TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology
Wardha Road, Nagpur - 441108
Accredited with NAAC A+ Grade
Approved by AICTE, New Delhi, Govt. of Maharashtra
(An Autonomous Institute Affiliated to RTM Nagpur University)

Department of CSE-Data Science

Session 2023-2024

Unit 1:Introduction **Subject: Artificial Intelligence**

Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think and act like humans. It involves the development of algorithms and computer programs that can perform tasks that typically require human intelligence such as visual perception, speech recognition, decision-making, and language translation. AI has the potential to revolutionize many industries and has a wide range of applications, from virtual personal assistants to self-driving cars.

Before leading to the meaning of artificial intelligence let understand what is the meaning of Intelligence-
Intelligence: The ability to learn and solve problems. This definition is taken from webster's Dictionary.

History Of AI

The time between when the phrase "artificial intelligence" was created, and the 1980s was a period of both rapid growth and struggle for AI research. The late 1950s through the 1960s was a time of creation. From programming languages that are still in use to this day to books and films that explored the idea of robots, AI became a mainstream idea quickly.
The 1970s showed similar improvements, such as the first anthropomorphic robot being built in Japan, to the first example of an autonomous vehicle being built by an engineering grad student. However, it was also a time of struggle for AI research, as the U.S. government showed little interest in continuing to fund AI research.
Notable dates include:

- 1958: John McCarthy created LISP (acronym for List Processing), the first programming language for AI research, which is still in popular use to this day.
- 1959: Arthur Samuel created the term "machine learning" when doing a speech about teaching machines to play chess better than the humans who programmed them.
- 1961: The first industrial robot Unimate started working on an assembly line at General Motors in New Jersey, tasked with

- transporting die casings and welding parts on cars (which was deemed too dangerous for humans).
- 1965: Edward Feigenbaum and Joshua Lederberg created the first "expert system" which was a form of AI programmed to replicate the thinking and decision-making abilities of human experts.
- 1966: Joseph Weizenbaum created the first "chatterbot" (later shortened to chatbot), ELIZA, a mock psychotherapist, that used natural language processing (NLP) to converse with humans.1968: Soviet mathematician Alexey Ivakhnenko published "Group Method of Data Handling" in the journal "Avtomatika," which proposed a new approach to AI that would later become what we now know as "Deep Learning."
- 1973: An applied mathematician named James Lighthill gave a report to the British Science Council, underlining that strides were not as impressive as those that had been promised by scientists, which led to much-reduced support and funding for AI research from the British government.
- 1979: James L. Adams created The Standford Cart in 1961, which became one of the first examples of an autonomous vehicle. In '79, it successfully navigated a room full of chairs without human interference.
- 1979: The American Association of Artificial Intelligence which is now known as the Association for the Advancement of Artificial Intelligence (AAAI) was founded.

Application of AI:

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.
Following are some sectors which have the application of Artificial Intelligence:

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:

1. AI in Astronomy

- **Automated Celestial Object Identification:** AI systems can automatically identify and classify celestial objects in astronomical images, aiding in discovering new stars, galaxies, and other cosmic phenomena. In simple words, AI can spot and sort out things in space by looking at pictures. It's like having a cosmic detective that finds new stars, galaxies, and other mysterious objects without human help.
- **Exoplanet Hunting:** AI helps astronomers find planets outside our solar system by looking at lots of data. It can notice tiny changes in the light from stars, which tell us there might be planets around them, such as those caused by exoplanet transits.
- **Analyzing Space Information:** AI plays a crucial role in the study of space. It assists scientists by carefully examining vast amounts of complex data gathered from space observations. This helps astronomers uncover sophisticated patterns, unusual phenomena, and connections that might be otherwise very difficult to notice. Essentially, AI acts as a dedicated assistant, sifting through the cosmic data haystack to find the valuable needles of knowledge.
- **Watching Space Events in Real-time:** AI-powered tools can keep a constant eye on the night sky, looking out for sudden happenings like exploding stars (supernovae) or bursts of powerful gamma rays. This allows scientists to quickly study these events in more detail when they occur.
- **Making Telescopes Smarter:** AI is like a brain for telescopes. It helps them work better by changing their settings on the fly. For example, if the weather gets cloudy or if scientists want to study something specific in space, AI can adjust the telescope to get the best results. It's like having a telescope that can think and adapt to the situation.

