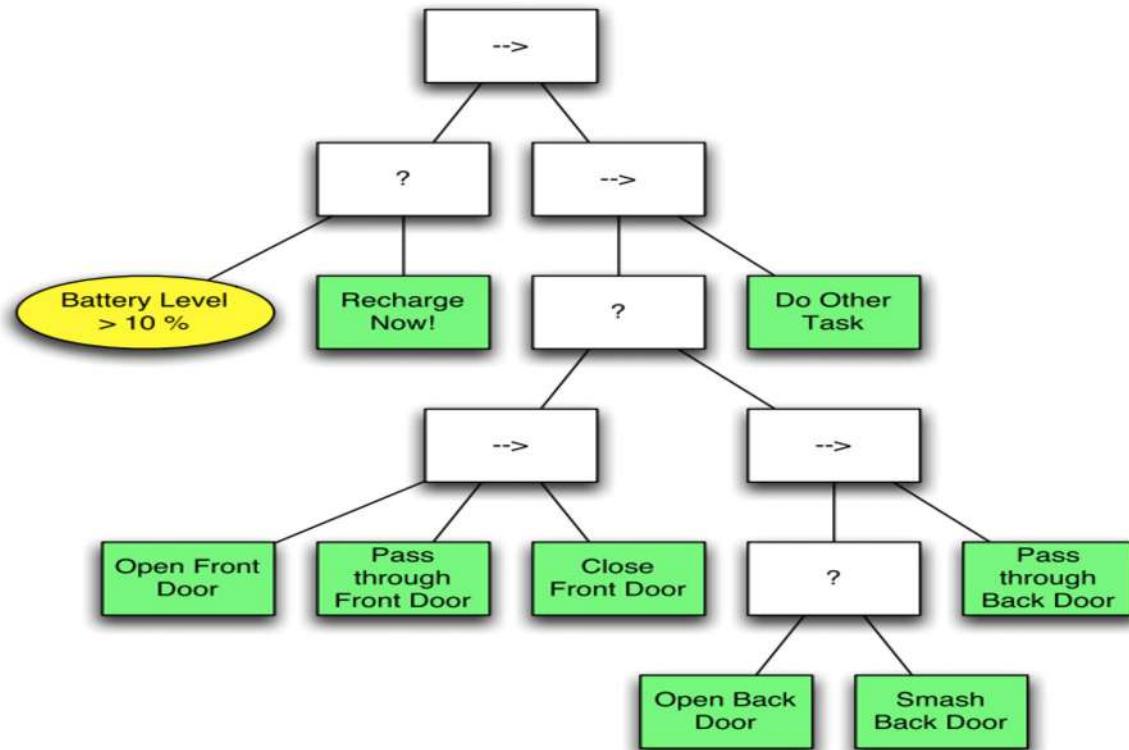


Execution Nodes

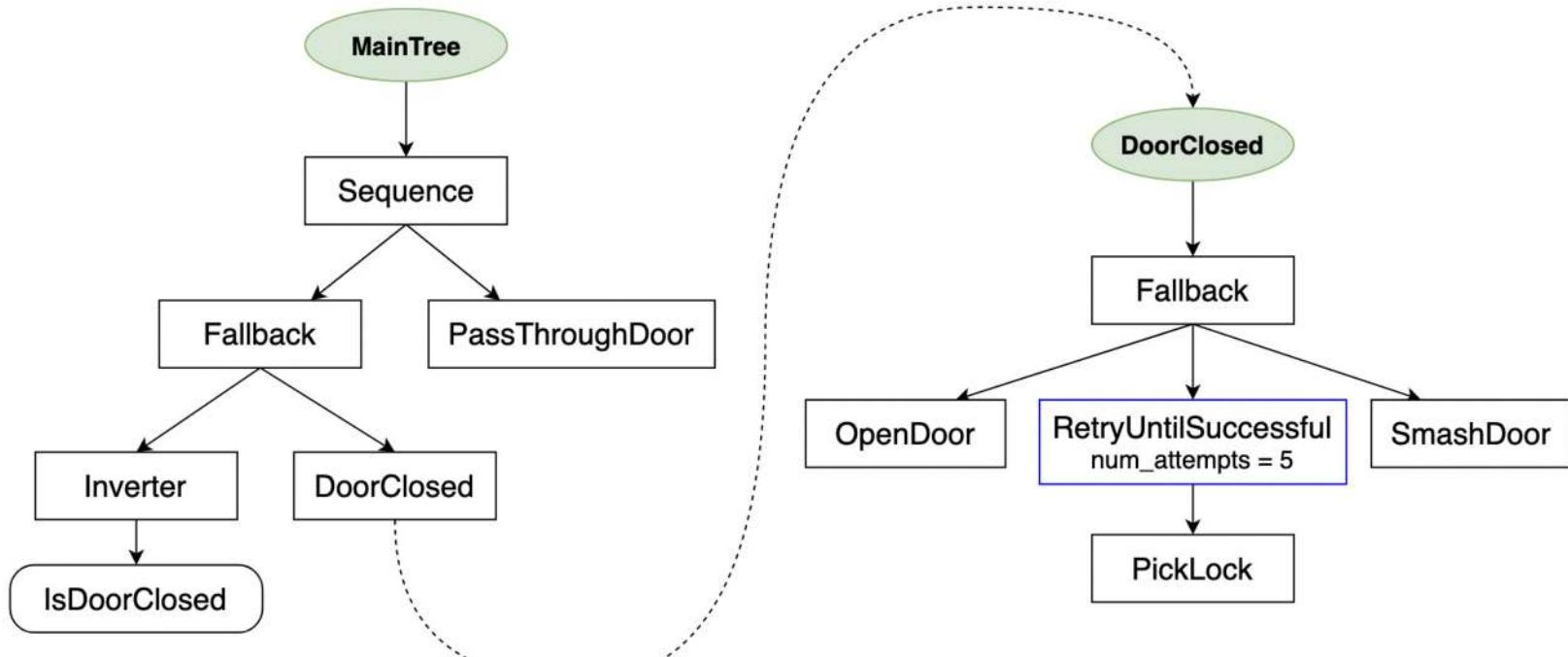
The execution nodes are the primitive building blocks of the behaviour tree. They perform some computation and return a state value (functional).

- ❖ **Condition node** checks whether a certain conditions has been meet and returns TRUE or FALSE to its parent
- ❖ **Action node** (behaviour in our case) performs computations to change the agent state and returns TRUE, FALSE or RUNNING to its parent.

Example BT



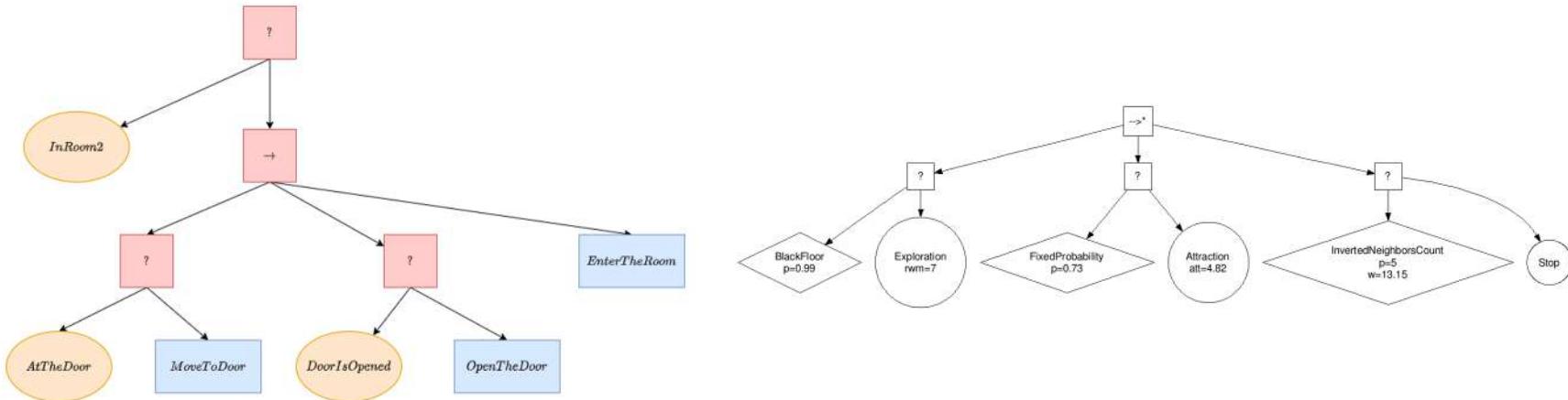
Modularity of BT



BENEFITS OF BT

- ❖ **Maintainability:** transitions in BT are defined by the structure, not by conditions inside the states. Because of this, nodes can be designed independent from each other, thus, when adding or removing new nodes (or even subtrees) in a small part of the tree, it is not necessary to change other parts of the mode.
- ❖ **Scalability:** when a BT have many nodes, it can be decomposed into small sub-trees saving the readability of the graphical model.
- ❖ **Reusability:** due to the independence of nodes in BT, the subtrees are also independent. This allows the reuse of subtrees among other trees or projects.

Other examples



Example. Design Behaviour Tree (5 min)

1. Design a Behaviour Tree for a mobile robot navigating in a supermarket.

Genetic Algorithm Pseudocode

Algorithm 6: Pseudocode of a GA.

- 1 Randomly generate an initial population of I individuals.
 - 2 Calculate cost of each individual i in the population.
 - 3 Select individuals for next generation.
 - 4 Change individuals randomly.
 - 5 Repeat 2-4 until criteria is achieved.
-

BT crossover



BT crossover

Crossover switches a subtree or a single leaf node of a BT with another BT at random. The key is that all parts of the BT can be switched, from single behaviors to large subtrees. After selection, the parents are randomly paired and a random node from each BT is chosen. The crossover can be as easy as switching two leaf nodes between two parents, but can also be used to inherit large subtrees.

BT crossover (2)

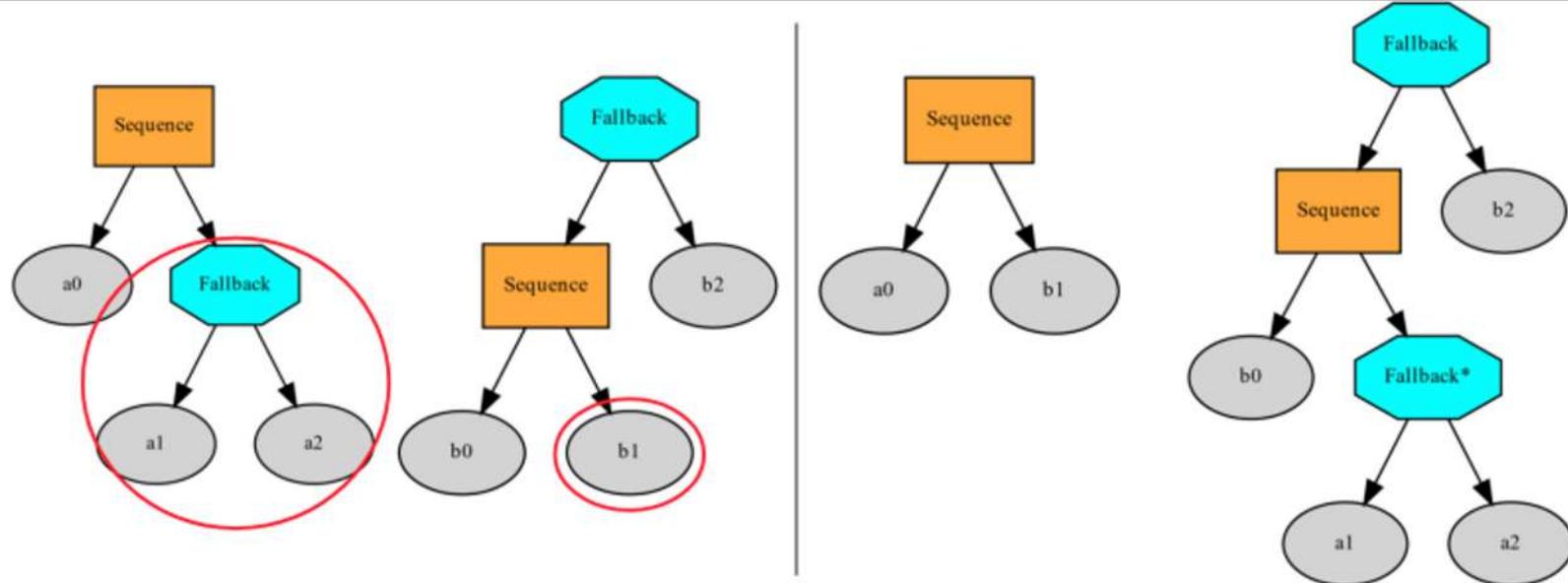


Figure 22: Result of a simple crossover. The parents before crossover can be seen to the left with red circles surrounding the crossover parts. The right part of the tree shows the offspring result.

BT mutations



BT mutations

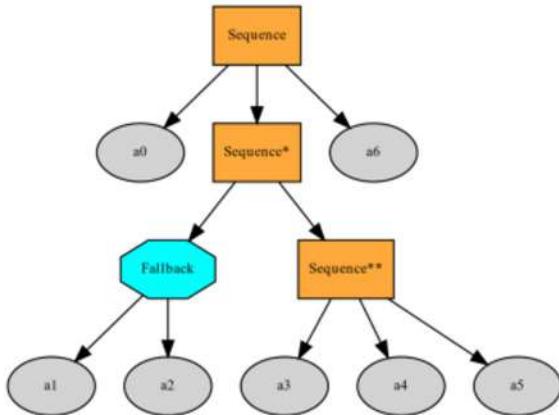
Adding, deleting and switching a node

Adding nodes

- ❖ **Add leaf node:** Since the size of the final BT is not known, the mutation can add a random leaf node to any position of the BT.
- ❖ **Add control node:** Adding a control node to the BT requires it to have some children. The control node inherits all the remaining nodes to the right of where it is added, creating a subtree with all the nodes and subtrees of the current level. If the current position that the control node is added to does not have children to the right, a new position is chosen randomly.

Add a leaf node

['s(', 'a0', 's(', 'f(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']



['s(', 'a0', 's(', 'f(', 'a1', 'a2', ')', 's(', 'a3', 'b0', 'a4', 'a5', ')', ')', 'a6', ')']

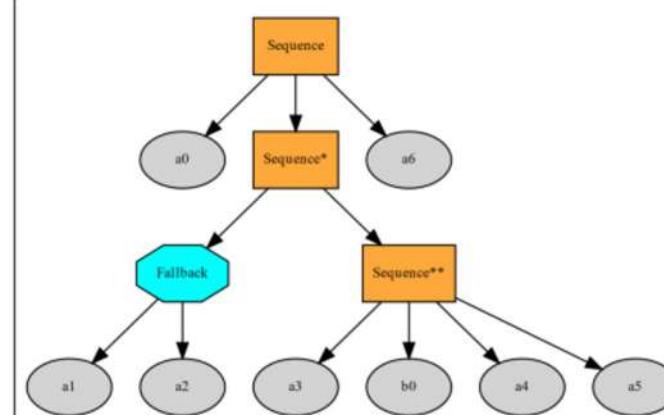
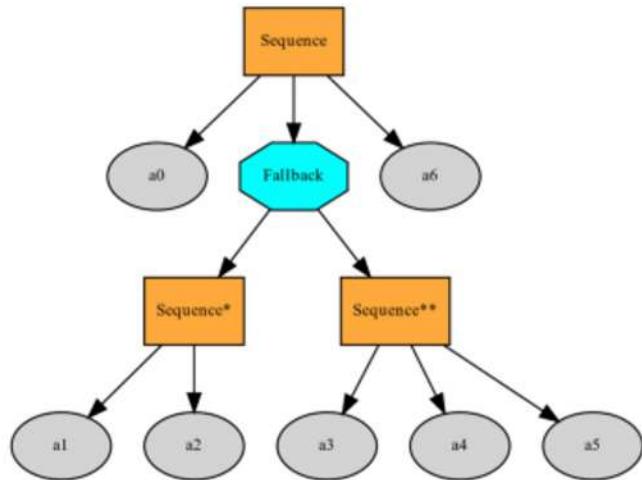


Figure 19: Add a leaf node “b0” into sequence node.

Add a control node

['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']



['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'f(', 'a4', 'a5', ')', ')', 'a6', ')']

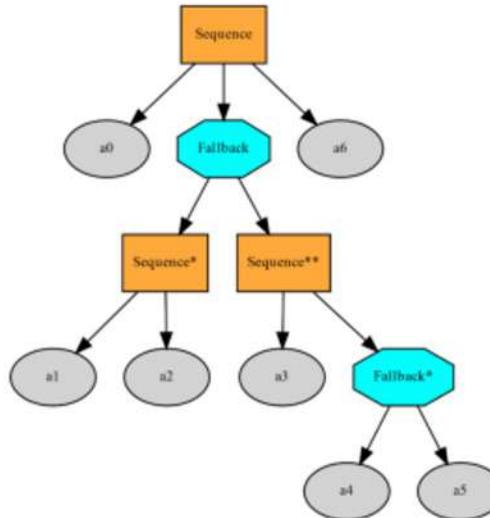
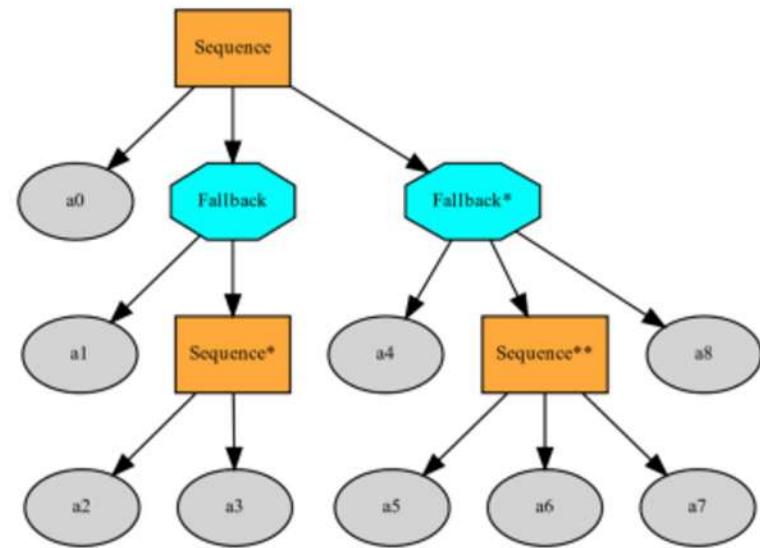
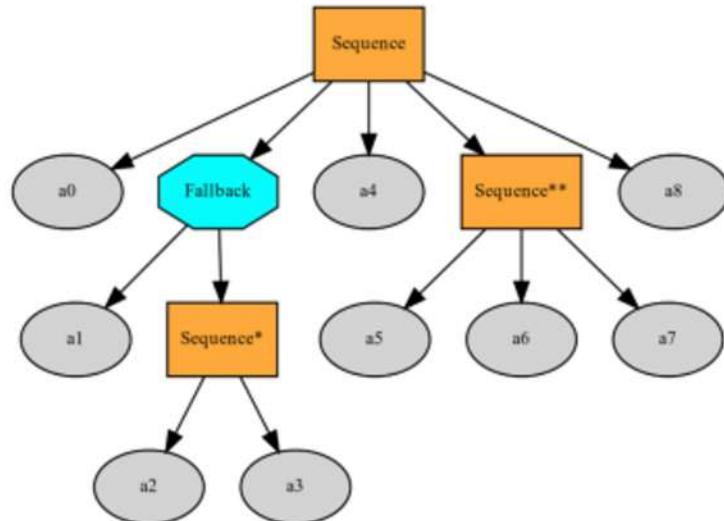


Figure 38: Add control node, which inherits the remaining leaf nodes to the right of its parent.

Add a control node (2)



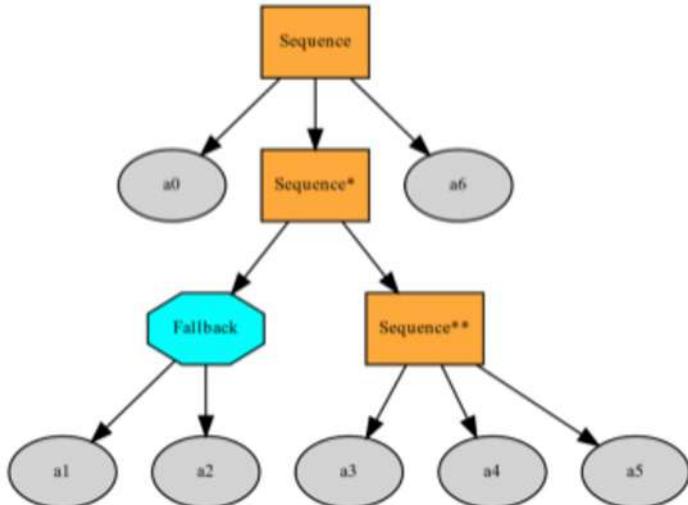
The control node inherits the remaining nodes to the right of its parent.