2. AI in Healthcare

- **Helping Doctors See Inside the Body Better:** AI is like a super helper for doctors when they look at pictures of the inside of a patient's body, like X-rays or MRIs. It uses smart algorithms to find
- things like problems, tumors, or broken bones very accurately. This means doctors can figure out what's going on faster and more accurately, which is great for patients and for better diagnosis.

- **Detecting Health Problems Early:** AI acts as a health detective. It looks at your health information to find out if you might get certain diseases in the future. When it sees a high risk, doctors can step in early to help you stay healthy. This is really important for conditions like diabetes and heart problems because catching them at this time means better treatment and less trouble for the patient.
- **Developing Medications Quickly and Cost-Effectively:** AI acts like a super scientist in the lab. It uses certain algorithms to predict how different chemicals can fight diseases. This helps us make new medicines much quicker and at a low cost. So people can get the treatments they need sooner, and it doesn't cost as much money to manufacture them.
- **Personalized Treatment Plans:** AI looks at your health information, like your genes, what happened to you before, and how you've responded to treatments. Then, it makes a special plan just for you. This means your treatment works better and doesn't give you as many problems. It's like having a personalized health coach, which helps in avoiding complications caused by improperly prescribed medicine.
- **Managing Hospital Functions and Resources:** AI acts like a manager for hospitals. It helps with things like when patients come in, where to put resources like doctors and supplies, and how to make sure everything runs well. It can even guess how many patients might come in ahead of time, so hospitals use their staff and resources in the best way possible.

3. AI in Gaming

- **Smart Game Characters:** AI is like the brains behind game characters that aren't controlled by players. They make these characters, called NPCs, act more like real people or clever enemies. They can learn from what players do and change their behavior, which makes games more exciting and lifelike. Imagine playing a game where the bad guys learn and adapt to your moves - that's what AI does.
- **Creating Game Worlds with AI:** AI can make parts of video games all on its own. It can create levels, maps, and places to explore without
- people having to make them by hand. This means games can have bigger and more interesting worlds because AI does a lot of the work, kind of like a game world builder. It helps game developers, too.

- **Making Games Look and Feel Real:** AI helps to make games look and act more like the real world. They create graphics that look just like the things we see, and they make how things move in games feel realistic, like in real life. They even guess what players might do next so the game looks smooth and natural.

4. AI in Finance

- **Identifying and Prevention of Fraud:** AI keeps an eye on bank transactions all the time. They act like super detectives who can spot strange things happening with money, like someone using a credit card in a weird way. When they see something fishy, they raise the alarm and help the bank stop bad people from stealing money. This happens really fast, without needing people to check every transaction.
- **Automated Trading:** AI helps a skilled trader who works automatically. It uses various algorithms to swiftly buy and sell stocks while analyzing all the market information. This boosts trading strategies, making investments more efficient and profitable.
- **Risk Control:** AI helps in examining lots of data to check how risky something is, like giving out loans or making investments. It looks at things like whether someone can pay back a loan or how safe an investment is. This helps banks and investment firms make smarter choices so they don't lose money and can help others save and grow their money.

5. AI in Data Security

- **Anamoly Detection:** AI works as a digital detective. It looks at big piles of data and watches for anything strange or out of the ordinary, like someone sneaking into a digital vault or trying to steal secrets. When it sees something fishy, it raises the alarm, helping to keep important data safe from cyber-attacks.
- **Predicting Threats:** AI looks at past troubles and keeps an eye on new dangers that are popping up. By doing this, it can predict what bad things might happen in the future, like a security breach or a cyberattack. This way, companies can get ready in advance to protect their important data, sort of like putting up a strong fortress before any
- attack happens.
- **Automated Safety Response:** AI acts like a digital guardian that can respond when there's trouble. If it sees something bad happening, like a

cyberattack, it can automatically take action. It might isolate the part that's under attack. This way, it keeps your important stuff safe in the digital world.

6. AI in Social Media

- **Smart Suggestions:** AI helps as a guide on social media. It watches what you like and what you do, and then it suggests things you might enjoy, like posts, videos, or ads. It acts as someone who knows your tastes and shows you stuff you're really into, making your social media experience more enjoyable and personalized.
- **Virtual Assistants and Chatbots:** AI chatbots and virtual assistants act as digital helpers on social media. They're quick to respond and can talk to you just like a real person. They answer your questions, share information, and even help with problems. It's like having an assistant available 24/7, making your social media experience smoother and more helpful.
- **Sentiment Analysis:** AI can figure out how people feel on social media. It looks at what they say in comments and posts and decides if it's a happy, sad, or neutral kind of message. This helps companies understand what people think so they can react in the right way. It's like having a mood gauge for the internet so businesses can make their customers happier.
- **Trend Analysis:** AI keeps track of all the chats and what's popular right now. This helps companies and regular folks understand what everyone's thinking and talking about. It acts as a social media news reporter that keeps customers in the loop about what's hot and what people are buzzing about.

7. AI in Travel & Transport

- **Optimization of Route:** AI plays a crucial role in optimizing travel routes, be it for parcel deliveries, public transportation, or personal trips. It efficiently calculates the swiftest and most economical paths from one point to another point, resulting in reduced travel time, minimized fuel consumption, and cost savings. Essentially, it serves as a pocket-sized travel advisor, enhancing the speed and budget-
- friendliness of your journeys.
- **Smart Security Screening:** AI helps in keeping traveling safely. It uses special skills to scan bags and people quickly. It can spot things that

might be dangerous and make security checks faster and smoother. This means you can fly knowing that the airport is working hard to keep you safe without making your travel a hassle.

- **Chatbots for Travel Support:** AI chatbots are like digital travel helpers. These chatbots are capable of aiding you in various tasks such as reserving tickets, suggesting interesting destinations to explore, and providing responses to your inquiries, much like an affable travel consultant. This elevates the convenience and pleasure of your travel adventures, as you can access assistance whenever it's required, even during late-night hours.
- **AI Prevents Breakdowns:** AI works like a fortune teller for machines like cars, planes, and roads. It predicts when they might get sick and need fixing. This way,
- we can fix them before they break down and cause problems. It keeps everything

8. AI in Automotive Industry

- **Self-Driving Cars:** AI is like the brain of self-driving cars. It looks at what's happening around the car using various sensors and decides what the car should do, like turning or stopping. It's like having a super-smart driver that doesn't need a person. This makes cars drive on their own, making travel more convenient and safer because there's no need for a human to steer.
- **Advanced Driver Assistance Systems (ADAS):** AI adds extra smarts to your car to keep you safe. It possesses the capability to autonomously adjust your vehicle's speed while on the highway, assist in maintaining your lane, and swiftly engage the brakes when detecting potential hazards. These intelligent functionalities function akin to a co-pilot, ensuring your safety by preventing accidents and ensuring your safe arrival at your intended destination.
- **Streamlining Production Processes:** AI watches over machines, checks if they're healthy, and makes sure they don't break. It also helps with ordering materials and makes sure everything is made just right. This makes things faster, cheaper, and better quality, like having a super factory manager.

- **Voice Recognition:** AI-driven voice recognition systems allow drivers to control various functions in their vehicles, such as navigation, music, and communication, using natural language.

9. AI in Robotics:

- **Self-Moving Robots:** AI makes robots really smart at moving around on their own. It's like giving them a built-in GPS and a clever brain. They can figure out where to go and how to get there without bumping into things or needing a person to show them the way. This helps them do tasks like delivering packages or exploring places on their own, making them super independent.
- **Object Recognition and Manipulation:** AI gives robots sharp eyes and clever hands. It helps them see objects clearly and then pick them up and move them just right. This is super useful, especially in places like warehouses, where they can do things like sorting and packing items accurately.
- **Collaboration of Humans and Robots:** AI makes it possible for robots to be great team players with people. They can work alongside humans, helping out and learning from them. If a person does something, the robot can understand and follow their lead. This makes workplaces safer and more efficient, like having a trusty robot colleague who understands and supports you.

10. AI in Entertainment

- **Recommendation of Content:** AI looks at what customers have liked before, such as movies or music, and suggests new things that they might enjoy. It's like having a personal entertainment guide, making their experience more enjoyable by offering just what they like.
- **AI as a Creative Assistant:** AI acts as a creative sidekick for artists and creators. It can make music, art, and videos or help improve what they create. It's like having a helper that speeds up the creative process, making it easier to bring new ideas to life. This way, artists can focus more on their vision, and AI handles the technical bits.
- **Live Event and Performance Enhancements:** AI makes live events and performances even cooler. It can translate what people are saying in real time, add cool effects that blend with what's happening, and even predict what the audience will like. This makes shows and events

- more exciting and enjoyable for everyone there. It's like having a magic touch that brings performances to life in new and amazing ways.