Deleting nodes

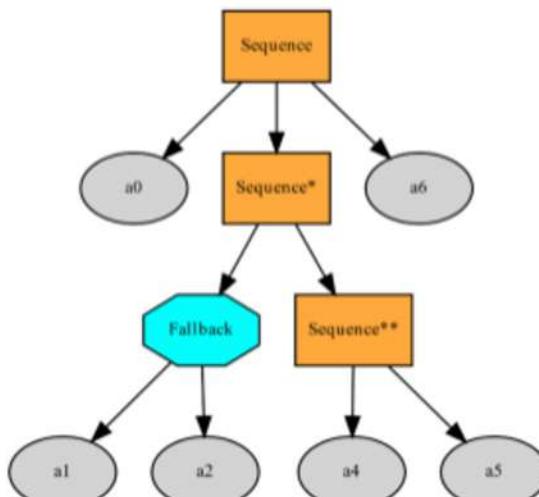
- ❖ Deleting a leaf node
- ❖ Deleting a control node

Delete a leaf node

['s(', 'a0', 's(', 'f(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']



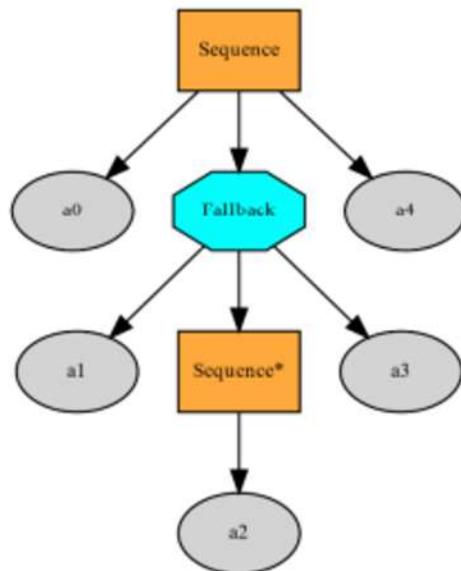
['s(', 'a0', 's(', 'f(', 'a1', 'a2', ')', 's('a4', 'a5', ')', ')', 'a6', ')']



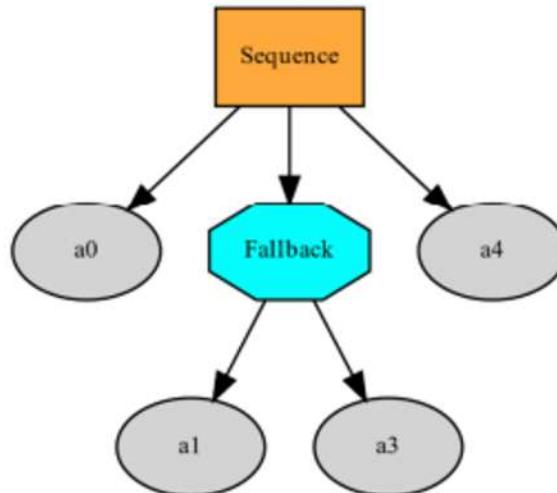
Delete leaf node “a3”

Delete a leaf node - Special case

['s(', 'a0', 'f(', 'a1', 's(', 'a2', ')', 'a3', ')', 'a4', ')']



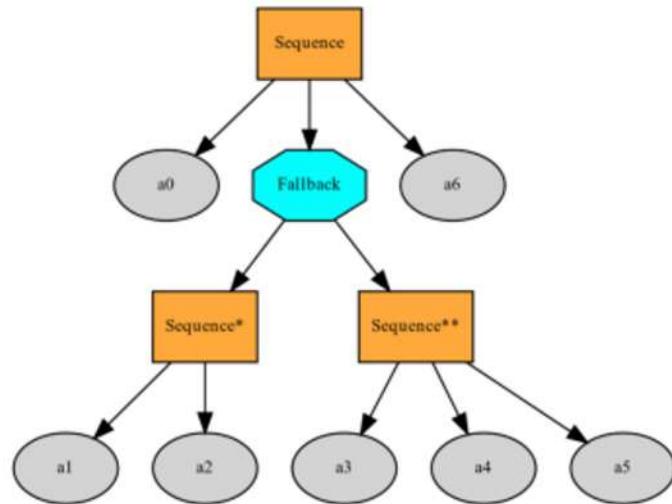
['s(', 'a0', 'f(', 'a1', 'a3', ')', 'a4', ')']



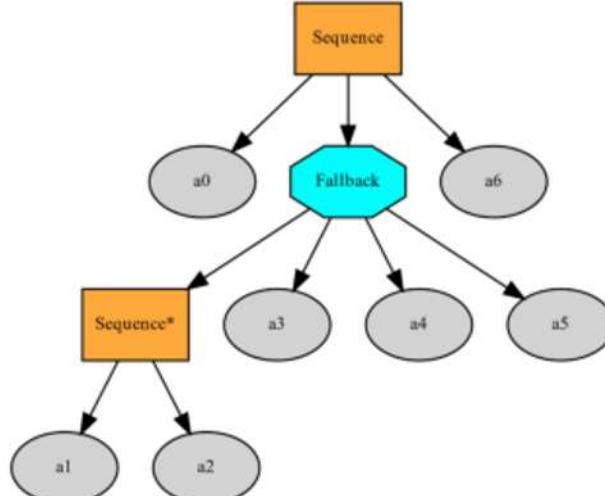
When deleting the leaf node "a2" that is the only child of a control node, also delete the control node.

Delete control node

```
['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']
```



```
['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']
```



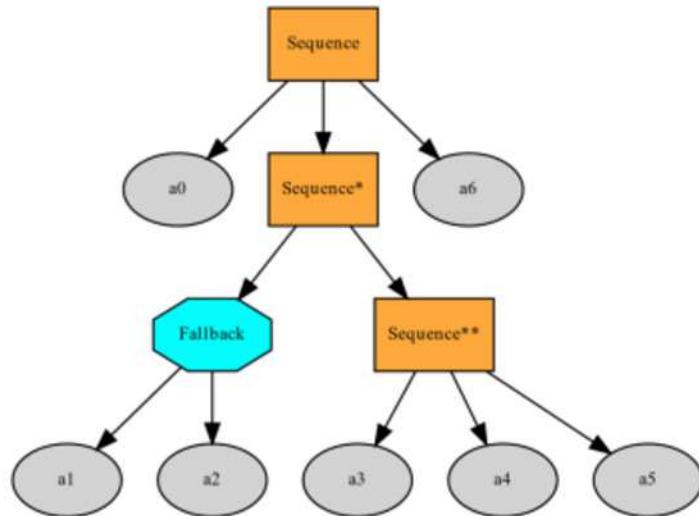
The parent keeps the control node's children.

Switching nodes

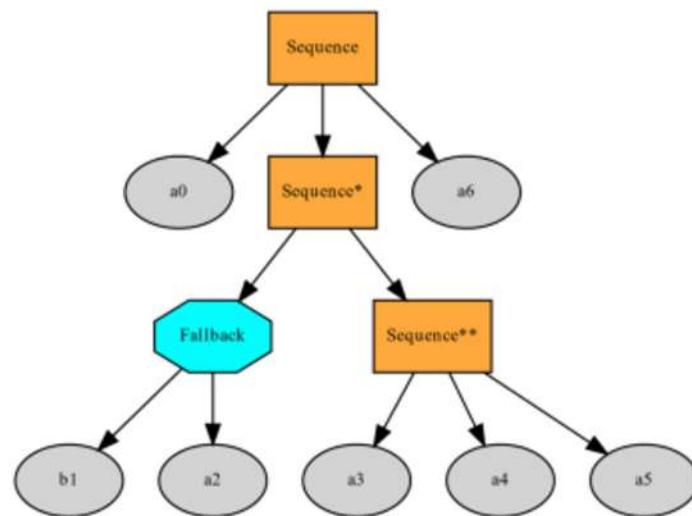
- ❖ Switch a leaf node into another leaf node
- ❖ Switch a control node into another control node
- ❖ Switch a leaf note into a control node
- ❖ Switch a control node into a leaf node

Switch a leaf node into another leaf node

['s()', 'a0', 's()', 'f()', 'a1', 'a2', ')', 's()', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']

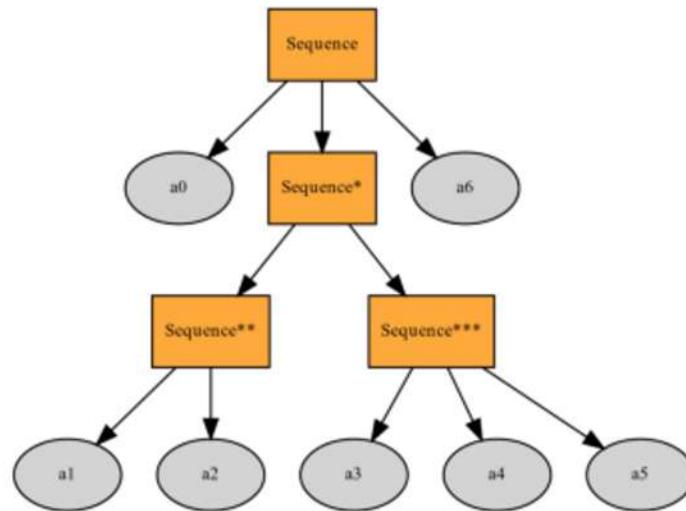


['s()', 'a0', 's()', 'f()', 'b1', 'a2', ')', 's()', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']

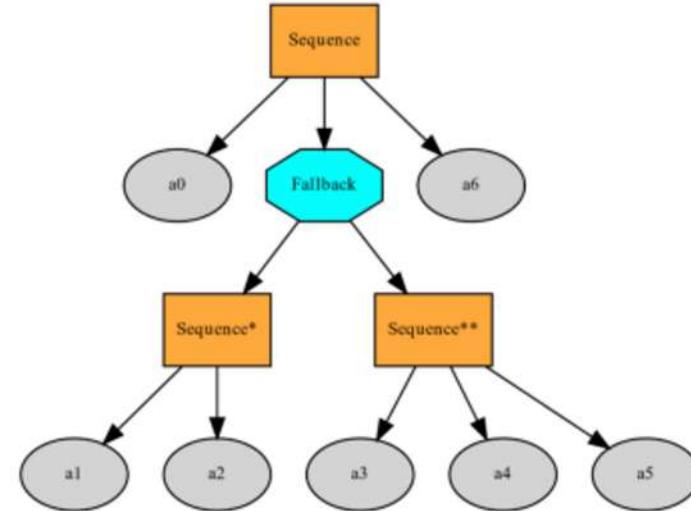


Switch a control node into another control node

['s(', 'a0', 's(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']

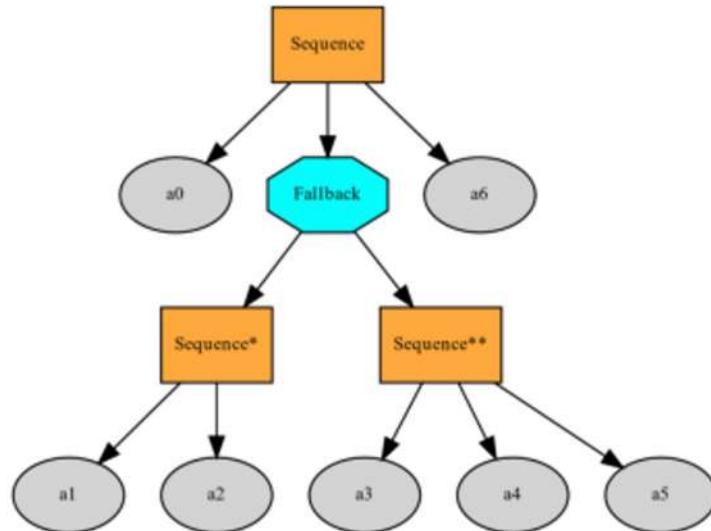


['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']

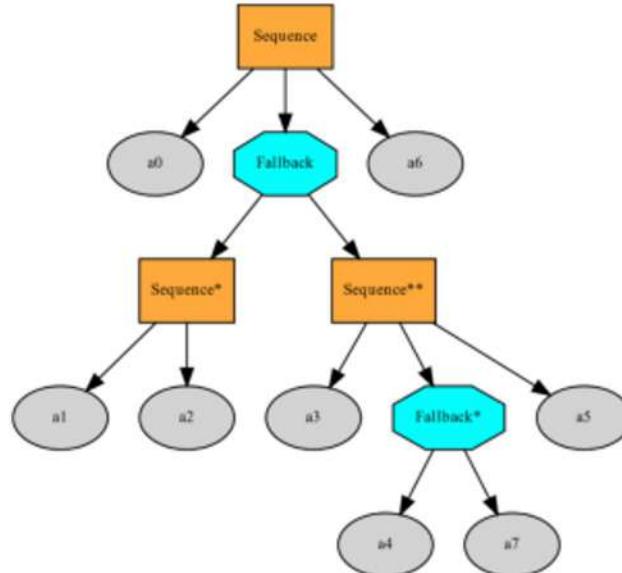


Switch a leaf node into a control node

```
['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']
```



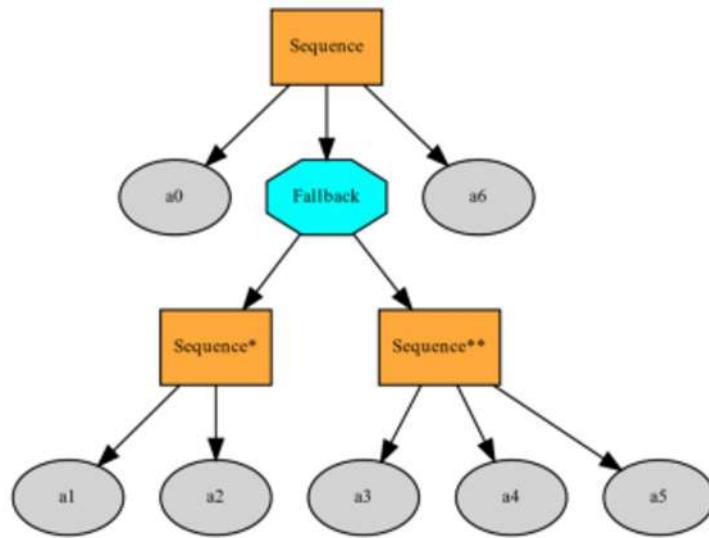
```
['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'f(', 'a4', 'a7', ')', 'a5', ')', ')', 'a6', ')']
```



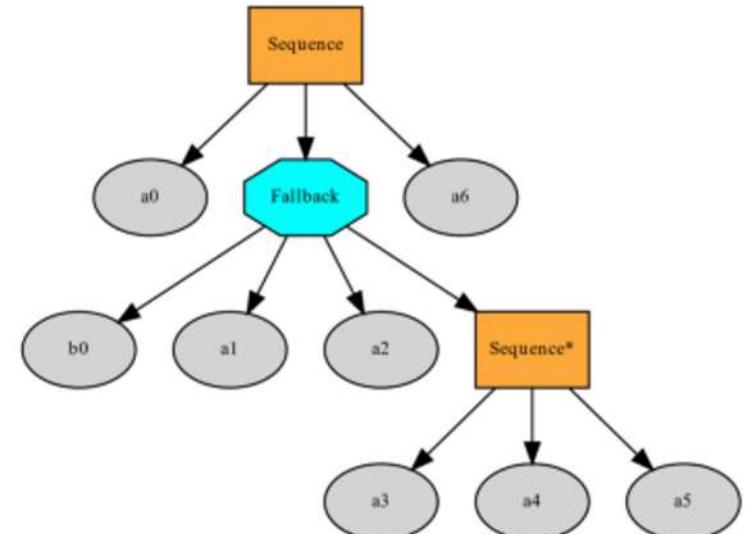
The control node keeps the mutating leaf node and adds a random behavior

Switch a control node into a leaf node

```
['s(', 'a0', 'f(', 's(', 'a1', 'a2', ')', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']
```



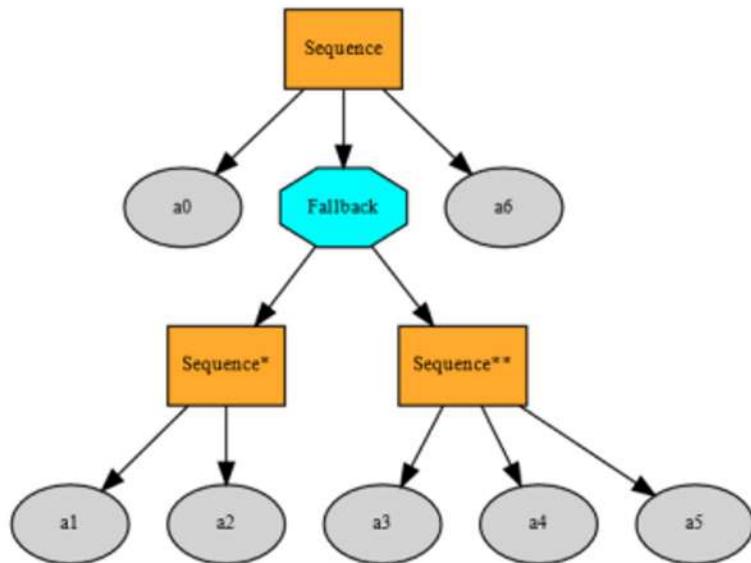
```
['s(', 'a0', 'f(', 'b0', 'a1', 'a2', 's(', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']
```



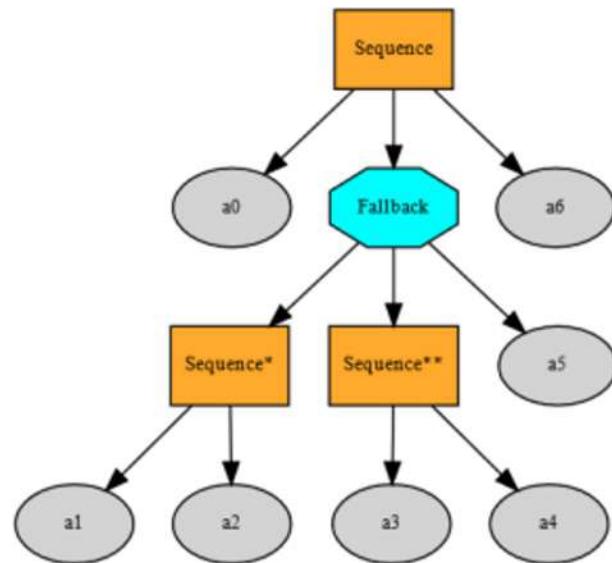
The bottom left control node is switched to a leaf node "b0".

Modifying the tree structure

```
['s()', 'a0', 'f()', 's()', 'a1', 'a2', ')', 's()', 'a3', 'a4', 'a5', ')', ')', 'a6', ')']
```



```
['s()', 'a0', 'f()', 's()', 'a1', 'a2', ')', 's()', 'a3', 'a4', 'a5', ')', 'a5', ')']
```



Closing control nodes of the BT.

Three light-colored wooden blocks are arranged horizontally. The first block contains the letter 'Q', the second block contains the symbol '&', and the third block contains the letter 'A'. All letters are bold, black, sans-serif font. The blocks are resting on a dark brown wooden surface. The background is a soft-focus, warm-toned outdoor scene.