11. AI in Agriculture

- **Crop Observation and Control:** AI, with the help of various sensors, acts as a guardian for crops on the farm. It keeps an eye on them, making sure they're healthy and growing well. It tells farmers when it's the best time to plant, water, and harvest the most crops. It's like having a farm expert who ensures the fields are super productive so farmers can get the most out of their hard work.
- **Smart Farming for Efficiency:** AI makes farming super efficient. It helps farmers use just the right amount of things like fertilizer and pesticides, not too much and not too little. This means there's less waste, and the crops grow better. It's like having a precise chef in the field, making sure everything is just perfect for the plants to thrive and produce lots of food.
- **Automated Farming:** AI controls a number of machines like tractors and drones. These machines can plant seeds, remove weeds, and spray stuff on crops all by themselves. They do it super well and exactly as needed, like having expert farmers who never get tired and work perfectly, making farming easier and more efficient.
- **Monitoring Livestock:** AI uses special sensors and smart data analysis to make sure they're healthy and happy. If anything is wrong, it alerts the farmer. This way, the animals are well taken care of, and the farm can run smoothly. It's like having a watchful friend for the animals, making sure they're okay and the farm works better.

12. AI in E-commerce

- **Personalized Product Suggestions:** AI looks at what you've looked at and bought before and suggests things you might really like. It's like having a personal shopper who knows your style, making your online shopping more fun and helping you discover new things you might want to buy. Plus, it's great for the store because it helps them sell more, and as a customer, it saves your time.
- **Managing Inventory:** AI takes care of a store's shelves. It predicts how much of each product people will buy and automatically orders more when needed. In this manner, there exists an optimal balance of products, preventing excessive stock that ties up funds while also

ensuring an adequate supply to prevent customers from leaving without making a purchase.

- **Dynamic Pricing:** Artificial intelligence dynamically adjusts pricing according to demand, market competition, and inventory levels, ensuring customers receive optimal value while enhancing the store's profitability.

13. AI in education:

- **Education Content Creation:** AI acts as a teaching assistant for educators. It helps them make things like quizzes, lesson plans, and study materials. This makes teaching easier and better because educators have more time for students, and the materials are top-notch. It's like having a super-efficient helper who does the paperwork, leaving teachers more time to inspire students.
- **Virtual Learning Assistants:** AI is there to answer questions, explain things, and offer help whenever students need it, day or night. This makes learning easier and more fun because students have someone to turn to whenever they're stuck. It also takes some pressure off teachers because AI can handle common questions, leaving more time for personalized teaching.
- **Automated Assessment and Instant Feedback:** AI acts like a super-speedy homework checker. It looks at your assignments and tests and gives you grades and feedback right away. This aids in gauging your progress and pinpointing areas for potential enhancement. Furthermore, it alleviates some of your teacher's grading responsibilities, allowing them to dedicate more time to teaching rather than paper evaluation.
- **Customized Learning Routes:** AI figures out what you're good at and where you might need extra help. Then, it gives you the right stuff to learn and the best way to learn it. This makes learning easier and more fun.

Conclusion

The applications of AI are vast and diverse, touching nearly every aspect of our lives. From healthcare to finance, astronomy to gaming, and transportation to entertainment, AI is reshaping industries and propelling us into a future where the possibilities seem limitless. As AI continues to

advance, its impact on society is poised to grow, promising increased

efficiency, better decision-making, and innovative solutions to some of our most pressing challenges. Embracing and responsibly harnessing the power of AI will be key to unlocking its full potential and ensuring a brighter future for all.

History of Artificial Intelligence.

Artificial intelligence, for those who do not use it on a daily basis, seems like a concept typical of great film productions or science fiction books. But the truth is that it is a set of almost century-old concepts that are increasingly present and to which we resort, often without realising it. Find out what artificial intelligence is, what it is for, what its risks and challenges are and what we expect from it in the future**.**

Artificial intelligence has become a broad and revolutionary tool with countless applications in our daily lives. Able to give birth to robots that act with human-like responses and to respond to voice requests with practical functionalities in mobiles and speakers, artificial intelligence has attracted the attention of Information and Communications Technology (ICT) companies around the world and is considered the FourthTechnological Revolution following the proliferation of mobile and cloud platforms. Despite the innovation it brings to our lives, its history is a long process of technological advancement.

Definition and Origins of Artificial Intelligence

When we speak of "intelligence" in a technological context we often refer to the ability of a system to use available information, learn from it, make decisions and adapt to new situations. It implies an ability to solve problems efficiently, according to existing circumstances and constraints. The term "artificial" means that the intelligence in question is not inherent in living beings, but is created through the programming and design of computer systems.

As a result, the concept of **"artificial intelligence" (AI) refers to the simulation of human intelligence processes by machines and**

software. These systems are developed to perform tasks that, if performed by humans, would require the use of intelligence, such as learning, decision-making, pattern recognition and problem solving. For example, managing huge amounts of statistical data, detecting trends and making recommendations based on them, or even carrying them out.

Today,AI is not about creating new knowledge, but about collecting and processing data to make the most of it for decision making. It rests on three basic pillars.

Basics of Problem Solving:

The reflex agent of AI directly maps states into action. Whenever these agents fail to operate in an environment where the state of mapping is too large and not easily performed by the agent, then the stated problem dissolves and sent to a problem-solving domain which breaks the large stored problem into the smaller storage area and resolves one by one. The final integrated action will be the desired outcomes.

On the basis of the problem and their working domain, different types of problem-solving agent defined and use at an atomic level without any internal state visible with a problem-solving algorithm. The problem-solving agent performs precisely by defining problems and several solutions. So we can say that problem solving is a part of artificial intelligence that encompasses a number of techniques such as a tree, B-tree, heuristic algorithms to solve a problem.

We can also say that a problem-solving agent is a result-driven agent and always focuses on satisfying the goals.

There are basically three types of problem in artificial intelligence:
1. Ignorable: In which solution steps can be ignored.
2. Recoverable: In which solution steps can be undone.
3. Irrecoverable: Solution steps cannot be undo.
Steps problem-solving in AI: The problem of AI is directly associated with the nature of humans and their activities. So we need a number of finite steps to solve a problem which makes human easy works.
These are the following steps which require to solve a problem :

- **Problem definition:** Detailed specification of inputs and acceptable system solutions.
- **Problem analysis:** Analyse the problem thoroughly.
- **Knowledge Representation:** collect detailed information about the problem and define all possible techniques.
- **Problem-solving:** Selection of best techniques.

Components to formulate the associated problem

Initial State: This state requires an initial state for the problem which starts the AI agent towards a specified goal. In this state new methods also initialize problem domain solving by a specific class.

Action: This stage of problem formulation works with function with a specific class taken from the initial state and all possible actions done in this stage.
Transition: This stage of problem formulation integrates the actual action done by the previous action stage and collects the final stage to forward it to their next stage.
Goal test: This stage determines that the specified goal achieved by the integrated transition model or not, whenever the goal achieves stop the action and forward into the next stage to determines the cost to achieve the goal.
Path costing: This component of problem-solving numerical assigned what will be the cost to achieve the goal. It requires all hardware software and human working cost.

Summer-time is here and so is the time to skill-up! More than 5,000 learners have now completed their journey from **basics of DSA to advanced level development programs** such as Full-Stack, Backend Development, Data Science.

A toy problem is intended to illustrate or exercise various problem solving methods. It can be given a concise, exact description. This means it can be used easily by different researchers to compare the performance of algorithms.

A real world problem is one whose solutions people actually care about, they tend not to have a single agreed upon description, but we will attempt to give the general flavor of their formulations.

Toy problems:

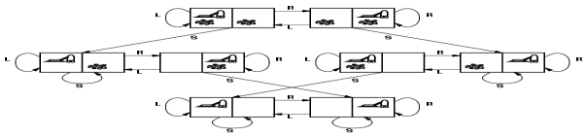
1] vacuum world. This can be formulated as a problem as follows:

State: The agent is in one of the two locations, each of which might or might not contain dirt. Thus, there are $2\times2^2=8$2×22=8 possible world standard.
Initial state: Any state can be designated as the initial state.

Successor function: This generates the legal states that result from trying the three actions (left, right and suck). The complete state space is shown in figure (3).

Goal test: This checks whether all the squares are clean.

Path cost: Each step costs 1, so the path cost, is the number of step in the path.