Q & A

Introduction to PDDL

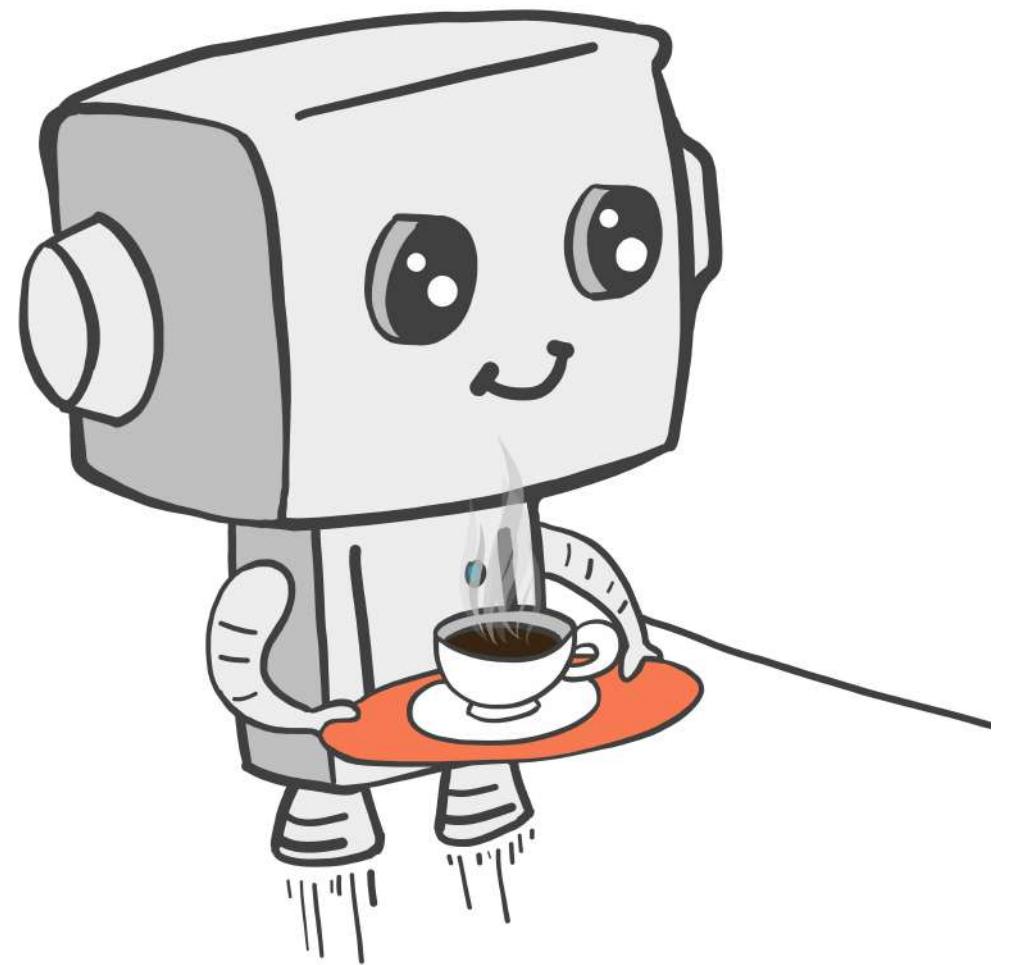
Darko Bozhinoski,

Ph.D. in Computer Science

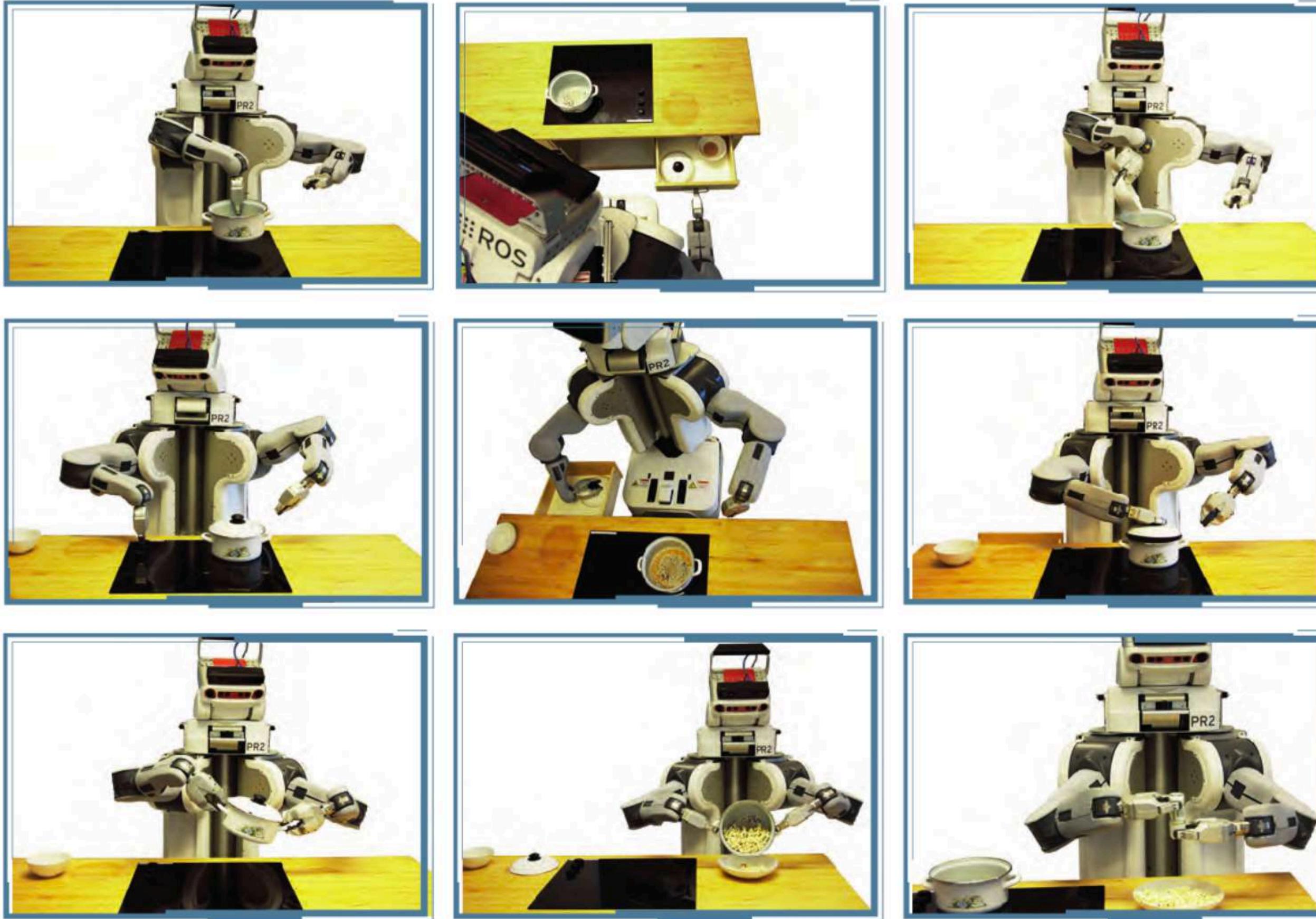
Motivations

- What is the Planning Domain Definition Language (PDDL)?

? ? ?
?



Motivations



- Formulated the problem:
 - First Order Logic (FOL)
- PROLOG

Fig. From [Beetz-2020]

Motivations

- Schema for a **Knowledge-enabled** agent:
- **Action plan:**
 - Action sequencer: solves the planning problem
- **Execution of the plan:**
 - If the goal can be satisfied actions are executed

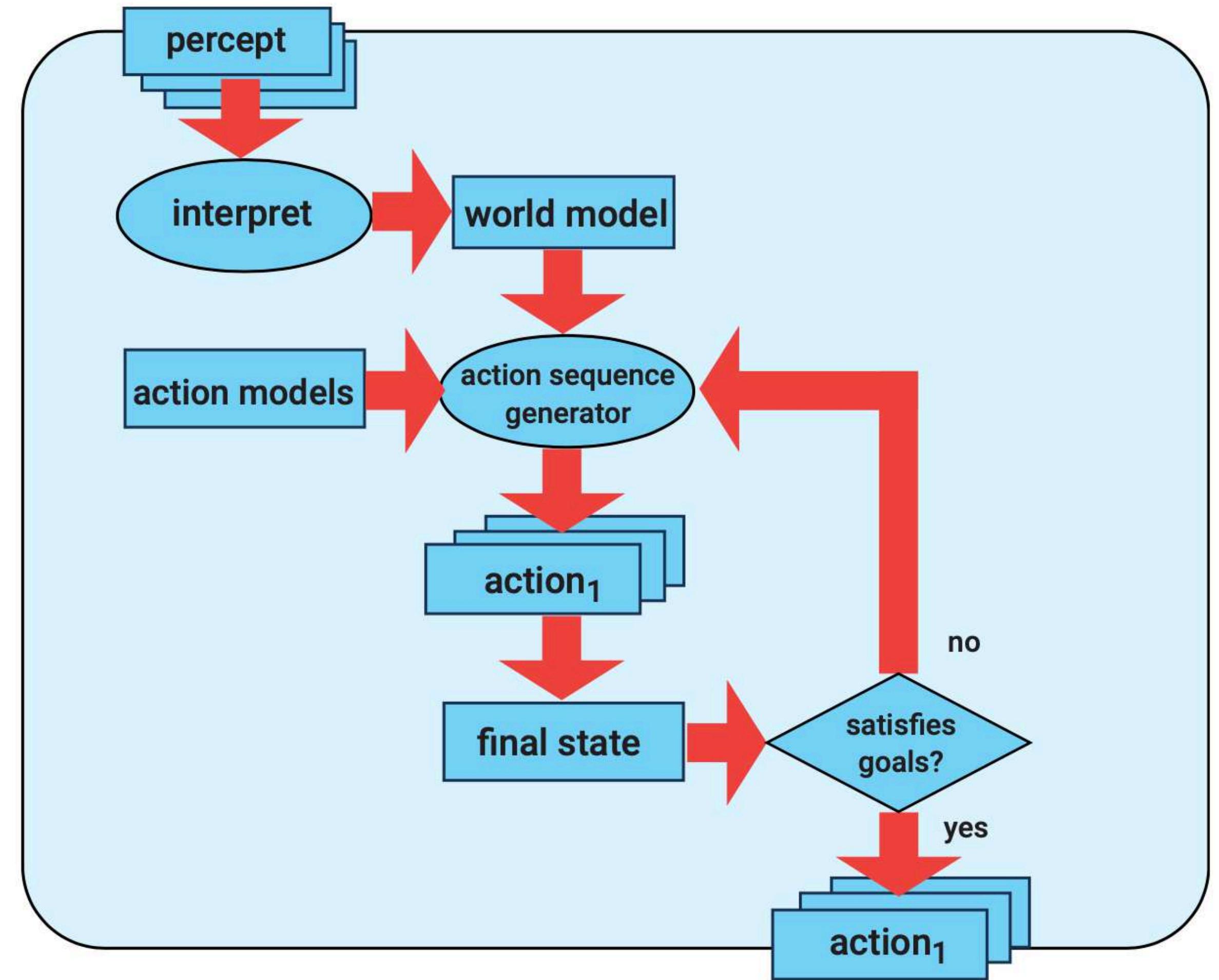


Fig. From [Beetz-2020]

Motivations

- PDDL:

A framework for formulating and solving planning problem.

What is planning?

Intuitive definition:

Planning is the **thinking** before **acting**

More precise definition:

Planning is the process of **choosing** and **scheduling actions** that lead towards a **goal**, based on a **high-level description** of the world

Two families of planning approaches

Domain specific planning

- Methods and representations are specific to a problem:
 - Path and motion planning
 - Manipulation planning
 - ...

Domain independent planning

- Applicable across very different domains
- Many different flavours
 - Offline/online
 - Discrete/continuous
 - Deterministic/non-deterministic
 - Fully/partially observable
 - Sequential/temporal

Planning Domain Definition Language

We need to formally describe:

- The environment in which we operate: the **initial state**, and the **goal**
- The **actions** that our robot can do, and how these affect the environment



PDDL - The classical planning problem

The classical (simplest) PDDL planning problem is described by a tuple:

$$\langle F, A, I, G \rangle$$

where:

- F is a set of boolean propositions (facts)
- A is a set of deterministic actions (with preconditions and effects)
- $I \in S$ is the initial state
- G is the goal state

The set of states in S is the power set of F , $S = 2^F$

PDDL - The classical planning problem

The classical PDDL assumes:

- Deterministic and instantaneous actions
- Boolean propositions to describe the world
- The world only changes according to the actions we apply (no external events)
- The system is fully observable (i.e. I is known)

PDDL - The classical planning problem

Classical PDDL might seem limiting:

- Assumptions might appear unrealistic for real use-cases

Why is it so popular then?

- It provides idealised settings to develop automated planners
- In many cases it is enough - simplified robotics task in controlled environments

Extensions:

- PDDL2.1, Probabilistic PDDL (PPDDL), RDDL, and many more...

PDDL - Structure

Planning tasks in classical PDDL relies on two separate files:

Domain file
domain.pddl

- **Predicates:** properties of the world
- **Operators:** the way in which the state can change
- First order logic (FOL): derive F and A

Problem file
problem.pddl

- **Objects:** the things in the world
- **Initial state**
- **Goal**

PDDL - Solvers

How to find a plan to achieve a goal?

- Solvers take as input a *domain.pddl* and *problem.pddl*, and generate a *plan*
- The planning problem can be seen as **network of states**, where actions represent transitions.
- It is a search problem: MCTS, A*...

Syntax and tools

Domain file
domain.pddl

PDDL - Syntax

```
-----  
(define (domain <domain-name>)  
  (:requirements  
   <requirements-list>)  
  (:types  
   <types-list>)  
  (:predicates  
   <predicate-list>)  
  (:action  
   <action-details>)  
)  
-----
```

- <domain-name> is the name of the world
- :requirements - which features of PDDL are being used
- :types - optional, type of objects
- :predicates - list of model's state, true or false facts
- :action - list of possible state transitions

Problem file
problem.pddl

```
-----  
(define (problem <title>)  
  (:domain <domain-name>)  
  (:objects  
    <object-list>)  
  (:init  
    <predicates>)  
  (:goal  
    <predicates>)  
)  
-----
```

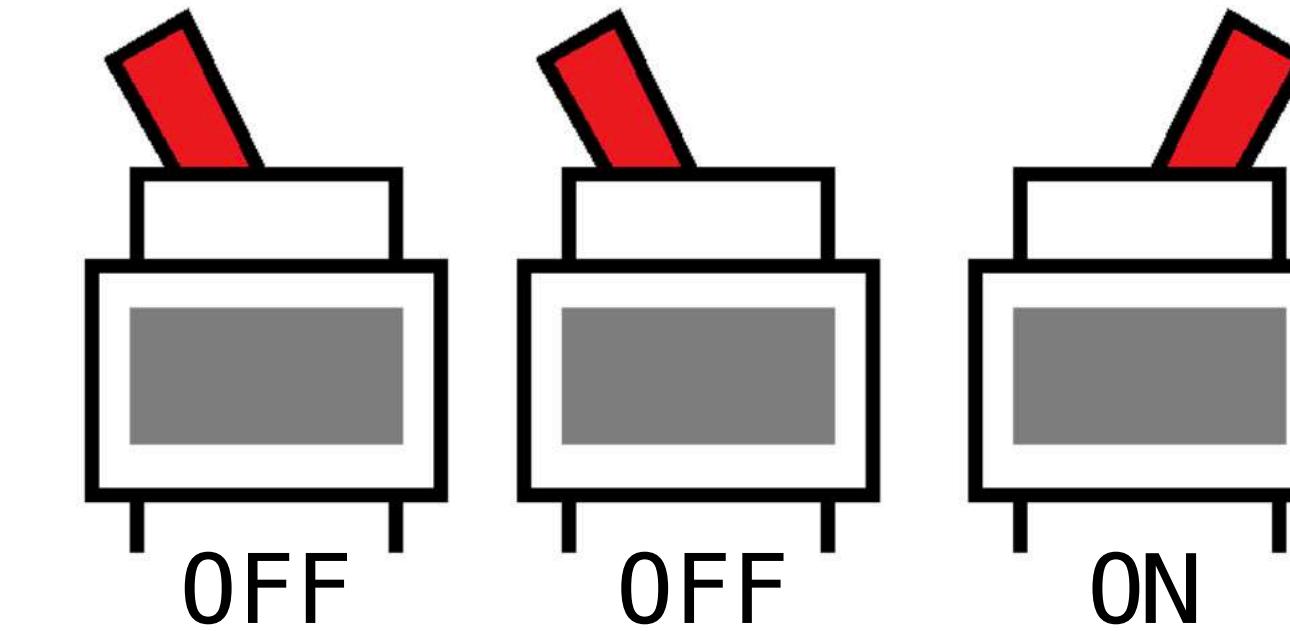
PDDL - Syntax

- <title> is the name you give to the problem
- <domain-name> refers to the relative domain.pddl
- :objects - things in the world you want to manipulate
- :init - the initial state of your objects
- :goal - final state(s) to be achieved

Classic PDDL - Example

domain.pddl

```
(define (domain simple_switches)
  (:requirements :typing)
  (:types switch)
  (:predicates (off ?s - switch)
               (on ?s - switch))
  (:action switch_on
    :parameters (?s - switch)
    :precondition (off ?s)
    :effect (and (not (off ?s))
                  (on ?s))))
```



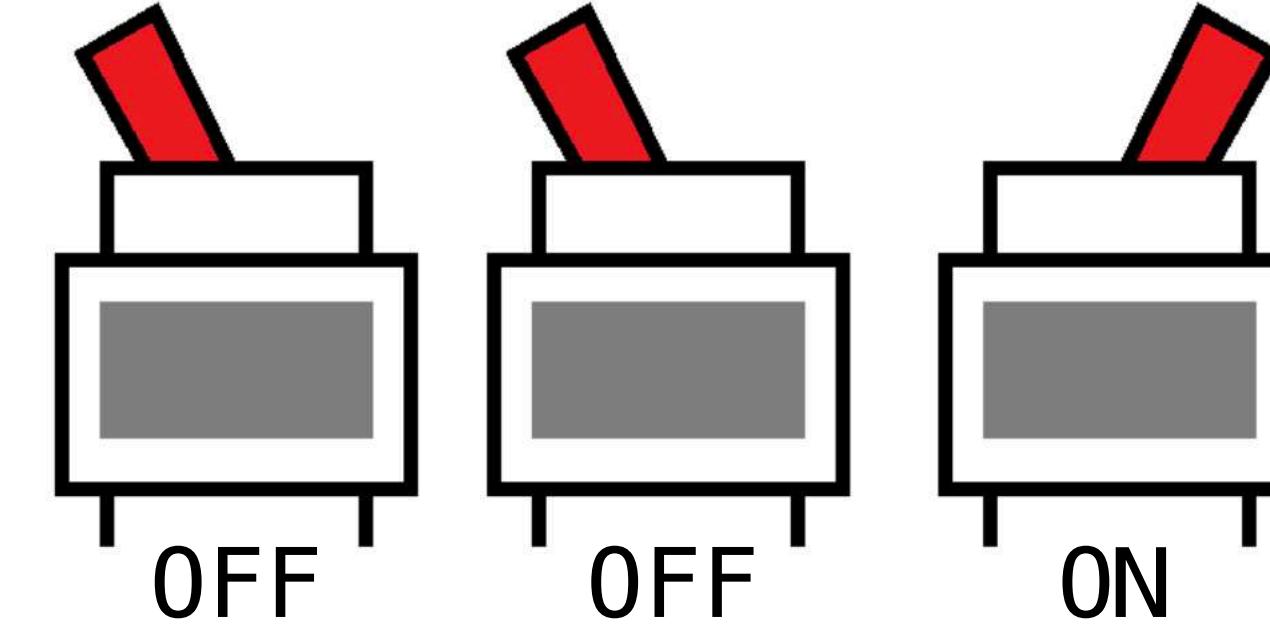
- An action has:
 - $\text{par}(a) \subseteq F$: parameters
 - $\text{pre}(a) \subseteq F$: simple conditions
 - $\text{add}(a) \subseteq F$: add effects
 - $\text{del}(a) \subseteq F$: delete effects
- The order of expressions is prefix
 - (**operator** <arguments>):

$$(\text{and } \langle\text{conjunct1}\rangle \dots \langle\text{conjunctN}\rangle)$$

Classic PDDL - Example

problem.pddl

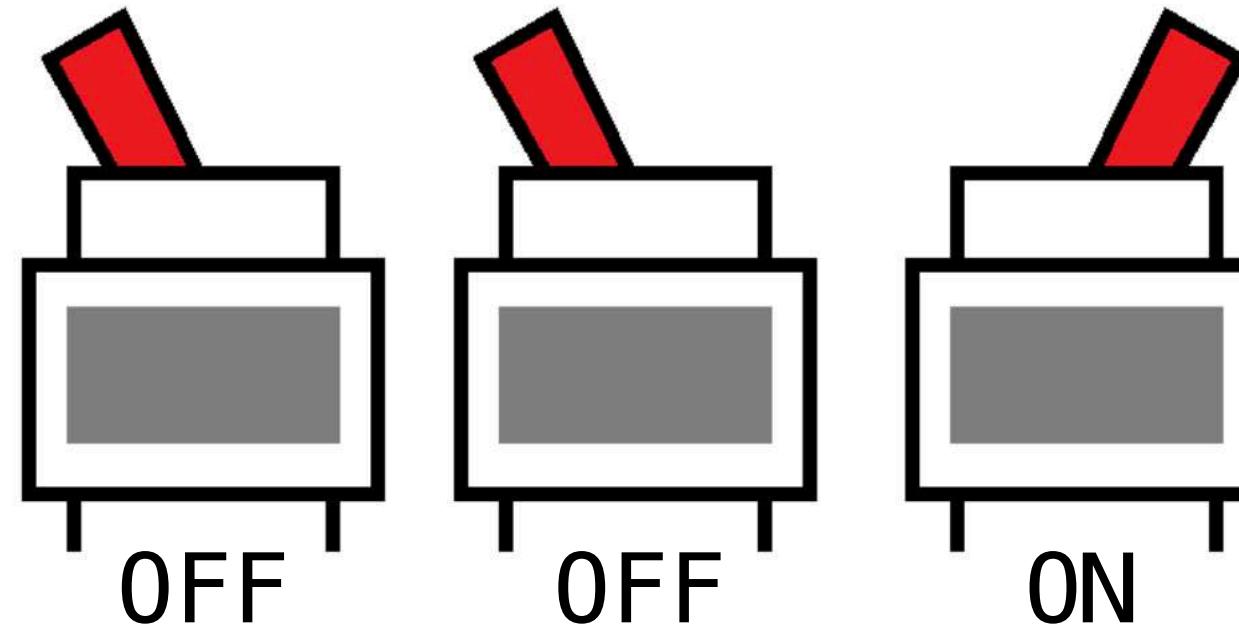
```
(define (problem switches_problem)
  (:domain simple_switches)
  (:objects s1 s2 s3 - switch)
  (:init (off s1) (off s2) (on s3))
  (:goal (and (on s1)
               (on s2)
               (on s3)))) )
```



- The domain in which we operate is `simple_switches`
- There are 3 objects of type `switch`
- Two out of three switches are `OFF`
- The goal is to have all the switches `ON`
- Different problems, same domain!!!