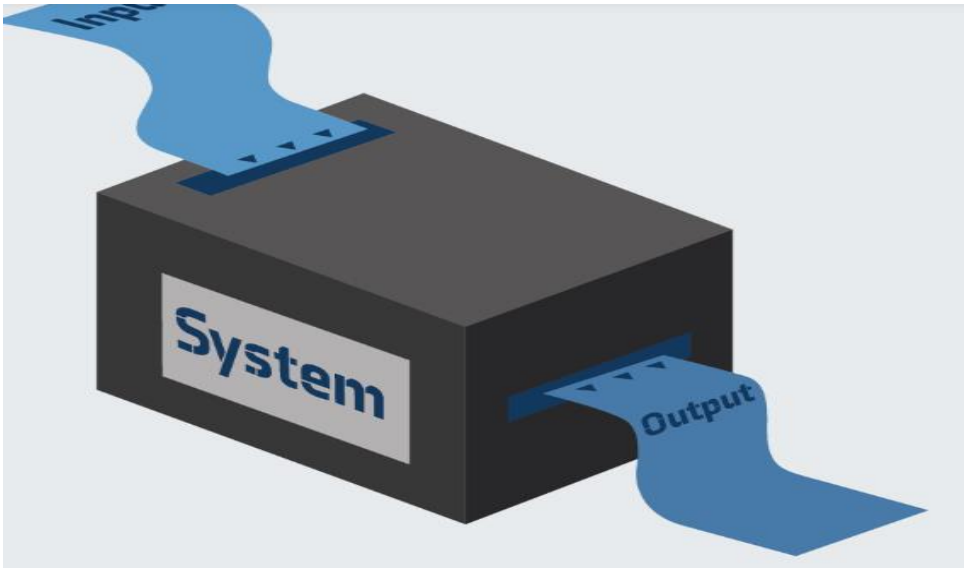


igure (3), the state space for the vacuum world. Arcs denote actions: L = Left, R = Right, S = Suck. Compared with the real world, this toy problem has discrete locations, discrete dirt, reliable cleaning and it never gets messed up once cleaned. One important thing to note is that the state is determined by both the agent location and the dirt locations. A larger environment with n locations has h22*h*22 states**.**

This step introduces the basics of production systems, their components and what systems are usually made up of.

A production system transforms input to output. Meaning, it's the systems that manufacture a product consisting of whatever components is needed to make it a reality**.**

Let's take a closer look at how input and output work with a system:



The black box in the picture represents the production system itself. On the left we have what goes into the system *(input)* and on the right is what comes out of the system *(output)*.

input goes into the system. When going through the system, it becomes transformed in some way making the output different than what came in, resulting in something with different characteristics. Of course, a system may have more than one input and output.

There are usually five types of ways of transforming input into output:

- **Separating** – One item enters and two or more exit. Example: A wooden plank is cut into two.
- **Putting together** – Several items enter and one exits. Example: Wooden planks that are glued together.
- **Detaching** – And item enters and exits shaped differently, alongside waste. Example: A block of wood is shaped with a lathe.
- **Forming** – An item enters and exits in a different shape, without waste. Example: A piece of metal is shaped by hammering the object.
- **Quality adaptation** – An item enters and exits with different characteristics. Example: Surface treatment of a metal object.

It is not always easy to directly understand what happens in the black box – often we simply accept that it's there. Further on in this activity we will examine what might happen within the production system black box. This is a large part of understanding production systems.

A production system in AI, also known as a production rule system, is a framework that can be implemented to create various software programs for executing different tasks. It is a unique and advanced kind of intellectual design that may be used to simulate human problem-solving abilities and build search algorithms for different programs. The AI production system retains tiny quanta referred to as productions, which help in understanding the process of problem-solving abilities

The rule and action are two fundamental elements in each production system. Declarative statements are used to encode the data in production systems. Knowledge representation is used to develop a production system that executes AI applications.

Production of different AI-based systems, such as computer software, mobile applications, and production tools, includes an effective production system. A production system

in **artificial intelligence** provides automation based on a particular set of rules to exhibit particular traits while participating in different situations.**Components of a Production System in AI**

A production system in AI is composed of three components:

Global Database

The global database consists of the architectural design of the production system. It is the primary data framework of the system. This database equips all of the skills and data necessary for completing a particular task.

Temporary and permanent global databases are two different kinds of global databases. The temporary global database is made up of short-term, situation-based actions. Whereas the permanent global database includes specific actions that cannot be modified or changed.

Production Rules

Production rules in AI are a group of rules that apply to information collected from a global database. Each rule has a precondition and a postcondition that the global database must either fulfill or not, depending on the rule. A production rule works effectively if a condition gets processed through it and meets the criteria specified by the global database.

Control System

A control system carries out the decision-making process. The control system determines which suitable rule is to be used and stops computations when a database termination condition is met. The control system settles issues when numerous rules are supposed to be executed simultaneously. The control system approach defines the set of rules that assesses the data from the global database before arriving at the right conclusion.

An intelligent agent is a combination of Agent Program and Architecture.
Intelligent Agent = Agent Program + Architecture
Agent Program is a function that implements the agent mapping from percepts to actions. There exists a variety of basic agent program designs, reflecting the kind of information made explicit and used in the decision process. The designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.

Architecture is a computing device used to run the agent program.

To perform the mapping task **four types of agent programs** are there. They are:

A rational AI agent is a system that makes optimal decisions to achieve its goals, while an intelligent agent learns and adapts to its environment. Read more about rational AI agents in the blog.

1.Structure Of Agent.
2. Model-based reflex agents.
3. Goal-based agents.
4. Utility-based agents.
We then explain in general terms how to convert all these into learning agents.

1. Structure Of Agent

The simplest kind of agent is the simple reflex agent. It responds directly to percepts i.e. these agent select actions on the basis of the current percept, ignoring the rest of the percept history.

An agent describes about how the **condition action rules** allow the agent to make the connection from percept to action.

Condition action rule: if condition then action

First, let’s understand what an agent is. An agent is an entity, any autonomous entity, that observes their environment, and acts to move towards their goal. For example, an ATM is an agent. It either gives or doesn’t give cash when it receives information about the amount and bank details.

An AI agent and it's environment

Now, let’s move to rational AI agents. A rational AI agent is the agent that does the right thing. What does this mean?

A rational AI agent is a system that takes steps to achieve the best outcome or, when there is uncertainty, the best expected outcome. Unlike other agents that might act based on rules, rational agents use their information to maximize their performance measure and gain the maximum benefit from their actions.

Suppose we have a robot that cleans floors, a simple way to see if it's doing a good job might be to look at how much dirt it picks up in an eight-hour day.

It's also important when we check on the robot's work. If we only look after an hour, we might think robots that start strong are the best, even if they don't do much after that. So, it's better to see how they do over a longer time, like the whole day or their entire working life.

What are the components of a rational AI agent?

Rational AI agents are designed to make the best possible decisions based on their environment, their states, goals and more. Let’s understand what makes up a rational AI agent and then we can understand how they work with an example.

The components of a rational agent facilitate this decision-making process, enabling the agent to perceive, reason, act, and learn from its actions. Here's an overview of the key components:

1. Sensors

Sensors allow the agent to perceive its environment. These can be physical sensors, like cameras, microphones, and temperature sensors in robotics, or virtual sensors, such as data inputs in software agents. Sensors provide the raw data necessary for the agent to understand its surroundings and make informed decisions.

2. Actuators

Actuators enable the agent to take actions in the environment. In physical robots, actuators might include motors, wheels, or arms. In software agents, actuators could be functions that initiate actions, such as sending an email, placing an online order, or adjusting parameters within a system. Actuators are the means through which the agent affects the world around it.

3. Performance Measure

The performance measure defines the criteria for success for the agent. It is a set of metrics used to evaluate how well the agent is achieving its goals. A well-defined performance measure guides the agent's decision-making process by providing a clear objective to maximize through its actions.

4. Agent Program

The agent program is the core logic that processes the input from sensors, decides what actions to take based on the current state and goals, and controls the actuators. This program can be based on simple rules, machine learning algorithms, or complex models that involve planning and reasoning. The sophistication of the agent program determines the agent's ability to make rational decisions.

5. Internal State

The internal state represents the agent's current understanding or model of the world, based on past perceptions and actions. This component is crucial for agents operating in complex or dynamic environments where it's necessary to track changes over time. The internal state

allows the agent to consider its history and predict future states of the environment, aiding in more sophisticated decision-making.

6. Learning Component

A learning component enables the agent to improve its performance over time based on experience. This could involve adjusting the agent program to better achieve its goals, refining its model of the world, or altering its strategy for decision-making. Learning mechanisms can range from simple feedback loops to advanced machine learning and deep learning techniques.

7. Knowledge Base

The knowledge base contains information and rules about the environment, tasks, and strategies for achieving goals. It supports the agent's reasoning and decision-making processes by providing a repository of facts and heuristics that the agent can draw upon when making decisions.