Classic PDDL - Example

A **plan** for a classical planning problem is a sequence of actions that are applicable from the initial state and lead to a state that satisfies the goal



```
( :goal (and (on s1)
              (on s2)
              (on s3)))
)
```

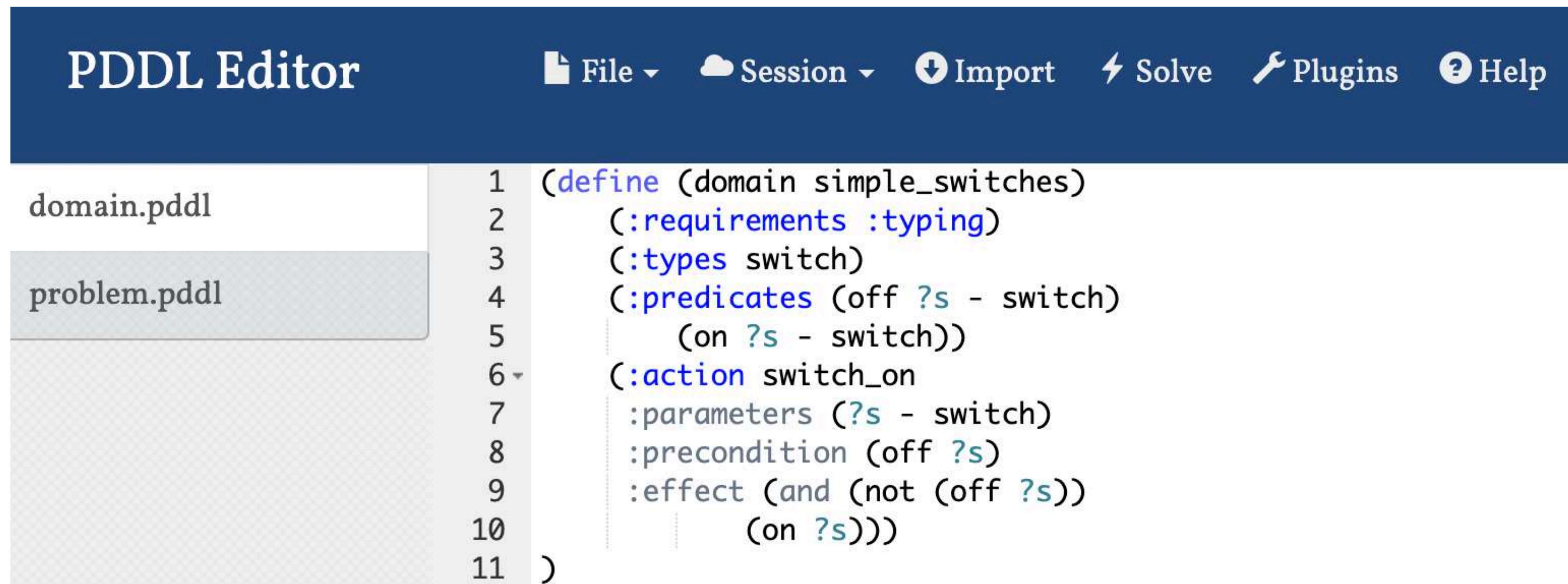
PLAN:

```
-----  
| (switch_on s1)  
| (switch_on s2)  
-----
```

Classic PDDL - Example

A **plan** for a classical planning problem is a sequence of actions that are applicable from the initial state and lead to a state that satisfies the goal

Online PDDL Editor: to write and solve PDDL planning problems:



PDDL Editor

File Session Import Solve Plugins Help

domain.pddl

problem.pddl

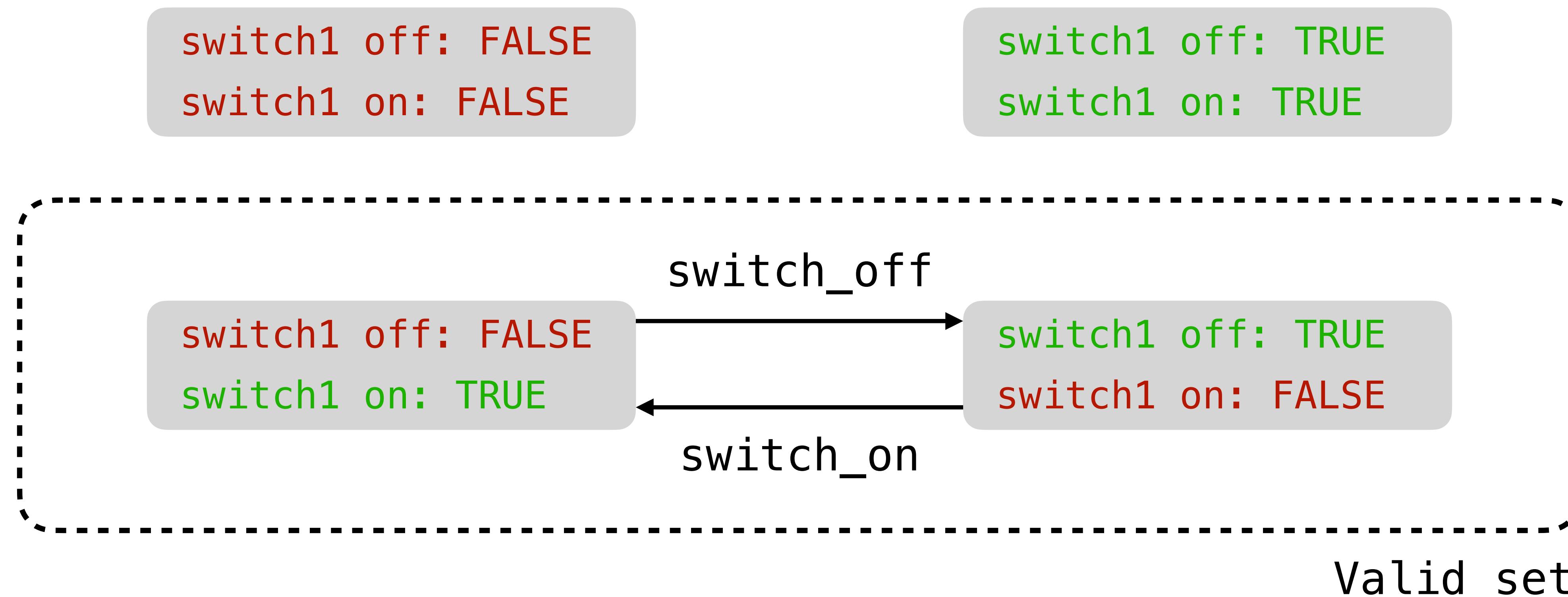
```
1 (define (domain simple_switches)
2   (:requirements :typing)
3   (:types switch)
4   (:predicates (off ?s - switch)
5                (on ?s - switch))
6   (:action switch_on
7     :parameters (?s - switch)
8     :precondition (off ?s)
9     :effect (and (not (off ?s))
10              (on ?s)))
11 )
```

Did you create the switch_off action?

```
(:action switch_off  
  :parameters (?s - switch)  
  :precondition (on ?s)  
  :effect (and (not (on ?s))  
             (off ?s))))
```

http://editor.planning.domains/#edit_session=wQgyhJKiR0wyquH

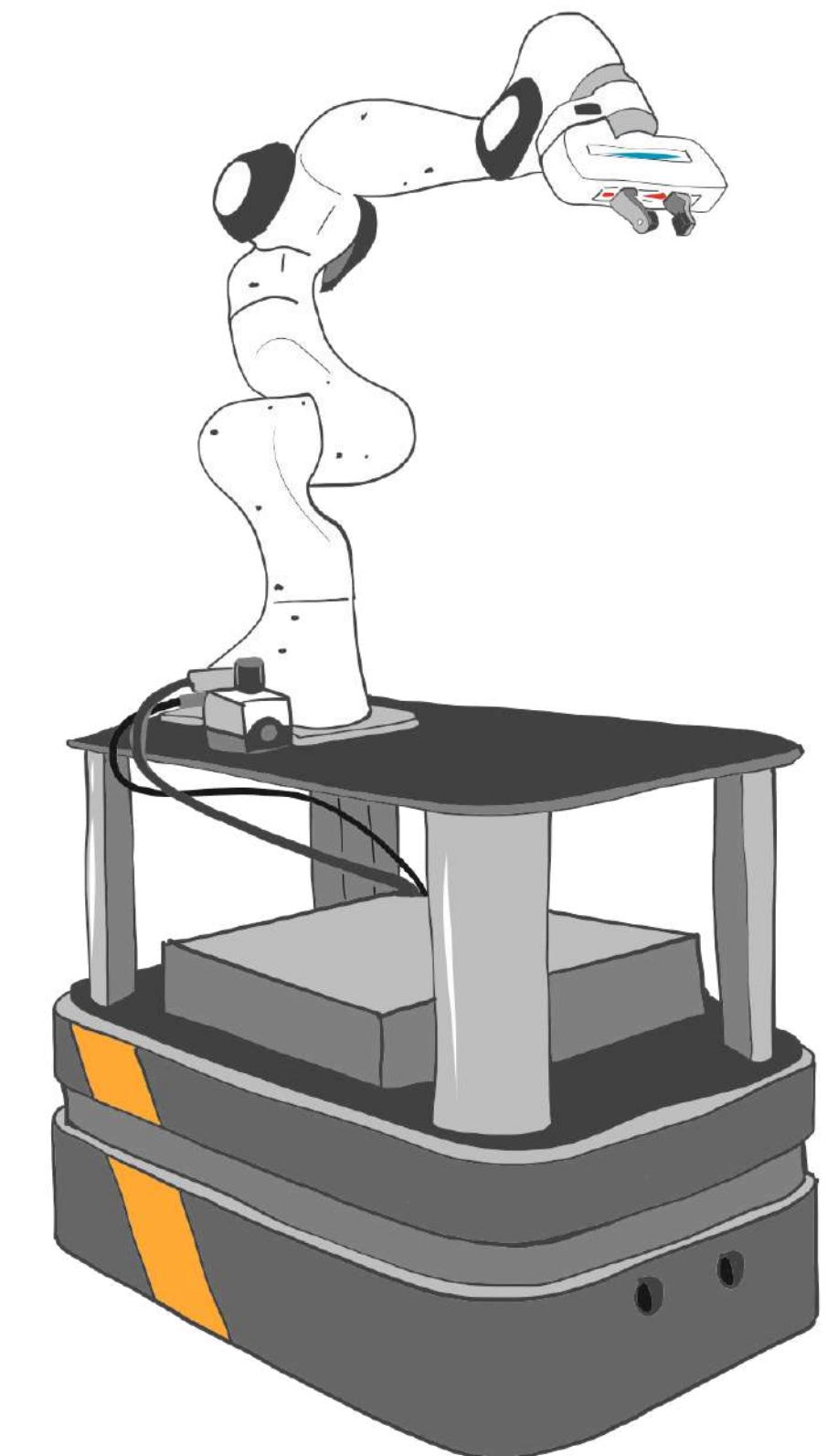
Add and delete effects



Temporal Planning - PDDL 2.1

- Up until now we considered classical planning with **instantaneous actions**
- For things to happen at the same time we need to model **duration** and **concurrency** of actions and events
- Complex inter-dependencies which determine what is physically possible
 - Example:

For a mobile manipulator navigating a store it might be possible to scan the environment while moving, but it might be impossible to attempt to pick something if not still



PDDL 2.1 - Tuple

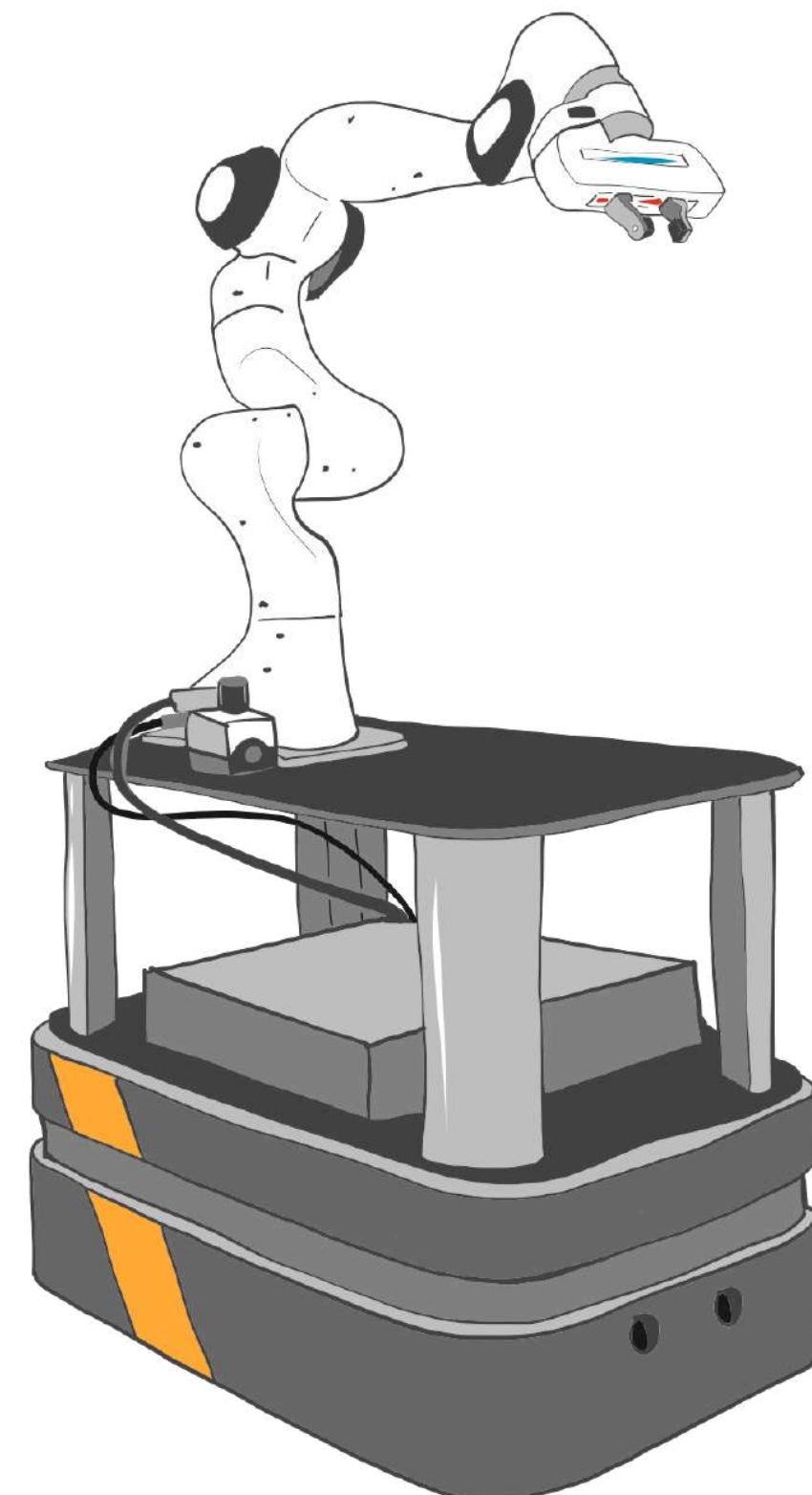
- PDDL 2.1 extends PDDL to more realistic scenarios where actions have durations and adds numeric variables and optimisation
- The temporal planning problem is a tuple: $\langle F, V, A, I, G \rangle$
- where:
 - F is a set of boolean propositions
 - V is a set of real numeric expressions
 - A is a set of deterministic actions which can be durative
 - I is the initial state and G is the goal

PDDL 2.1 - State and solution

- The state is now a tuple $S = (t, s_l, s_n)$:
 - $t \in \mathbb{R}$
 - Logical state $s_l \in F$
 - Numerical state $s_n : V \rightarrow \mathbb{R}_\perp$ (real numbers + *undefined* if not specified)
- Solution to a temporal plan:
 - No longer a simple sequence of actions
 - Timed schedule of (t_i, a_i)

Durative actions

- A durative action consists now of:
 - Action name
 - Parameters
 - Duration
 - Conditions: **at start, over all, at end**
 - Effects: **at start, at end**



PDDL 2.1 - Syntax

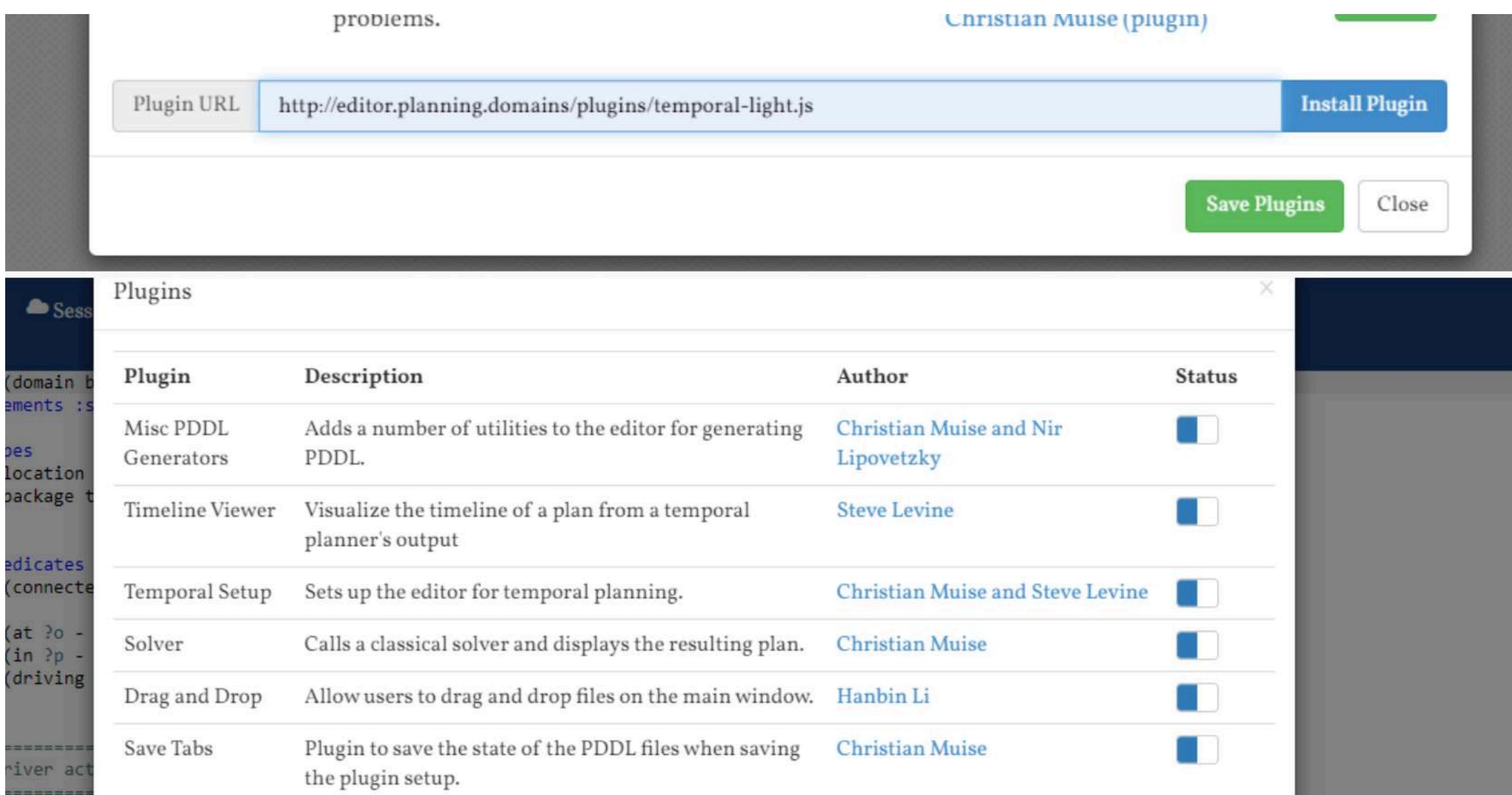
Durative Action

```
(:durative-action zoom_meeting
  :parameters (?s - student
               ?t - teacher)
  :duration (<= ?duration 120)
  :condition (and
               (at start (awake ?s))
               (over all (awake ?t)))
  :effect (and
            (at end (not (awake ?s)))))
```

- What is different from classical PDDL?
 - **:durative-action**
 - **:duration** - expressed as comparison (<, >, =, >=, <=)
 - **:condition** - not preconditions anymore
 - **:effects** - they happen either at start or at end

PDDL 2.1 - Online editor

PDDL Editor supports temporal planning with the right plugins:



http://editor.planning.domains/#read_session=EWjbgnhuUd

Exercise (20 min)

Imagine you are in charge of the stage lighting instruments to illuminate important live performance at international concerts. Your task for a particular venue in Moscow is to make sure that all 5 available stage lights are ON whenever the artist enters the stage. A colleague of yours provided you with the following PDDL files to solve the problem for a generic venue with 5 lights:

- PDDL Editor template: http://editor.planning.domains/#edit_session=LKC7RKc7VahSXgm

Unfortunately, the electrical system in Moscow was not installed properly and you cannot independently switch ON and OFF single lights as you please. In particular, a light can be switched ON if and only if there is already another adjacent light ON. Your task is to modify the given domain.pddl and problem.pddl templates and generate a plan to turn ON all the lights respecting the constraints. The initial state is the following:

- Initial state: {off, on, off, off, on}

ROSPlan

A ROS module to integrate planning into robotics for plan execution.

Planning in Robotics

Planning can be pretty abstract

- How can a plan be used to perform a task in the real world?

So far

- Platform
- Problem and Domain
- Generate a solution

Missing

- Update and maintain the world model
- Process and dispatch the plan



Planning in Robotics

Using task planning in robotics might be tedious

- A lot of different things need to be integrated:
 - State estimation - build and update a symbolic representation of the world
 - Domain and problem definition for the task
 - Generate the plan
 - Execute the plan according to runtime data

ROSPlan facilitates this integration

ROSPlan - Overview

ROSPlan integrates standard tools:

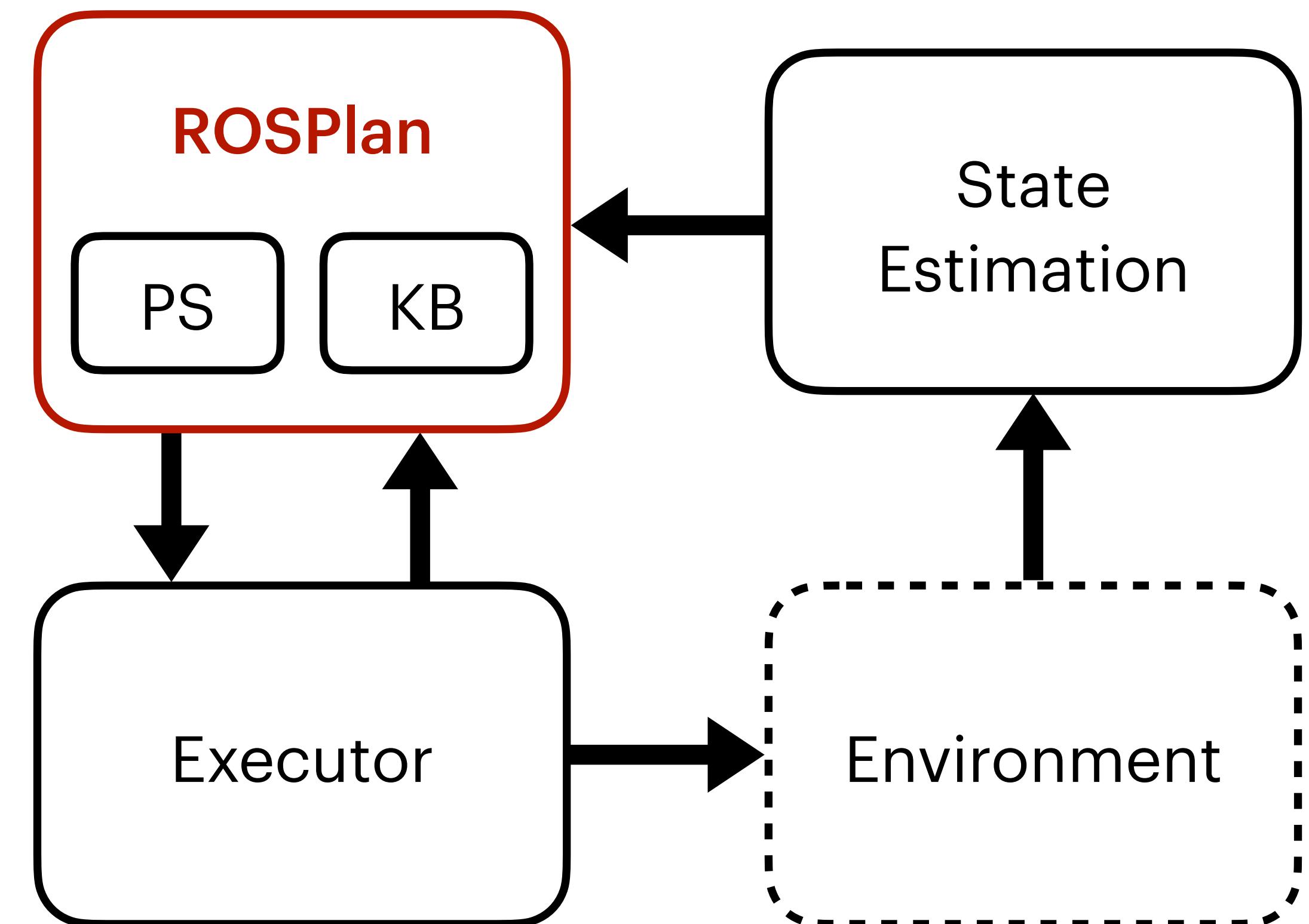
- Planning with PDDL 2.1
- ROS (Robot Operating System)

ROSPlan takes care of:

- Build the initial state from the PDDL problem
- Generate and validate the plan
- Dispatch the plan

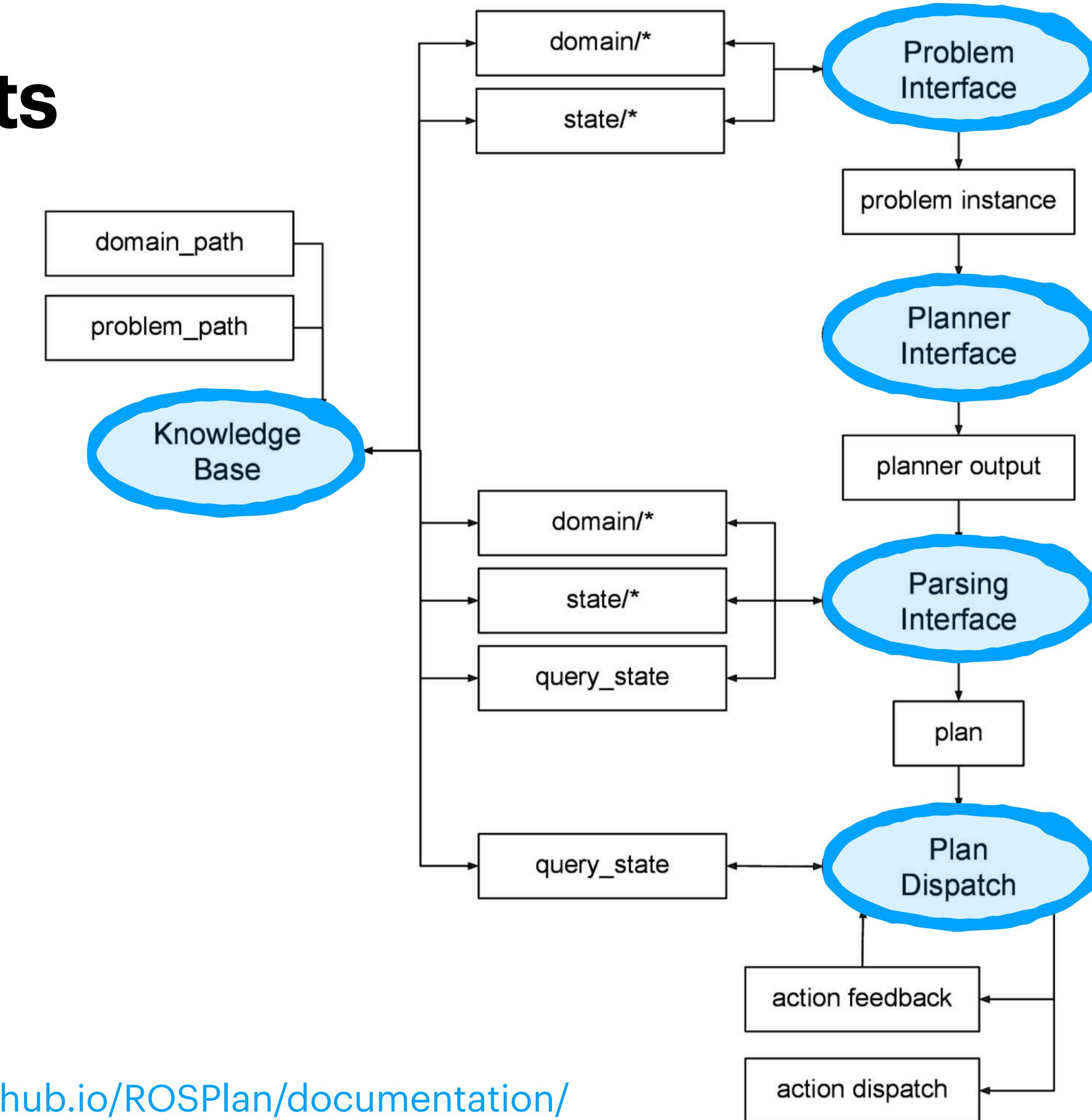
ROSPlan is domain/platform generic

PS: planning system
KB: knowledge base



ROSPlan - Components

- The Knowledge Base
 - The Problem Interface
 - The Planner Interface
 - The Parsing Interface
 - The Plan Dispatch



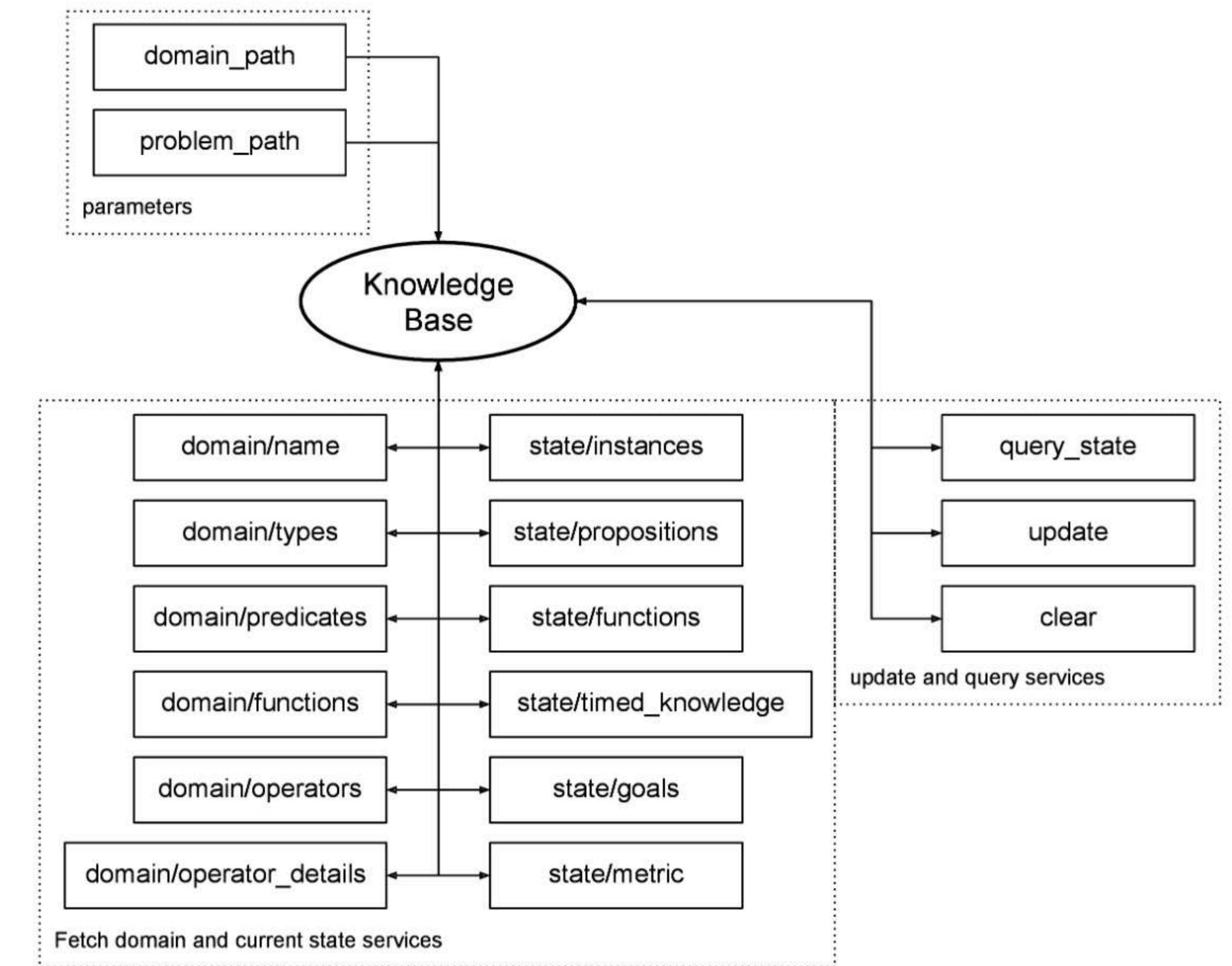
Resources

<https://kcl.ac.uk>

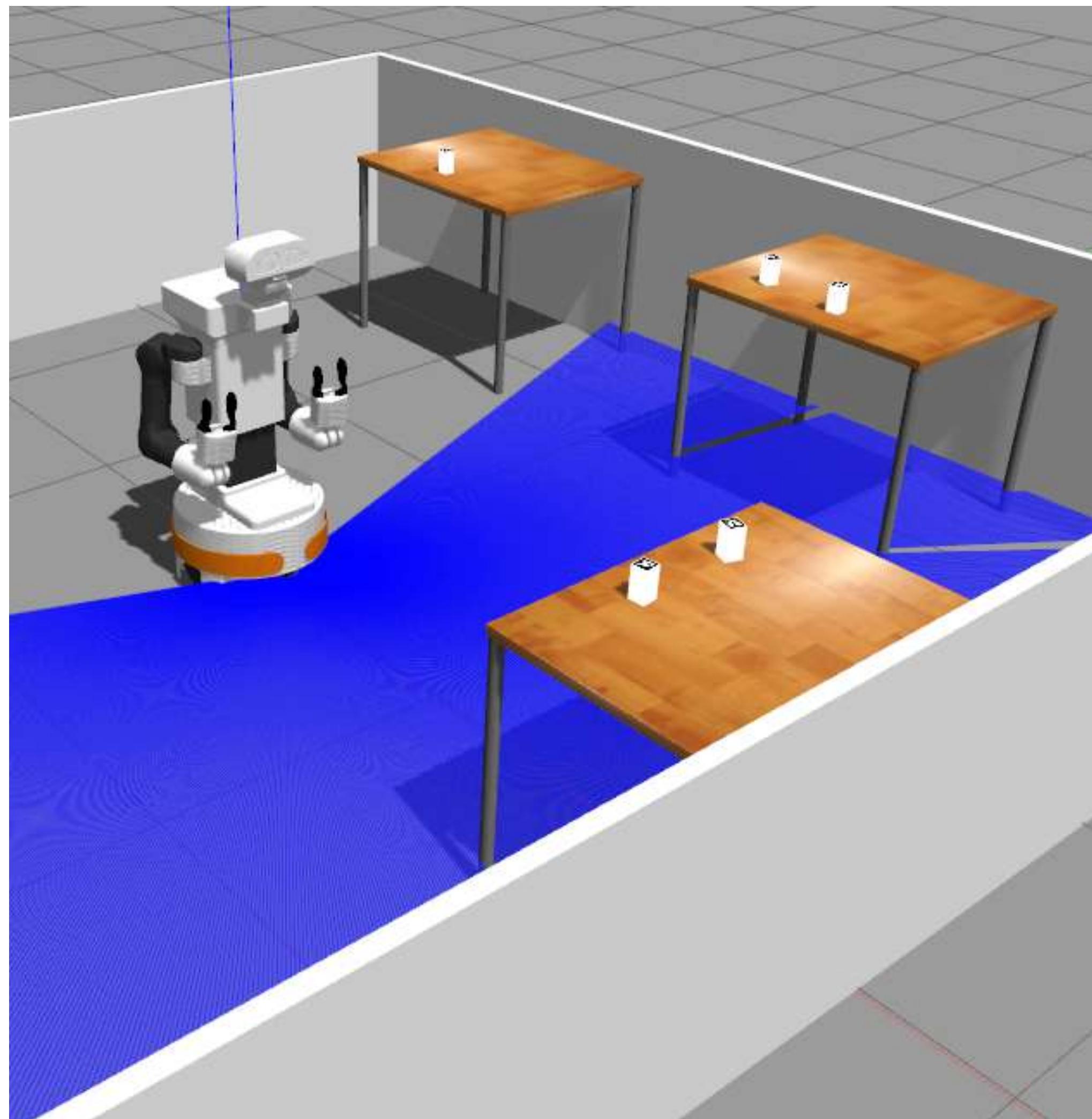
ROSPlan - Knowledge Base

The Knowledge Base:

- Loads a PDDL domain and problem from .pddl files
- Stores the state as a PDDL instance
- Is updated by ROS messages
- Can be queried



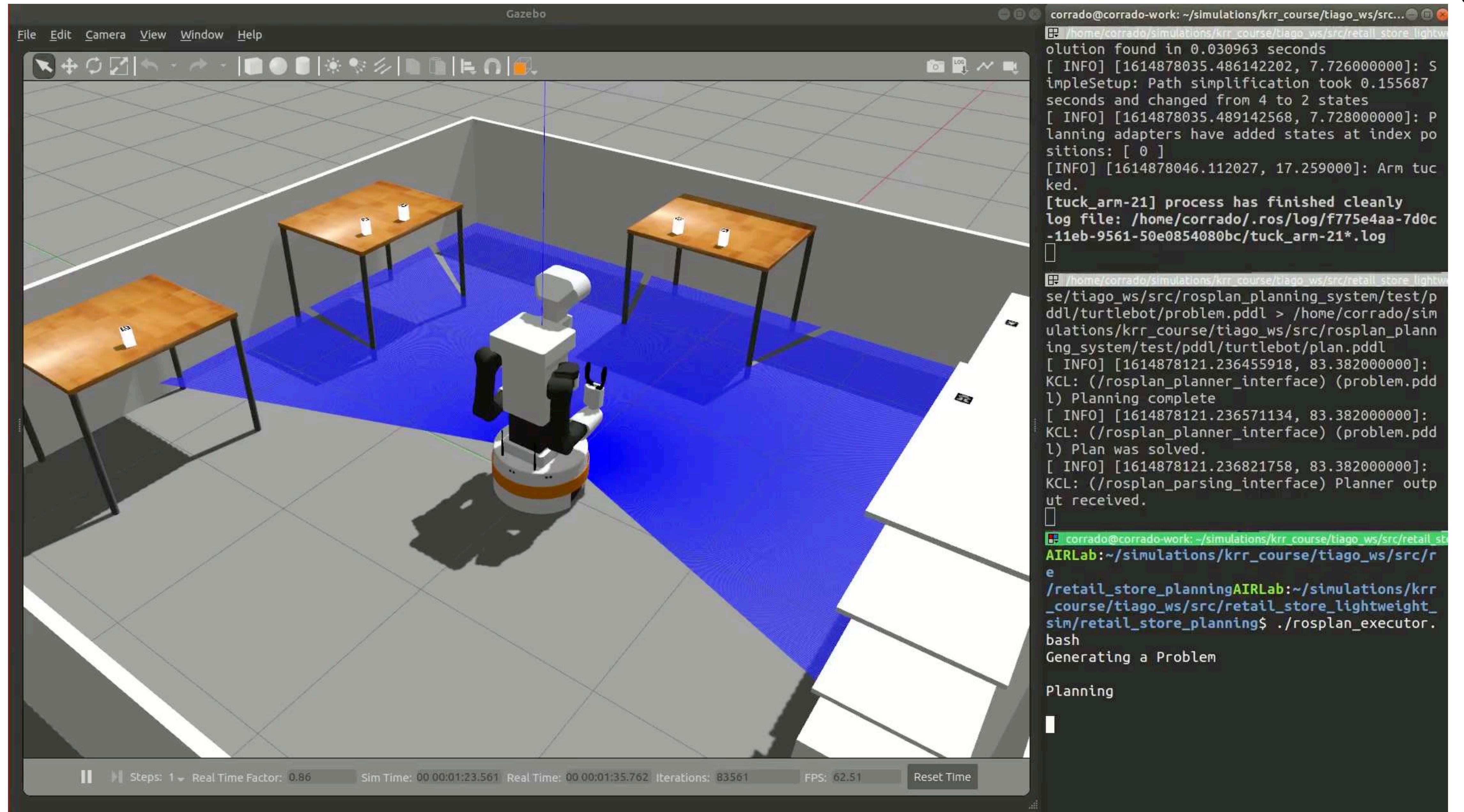
ROSPlan - Example



Pick some object from
somewhere

Domain file
domain.pddl

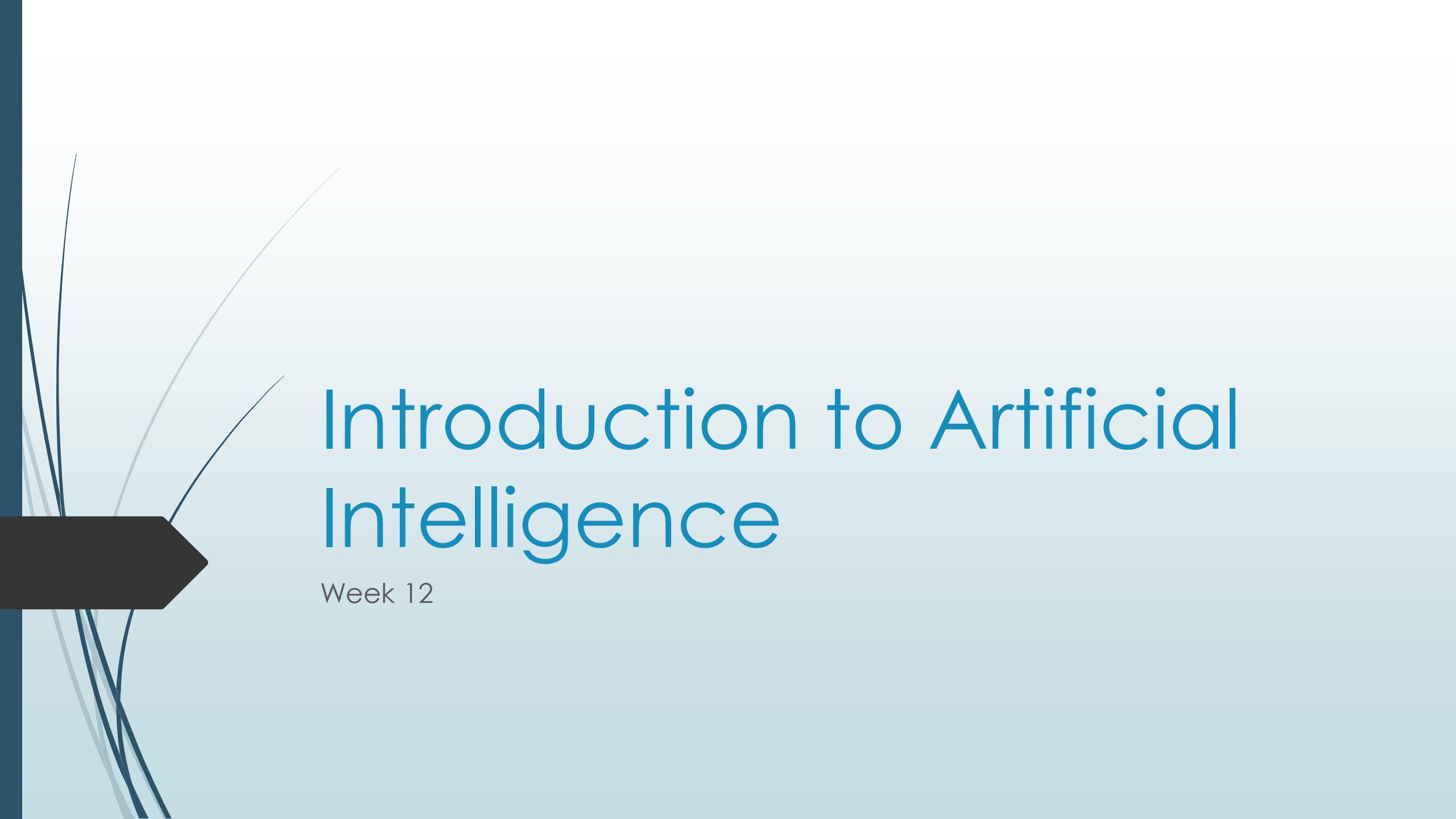
Problem file
problem.pddl



DEMO

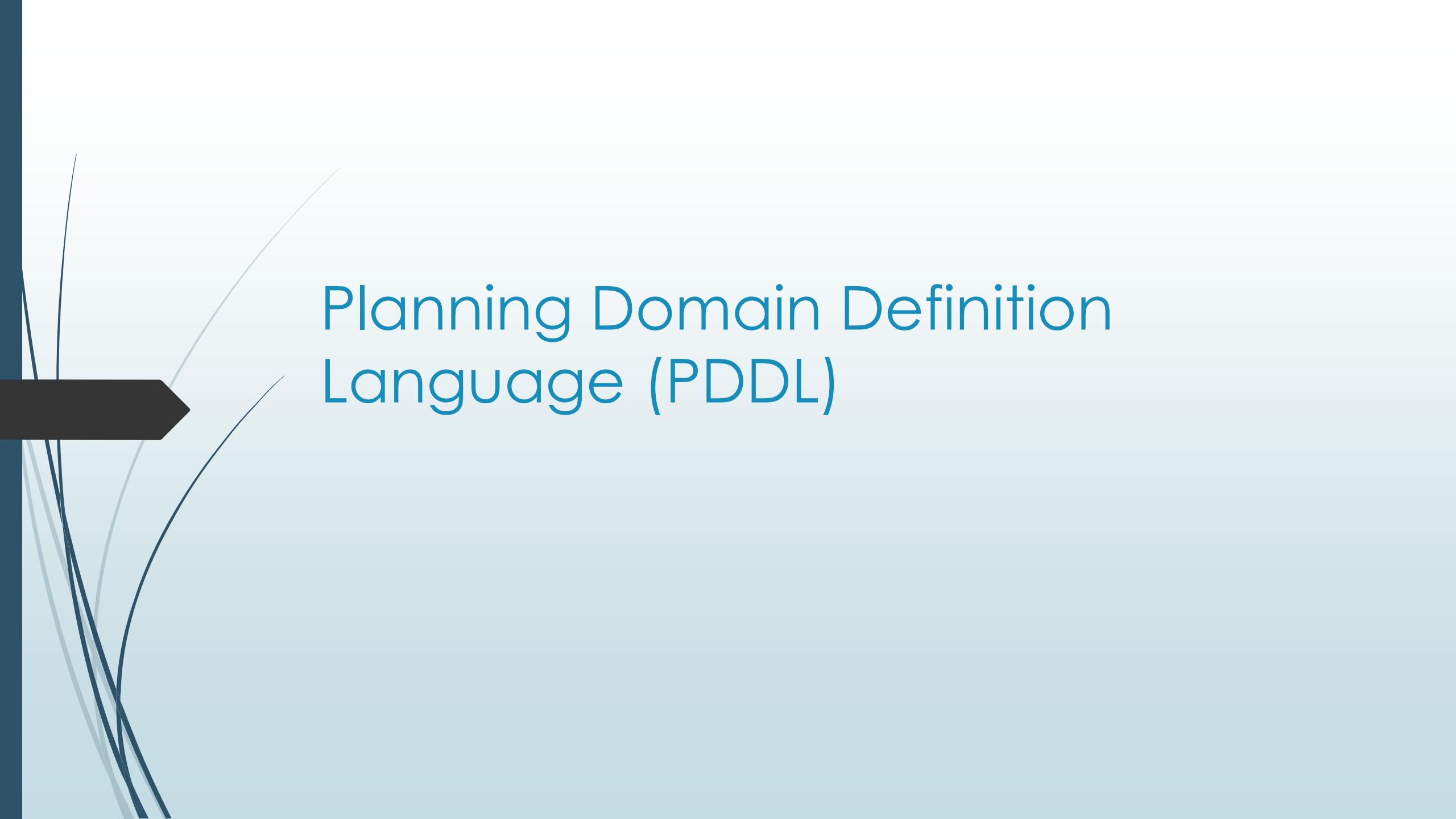
Solution for Robotic gripper manipulating boxes: http://editor.planning.domains/#edit_session=6GXQPcJRtBZWJN

Visualisation: <https://blocks-dot-ai-planning.appspot.com/>



Introduction to Artificial Intelligence

Week 12



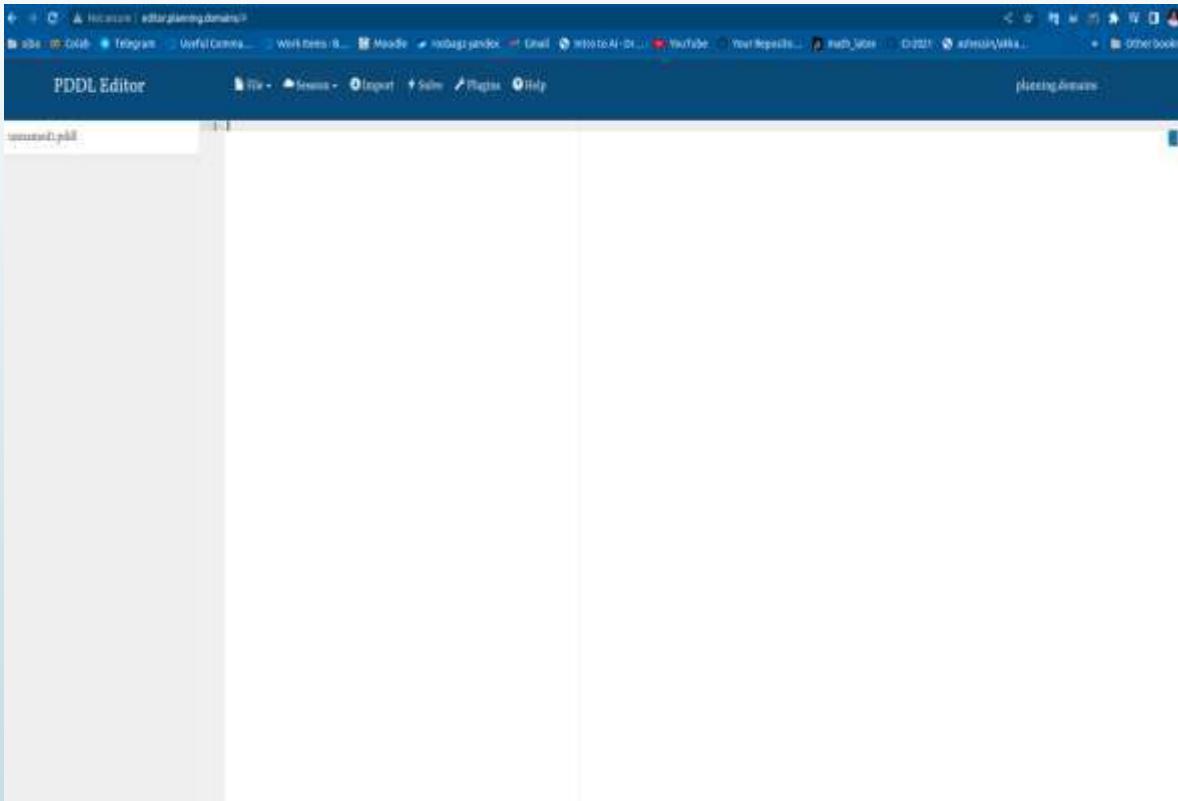
Planning Domain Definition Language (PDDL)

Introduction

- As planning has evolved, so has the language used to describe it
- Planning Domain Definition Language (PDDL) is a family of languages which allow us to define a planning problem
- PDDL one of the few languages designed for the purpose of creating a standard for Artificial Intelligence (AI) planning

Environment Setup

- We will use [PDDL Online Editor](#)

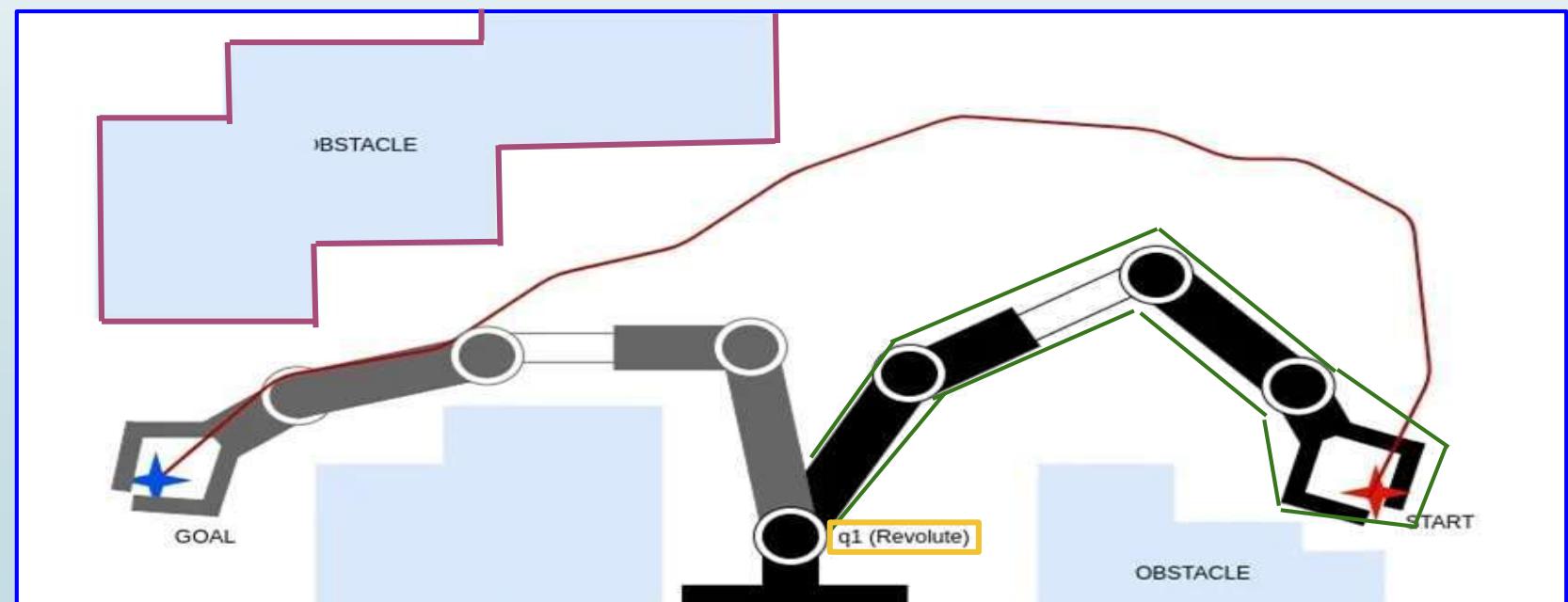


SCAN ME



Components of PDDL

- A **world** in PDDL is described by a set of states each containing a list of facts and/or objects
- A world begins with an **initial state**
- A world governed by a set of rules and **constraints** that limit which actions can be taken in each state
- Each **action** generally represents a **transition** to a different state



Components of PDDL

- There are certain things we need to keep track of in the “world”:
 - **Objects:** Things in the world that interest us
 - **Predicates:** Facts that we are interested in (e.g. properties of objects), which can be true or false
 - **Actions/Operators:** Ways of changing the state of the world, i.e. things that happen that change the facts
 - **An initial state:** The state of the world that we start in, i.e. things that are true at the start
 - **Goal specification:** The state of the world we want to end at, i.e. things that we want to be true at the end

PDDL Syntax

- First thing you need to know, PDDL files usually have the extension **.pddl**
- There are two PDDL files you need to learn the syntax of:
 - The Domain File (**domain.pddl**)
 - establishes the context of the world
 - determines what sorts of details the states can include (predicates)
 - what can we do to move between states in the world (actions).
 - The Problem File (**problem.pddl**)
 - represents an instance of the world we established in the domain
 - determines what is true at the start of the plan (initial state)
 - what we want to be true at the end of the plan (goal state).

PDDL Syntax

- The Domain File structure (`domain.pddl`)

```
(define (domain <domain name>)

  (:predicates
    <predicate-list>
  )

  ; optional block
  (:types
    <types-list>
  )

  (:action
    <action-details>
  )

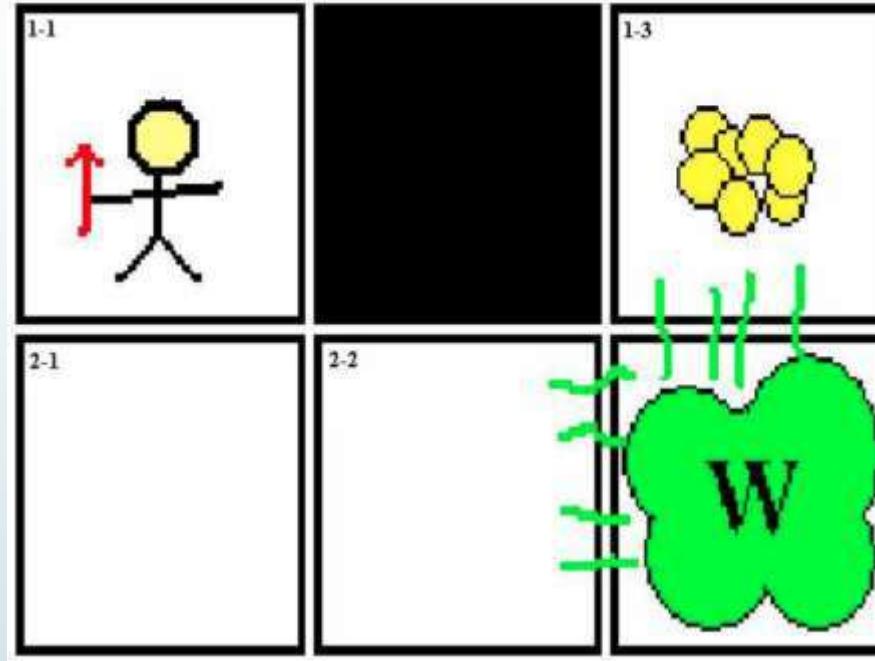
)
```

PDDL Syntax

- The Problem File structure (`problem.pddl`)

```
(define (problem <title>)
  (:domain <domain-name>)
  (:objects
    <object-list>
  )
  (:init
    <predicates>
  )
  (:goal
    <predicates>
  )
)
```

Example (WUMPUS WORLD)



Environment

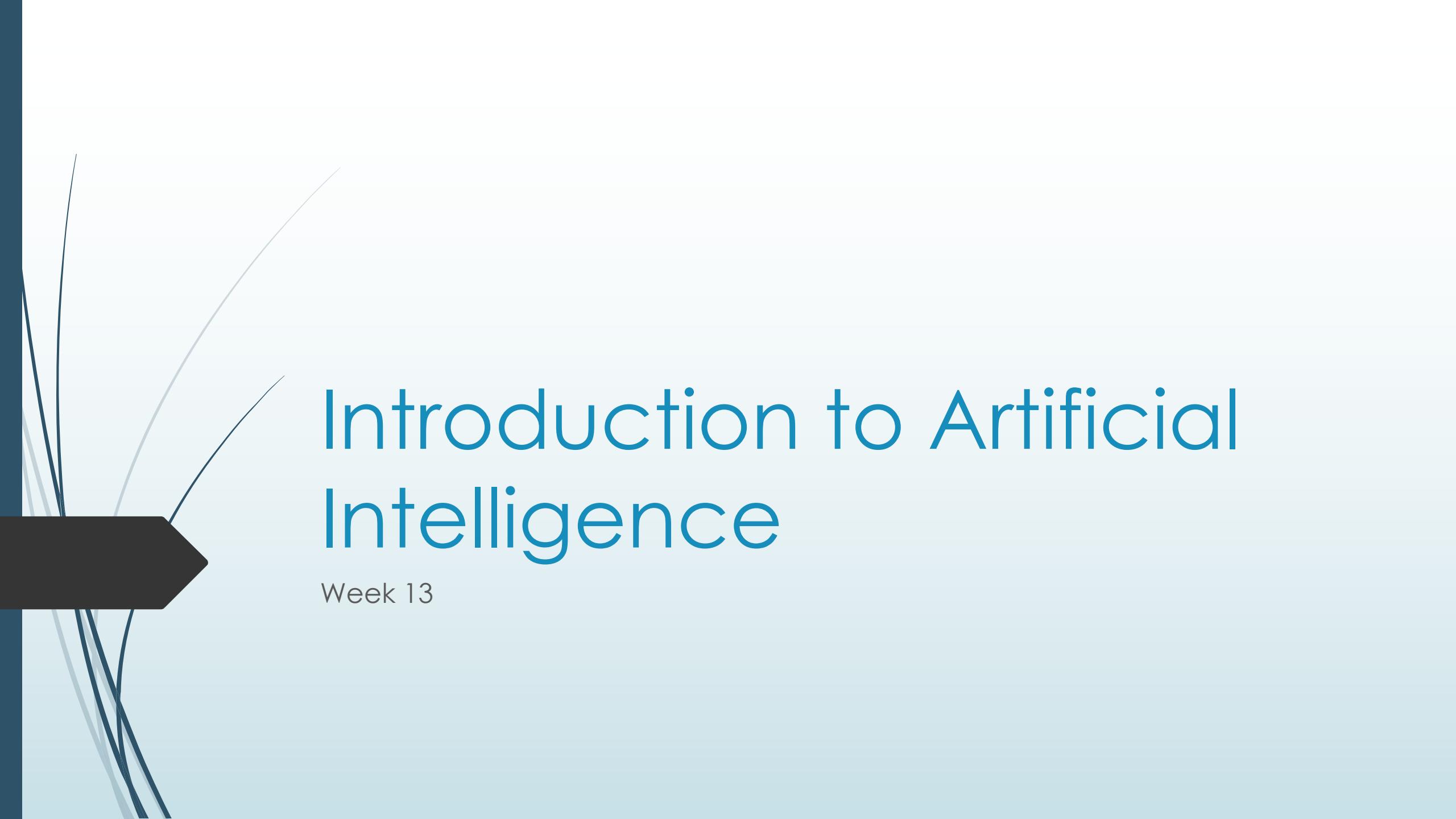
- sq 1-1 is agent
- sq 1-2 is pit
- sq 1-3 is gold
- sq 2-3 contain wumpus

Actions

- Move from sq to adj sq
- Kill Wumpus with arrow
- Take Gold

Goal

- Get gold & Return back to start pos



Introduction to Artificial Intelligence

Week 13

A decorative graphic on the left side of the slide features several thin, light-colored lines that intersect and curve across the frame. A solid dark grey arrow points from the bottom-left towards the center. Inside this arrow is a white number '2'.

2

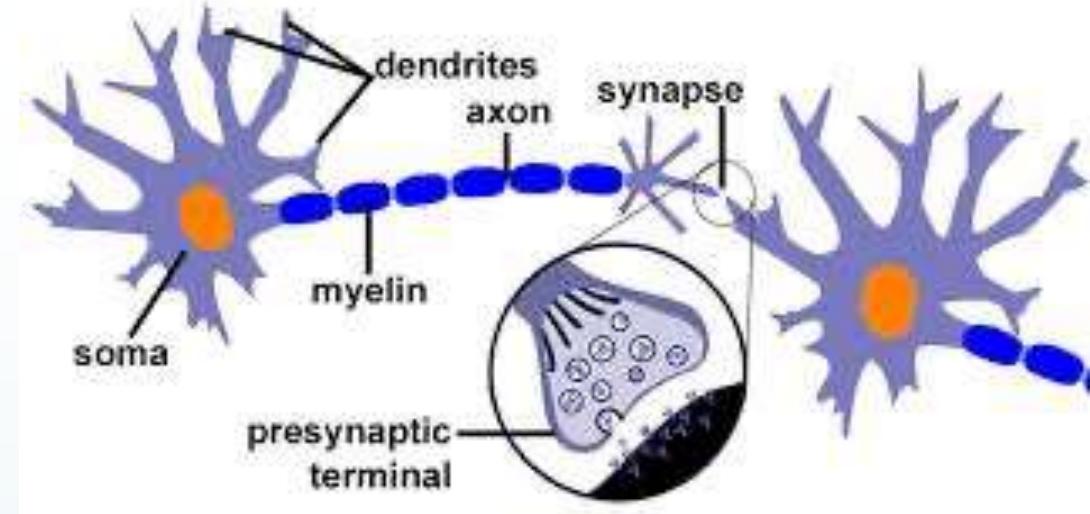
Artificial Neural Networks

Artificial Neural Networks

- ▶ Again – biological inspiration
 - ▶ Neurons in the brain
 - ▶ First models from 1957
 - ▶ F. Rosenblatt on contract to US Navy
 - ▶ Intended to be hardware machines
 - ▶ Implemented in software to prototype
 - ▶ Now seen in both software and hardware forms
 - ▶ “the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence” – NY Times
 - ▶ 1969 saw the death of many of these connectiveness models (Minsky-Perceptrons)
 - ▶ XOR could not be developed via such a model
 - ▶ Research funder saw the dream of AI to be a farce and pulled back money
 - ▶ AI Winter
 - ▶ Funding now returning – we are in an “AI Spring”?

Biology

- ▶ Neuron Cell
- ▶ Information
 - ▶ Processing
 - ▶ Message Passing
 - ▶ Input from the dendrites
 - ▶ Output from the synapse
- ▶ Electrical message passing model
 - ▶ Spike models from 1907 have been investigated



ANN

- Specified by:
 - Architecture
 - How are the connections made
 - What Neurons are connected to others
 - Feedforward
 - Recurrent
 - Neuron Model
 - What functions can the neurons take as inputs
 - Can restrict the learning algorithm (backpropagation requires the functions to be differentiable)
 - Learning Algorithm
 - Connected Neurons have weights in their connections
 - How do we make the updates to weights?
 - Backpropagation of errors
 - Evolutionary methods
 - Supervised or Unsupervised

(Un)Supervised Learning

Supervised

- ▶ Tasks with outcomes
- ▶ Tasks
 - ▶ Classification

Unsupervised

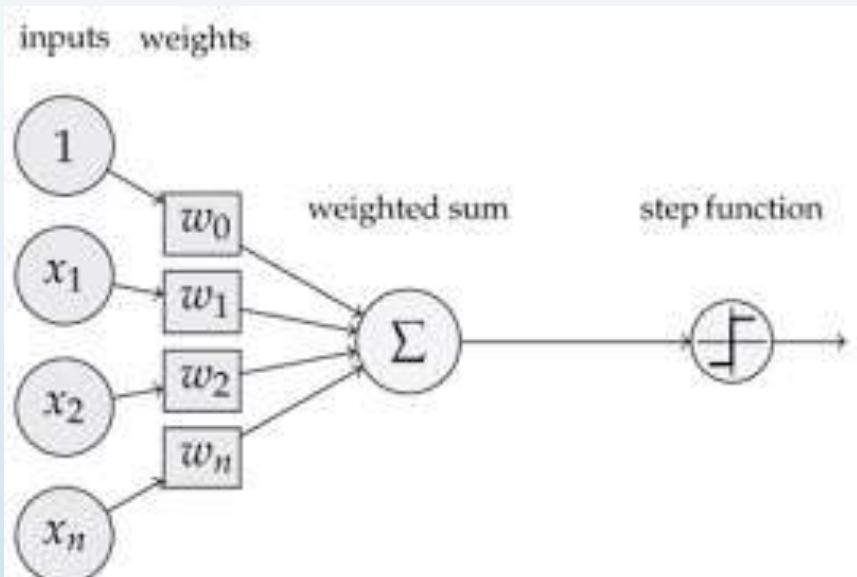
- ▶ Task inputs with no expected output
- ▶ Tasks
 - ▶ Clustering and Data understanding

ANN Usage Fields

- ▶ Speech recognition
- ▶ Computer vision
- ▶ Music, art generation
- ▶ Classification
- ▶ Non-linear tasks
- ▶ etc.

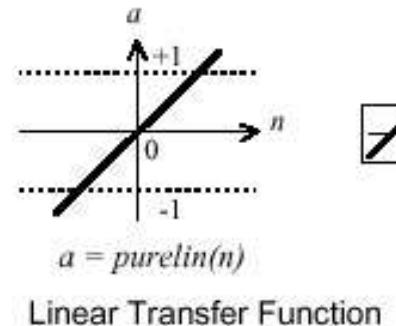
Perceptron

- ▶ Single layer feedforward networks
- ▶ Simplest version of a neural model
- ▶ $output = \text{step}_0(\sum w_i x_i)$, where $\text{step}_0(m) = 1 \text{ if } m > 0 \text{ and } 0 \text{ otherwise}$

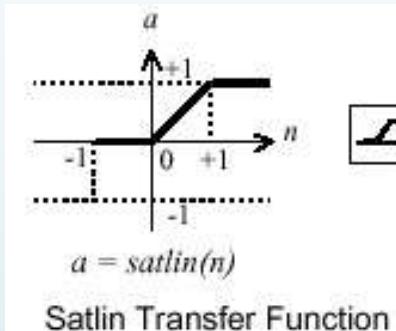


Activation Functions

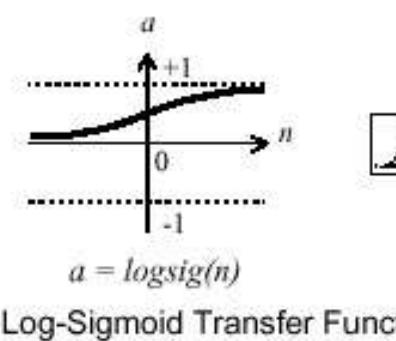
- ▶ Generally non-linear
 - ▶ Otherwise output is input
- ▶ Required to be differentiable in backpropagation training
 - ▶ Uses a calculation of error
 - ▶ Needs to get the integration of the differentiation to find error



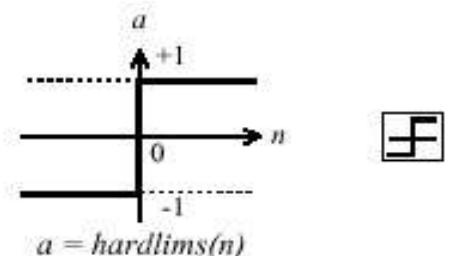
Linear Transfer Function



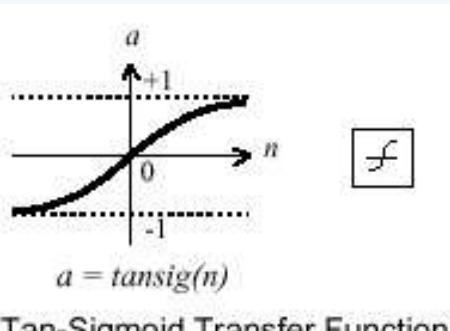
Satlin Transfer Function



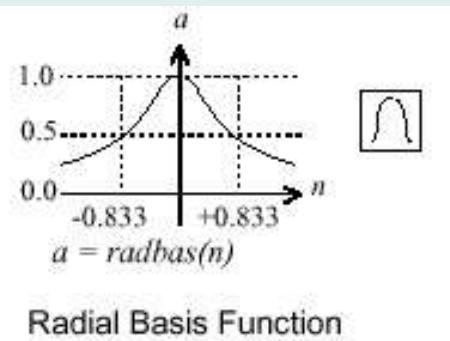
Log-Sigmoid Transfer Function



Symmetric Hard Limit Trans. Funct.

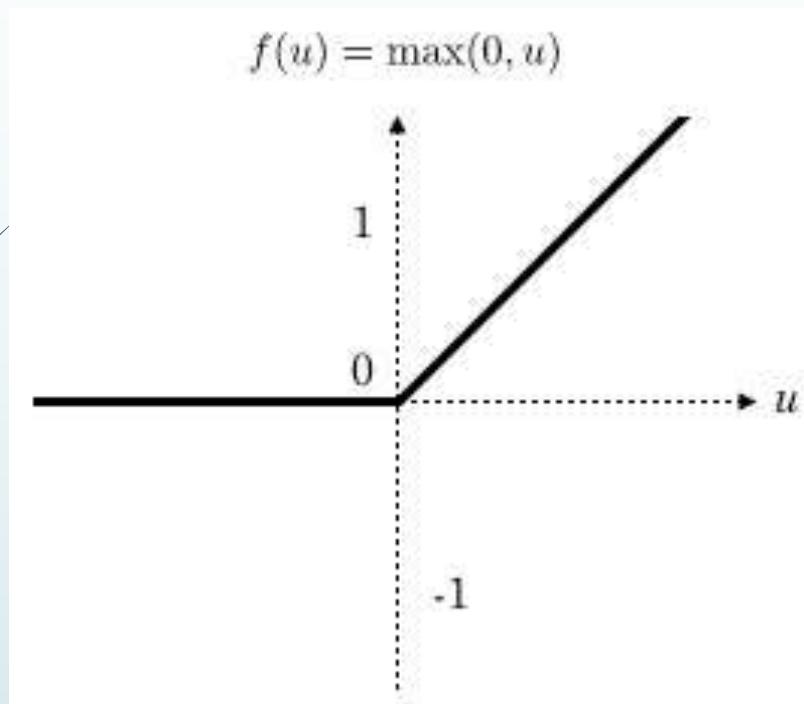


Tan-Sigmoid Transfer Function



Radial Basis Function

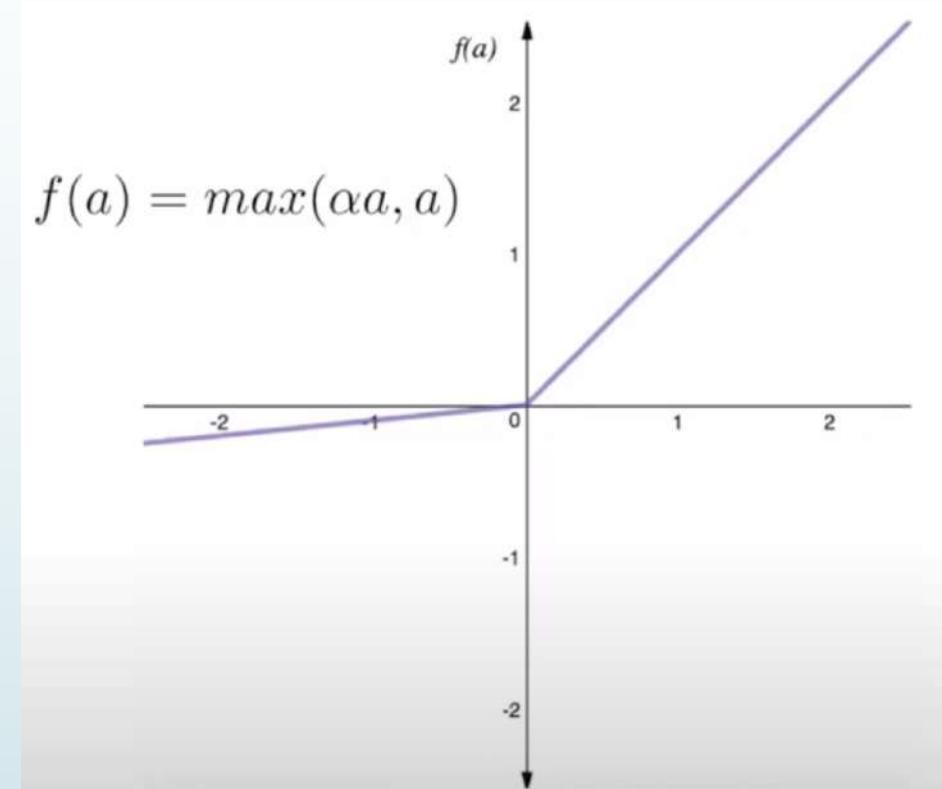
Rectified Linear Unit (ReLU)



- Where u is an input to neuron
- One of the most popular activation functions

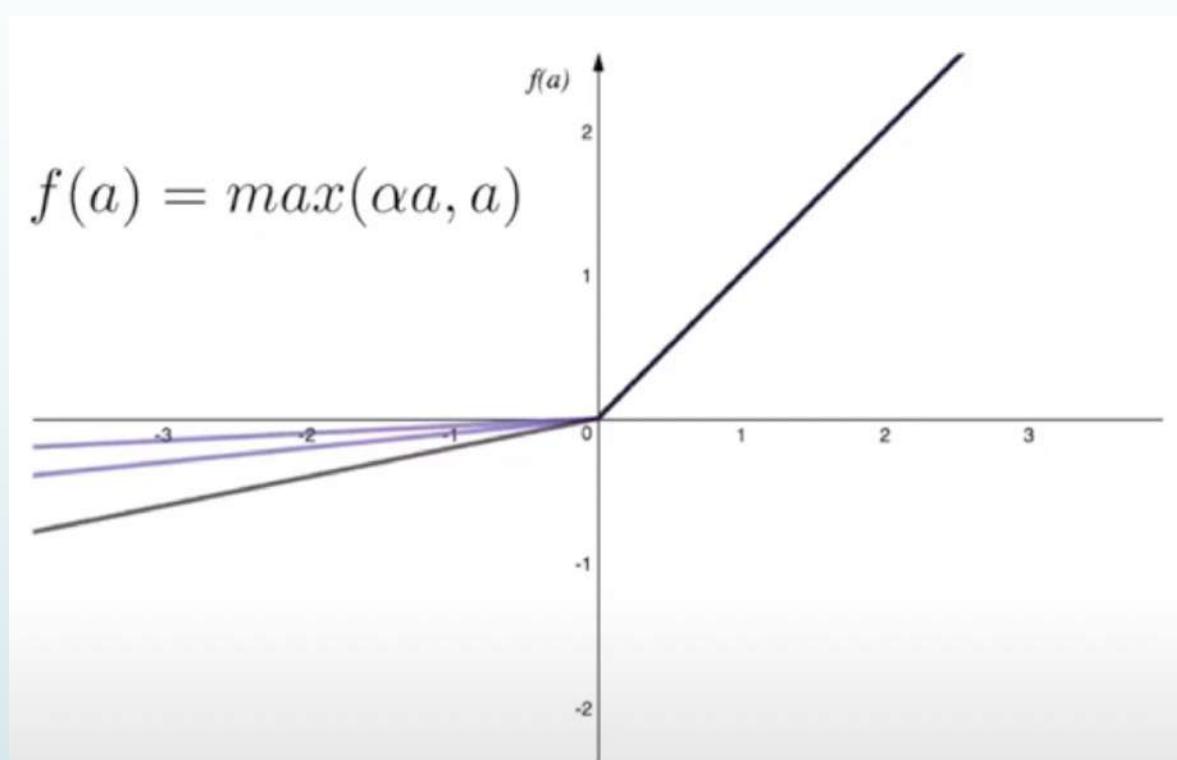
Leaky ReLU

- More advanced ReLU
- α is a hyperparameter chosen and fixed initially
 - Usually, small value

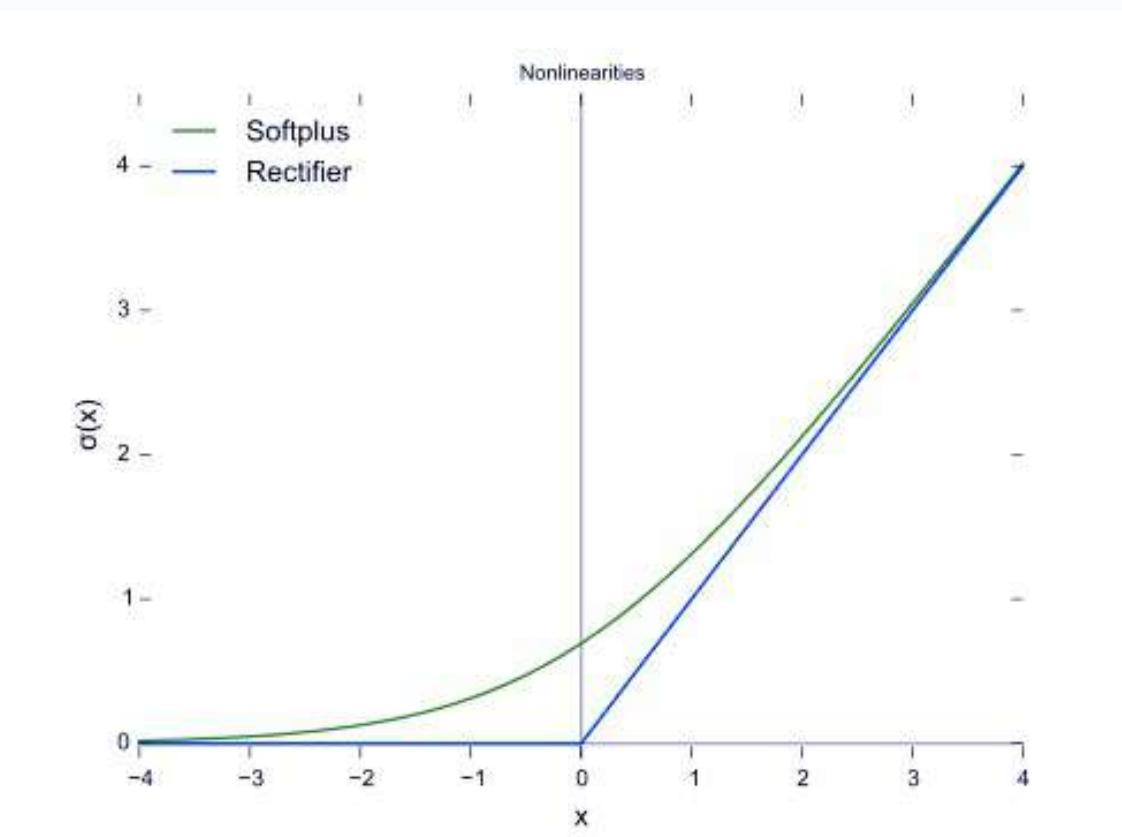


Parametric ReLU

- » Advanced Leaky ReLU
- » α parameter can be trained

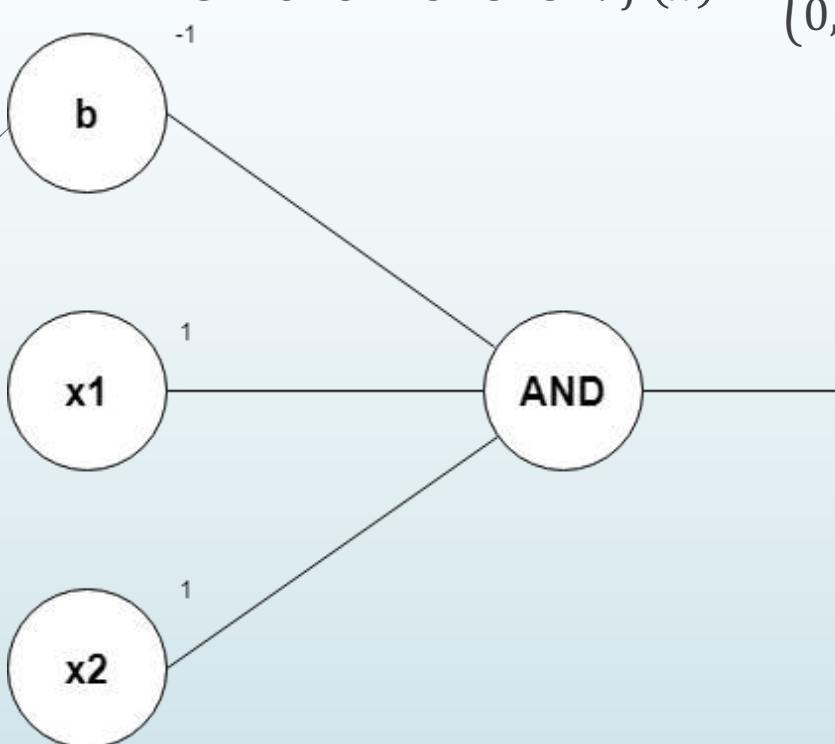


Softplus



- Softplus is an approximation to a ReLU which is $f(u) = \ln(1+e^u)$

AND Perceptron

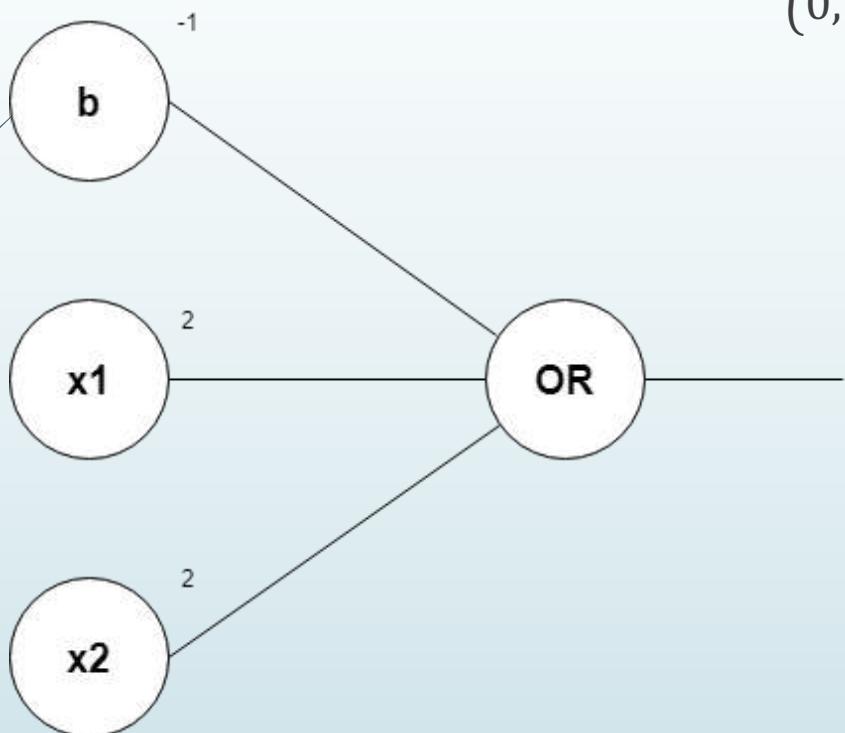


Innopolis University - Introduction to AI 2023 Fall

► Activation function: $f(x) = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases}$

w_0	w_1	w_2	x_1	x_2	$w_i x_i + b$	Activation
-1	1	1	0	0	-1	0
-1	1	1	0	1	0	0
-1	1	1	1	0	0	0
-1	1	1	1	1	1	1

OR Perceptron



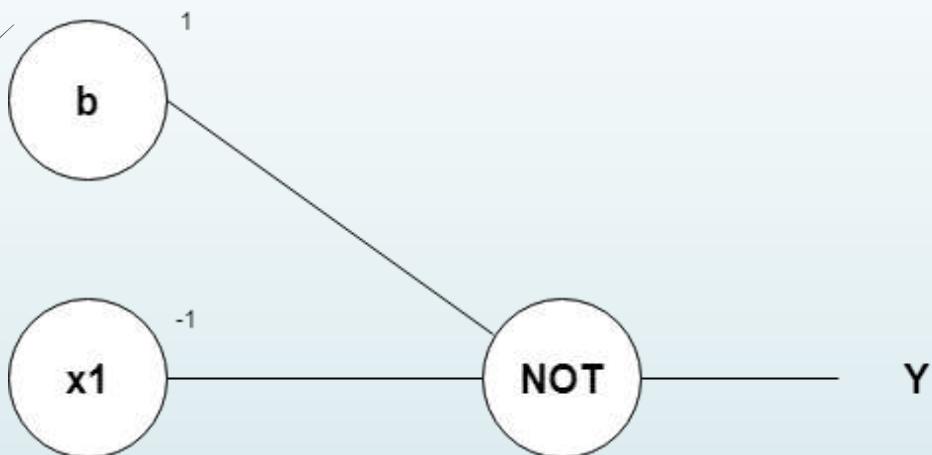
Innopolis University - Introduction to AI 2023 Fall

► Activation function: $f(x) = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases}$

w₀	w₁	w₂	x₁	x₂	w_ix_i+b	Activation
-1	2	2	0	0	-1	0
-1	2	2	0	1	1	1
-1	2	2	1	0	1	1
-1	2	2	1	1	3	1

NOT Perceptron

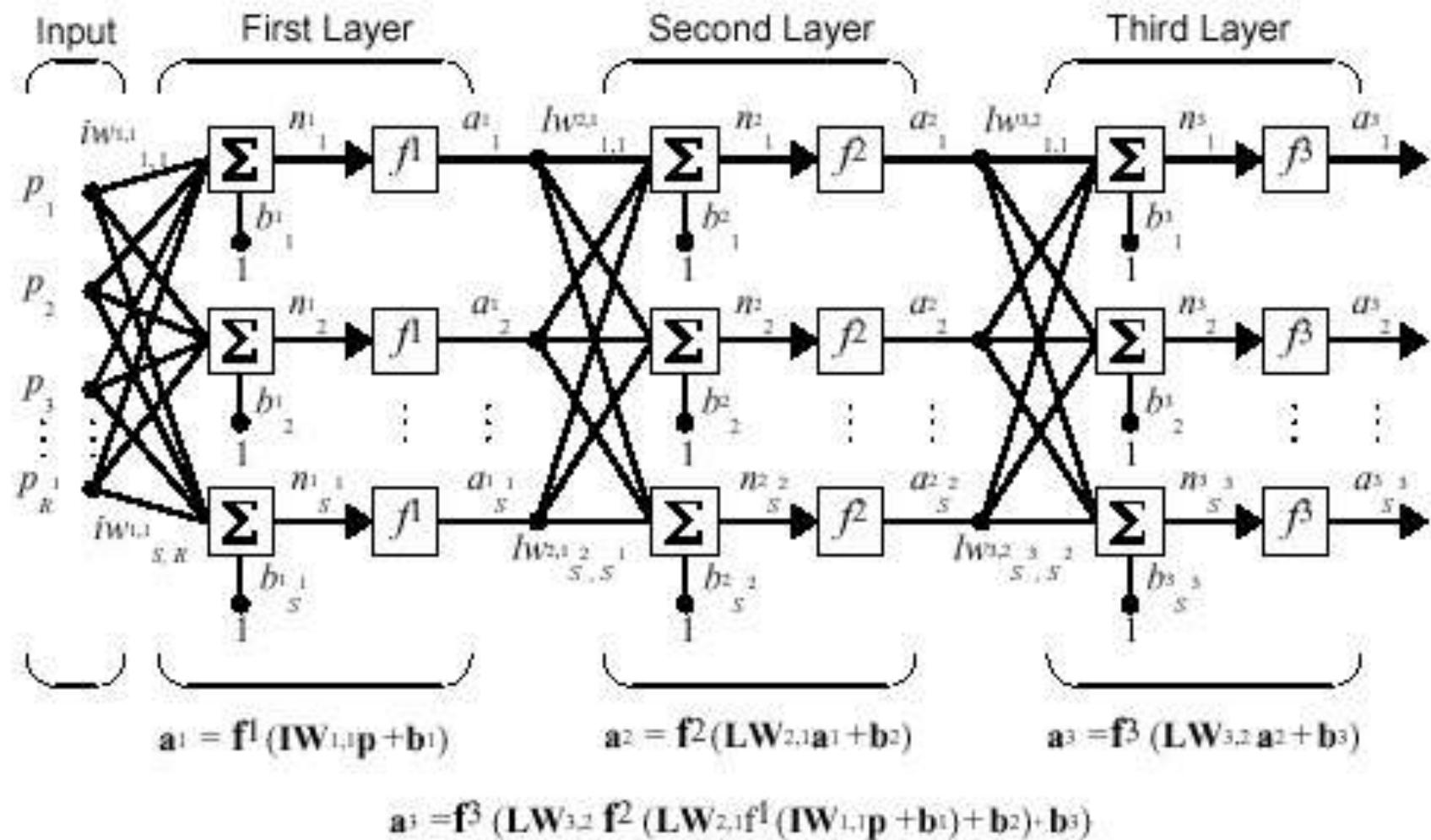
► Activation function: $f(x) = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases}$



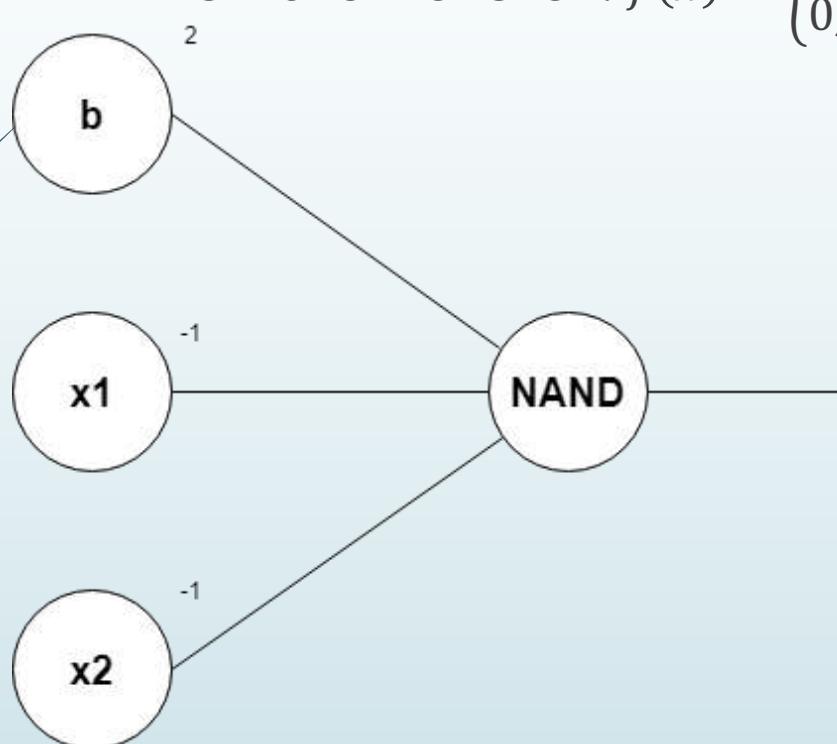
w ₀	w ₁	x ₁	w _i x _i +b	Activation
1	-1	0	1	1
1	-1	1	0	0

Y

Feedforward Network



NAND Perceptron



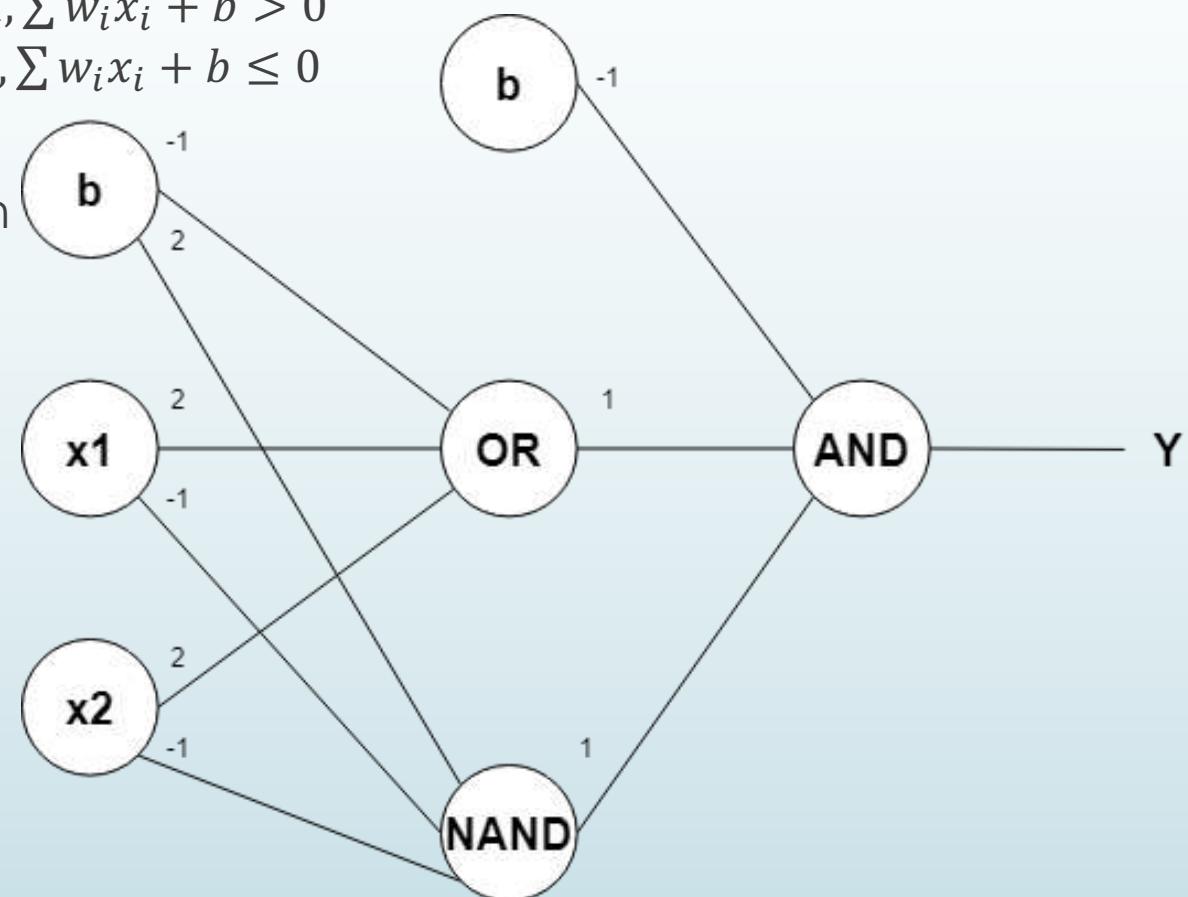
Innopolis University - Introduction to AI 2023 Fall

► Activation function: $f(x) = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases}$

w₀	w₁	w₂	x₁	x₂	w_ix_i+b	Activation
2	-1	-1	0	0	2	1
2	-1	-1	0	1	1	1
2	-1	-1	1	0	1	1
2	-1	-1	1	1	0	0

XOR Perceptron

- ▶ Activation function: $f(x) = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases}$
- ▶ Not each task can be solved by a single perceptron
- ▶ XOR is one of those tasks



Backpropagation on a Network

- ▶ Single layer can only deal with linearly separable problems
- ▶ To deal with multiple dimensions the network needs to expand in the number of layers
 - ▶ Increases the dimensionality; everything in a high dimensional space BECOMES linearly separable
- ▶ As this expands larger and larger the type of problems which can be faced expands
 - ▶ Deep Learning
 - ▶ Beware of Overtraining

Basic Training Algorithm

- ▶ Initialize the weights in the network (often randomly)
- ▶ **repeat**
 - ▶ **for** each example e in the training set **do**
 - ▶ $O = \text{neural-net-output}(\text{network}, e); // \text{forward pass}$
 - ▶ $T = \text{teacher output for } e;$
 - ▶ Calculate error $(T - O)$ at the output units;
 - ▶ Compute $w_j = w_j + \alpha * Err * I_j$ for all weights from hidden layer to output layer; $// \text{backward pass}$
 - ▶ Compute $w_j = w_j + \alpha * Err * I_j$ for all weights from input layer to hidden layer;
 - ▶ Update the weights in the network to the new weights;
 - ▶ **End for**
 - ▶ **until** all examples classified correctly or stopping criterion met
 - ▶ **return**(network)

Calculating Error in Backpropagation

- Each node is considered to be part of the error in the overall function based on its weight in the input

$$\text{error of hidden node } j = \sum_{i \in \text{outputs}} w_{ij} \delta_i$$

where δ_i is the error at output node i .

Issues with Backpropagation Networks

- ▶ Black Box (all ANN)
 - ▶ How did you get the answer – can we form any rules?
 - ▶ I had the program run and this thing was made
 - ▶ Have to change it into a lookup table and figure out why it has those weights
 - ▶ Large training samples needed for backpropagation to work effectively

Other Trainings and Connection Methods

- ▶ Neural Evolutionary Models
 - ▶ Have a GA train the network – even allow the GA to assemble the network
 - ▶ NeuroEvolution of Augmenting Topologies (NEAT)
- ▶ Hopfield Networks
 - ▶ Train much like perceptron
 - ▶ Not connected in a linear manner

Neural Evolution

- ▶ Treat the weights as being the elements of the chromosome
- ▶ The fitness function is the amount of error
- ▶ This allows for the use of non-differentiable functions in the elements
 - ▶ We do not need to rationalize the error caused by each node
 - ▶ This allows for problems without clear gradients to be attacked with neural methods
- ▶ Can also evolve the structure and connections within the network and build new infrastructures
 - ▶ Information propagation

Deep Learning

- ▶ Not a new topic – the biggest developments were done in the 1970s
 - ▶ Issue – AI Winter
 - ▶ Issue – Lack of High Performance Computing (HPC) infrastructures
- ▶ Removal of these problems has lead to a boom in Deep Learning Research
 - ▶ AI Winter – funding in AI is returning and expanding
 - ▶ HPC is now affordable to even start-ups for reasonable costs
 - ▶ Cloud Computing
 - ▶ Funding from government interests
 - ▶ Moves by Google/Amazon etc.

Attacking the Unassailable

- ▶ GO
 - ▶ If chess is a hard problem, then GO is an improbable problem
 - ▶ Chess n-focation of tree bounded by at most all moves that can be made by 16 pieces – each piece between 3 (pawn on non-home square) and 4*7 moves allowed to it (queen able to visit all other squares in a rank, file, or two diagonals), worst case upper bound is $16 \times 4 \times 7$ open moves on a turn (note very generous overage assuming all pawns are now queens)
 - ▶ In GO, upper bound of moves is to place a stone in one of a 64 by 64 board, 64×64 nodes to visit
 - ▶ How?

Research into GO

- ▶ Lots of work has been done on the problem of GO due to its complexity
- ▶ Full special sessions and Journal articles have been dedicated to advancements in play
- ▶ Many of these look at higher level structures rather than individual plays
 - ▶ Eyes (places you cannot put an enemy stone as it would be captured)
 - ▶ Board edges (Again building from the uncapturable nature of these squares)

Why is GO so important – it is only a game!

- ▶ Games provide interesting surrogate problems for real world issues
- ▶ GO is a game of structures and a game with huge number of potential moves
 - ▶ We are very interested in this problem as developments in this game demonstrate perhaps methods we could use to investigate other compleutive structures and look at other huge move problems
 - ▶ Cancer cells
 - ▶ Protein arrangements
 - ▶ Molecules

Alpha GO

- ▶ Developed in X by Google
- ▶ World leading method for the playing of GO
- ▶ Uses deep learning (large many multiple layer Neural networks trained by hours and hours worth of Go playing) in order to develop patterns in the play, and learn the correct structures
- ▶ MCTS is then used to produce the plays with the deep network as the evaluation method

Adversarial Networks

- ▶ Say we are not interested in classification, but in creation of objects which are close in style to an artist
 - ▶ Images of Cats
 - ▶ Music
- ▶ We know how to make good classifiers for a problem
 - ▶ We can say it is a cat or not a cat
 - ▶ We can say it is music or noise
- ▶ The generator's job is to become so good that our classifier says its output is in the right class

Forgers and Detectives

- ▶ The detective begins his work, looking at a number of real notes coming from the treasury department, he studies them. He also studies a group of randomly coloured sheets of paper
 - ▶ Once the detective can distinguish between banknotes (YES) and scraps of coloured paper (NO) to a sufficient extent he is given a badge and goes on the hunt
- ▶ The forger now begins its work, taking random scraps of paper as input (random strings) the forger outputs some fake currency
- ▶ The detective now looks at the work of the forger
 - ▶ If the forger fools the detective then they can spend the fake currency and are reward a point
 - ▶ If the forged bill is detected then the forger gets nothing

Forgers and Detectives (cont.)

- ▶ After seeing where the forger was correct or not, the forger trains based on this outcome, attempting to ensure that all these forgeries are not detected
 - ▶ The forger over time will become better and better at fooling the detective
- ▶ Once the forger is fooling the detective enough then the detective goes back to the crime lab and studies the current set of forgeries which are passing them and a new set of examples of real currency
 - ▶ The detective over time will become better than at not being fooled
- ▶ This process continues of training each to defeat the other in a zero sum game
 - ▶ The better the forger is, the worse the detective has become and vice versa

How Does this Now Generate Stuff

- ▶ We can remove the detective and then look at the forgeries
- ▶ As they are training against one and other and the network is sufficiently large, i.e. can build very large equations and add in new terms without disrupting other structures, it is likely that forgers become good enough to fool not just the detective but humans
- ▶ There is a risk of overtraining the network though and just producing the input reality sets
 - ▶ This is fixed by rotating and modifying the training samples and having enough representative samples

Libraries



TensorFlow



Keras



References

- ▶ [Perceptrons](#)
- ▶ [Neural Network Example](#)