How does a rational agent work?

A rational agent operates by consistently making decisions that lead to the best possible outcome or, in situations of uncertainty, the best expected outcome based on its understanding and the information available to it. Here's a simple breakdown of how a rational agent works:

1. **Perception of the Environment:** A rational agent starts by observing its surroundings through sensors or data input. This could be anything from the current market conditions for a trading agent to the visual input for an autonomous vehicle.

2. **Understanding Goals:** The agent has a clear goal or set of goals it aims to achieve. These goals are defined by the performance measure, which dictates what success looks like for the agent. For instance, a cleaning robot’s goal might be to clean a room as efficiently as possible.

3. **Decision-Making Based on Knowledge:** With its goals in mind, the agent uses its built-in knowledge or model of the world to make decisions. This knowledge helps the agent predict the outcomes of various actions. In more complex scenarios, this step might involve reasoning or planning several steps ahead to determine the best course of action.

4. **Learning from Feedback:** Many rational agents are designed to learn from the outcomes of their actions. If an action gets the agent closer to its goal, the agent takes note and is more likely to repeat that action in similar situations in the future. Conversely, if an action doesn't lead to a desired outcome, the agent will try to avoid that action under similar circumstances.

5. **Taking Action:** Based on its decision-making process, the agent then takes an action that it believes will maximize its performance measure. The action is executed through actuators or output mechanisms that allow the agent to interact with its environment.

6. **Repeat the Process:** The rational agent continuously goes through this cycle of perceiving, deciding, and acting, learning from the outcomes to improve its performance over time.

Example of Rational agent

We just saw the different components of a rational AI agent, let's take a look at an example of an autonomous vehicle and see why it is a rational agent.

- **Perception:** The vehicle uses cameras and sensors to understand its surroundings, including other vehicles, road signs, and pedestrians.
- **Understanding Goals:** Its goals include reaching a destination safely and efficiently while following traffic laws.
- **Decision-Making:** The vehicle uses its knowledge of the roads, traffic laws, and current traffic conditions to make decisions, such as when to speed up, slow down, or change lanes.
- **Learning:** Over time, the vehicle learns from experiences, such as which routes are faster at different times of day or how to better anticipate the actions of other drivers.
- **Action:** The vehicle executes its decisions by steering, accelerating, or braking.
- **Repeat:** This process continues throughout the journey, with the vehicle constantly adjusting its actions based on new input and learned experiences to achieve its goal effectively.

Intelligent agent vs Rational Agent

intelligent agents and rational agents stand out, each defined by unique characteristics and operational frameworks. Let's explore these two types of agents in detail, understanding their definitions, key characteristics, and how they navigate their environments to achieve their goals.

Intelligent Agents: Autonomy and Learning at the Forefront

An intelligent agent is a system that perceives its environment and takes actions to achieve specific goals. These agents are marked by several key characteristics:

- **Autonomy**: Intelligent agents operate independently, making decisions without human intervention. This autonomy allows them to perform tasks and make choices based on their perception and understanding of the environment.
- **Adaptability**: They can adjust their behavior based on changes in their environment. This adaptability is crucial for navigating complex and dynamic settings.
- **Learning Capability**: Intelligent agents have the ability to learn from experiences, improving their performance over time. This learning can occur through various methods, such as reinforcement learning, allowing the agent to better achieve its objectives through experience.

Goal-Oriented: These agents are designed with specific goals in mind. They use their learning and adaptation capabilities to navigate complexities and make decisions that align with their objectives.

Rational Agents: Optimal Decision-Making and Utility Maximization

A rational agent, on the other hand, is defined as a system that always makes the best possible choice to maximize its expected utility, based on its knowledge. The operation of rational agents is characterized by:

- **Optimality**: Rational agents strive for the optimal outcome, making decisions that maximize their expected utility. This approach ensures that the agent is always working in its best interest, based on its understanding of the environment.
- **Utility Maximization**: At the core of a rational agent's operation is the aim to maximize a clearly defined utility or performance measure. This focus on utility maximization guides all decision-making processes.
- **Adaptability**: While adaptability is also crucial for rational agents, it is specifically oriented towards achieving better outcomes as defined by predefined performance measures. This means that a rational agent will adjust its strategy if it leads to a better achievement of its goals.
- **Learning**: Similar to intelligent agents, rational agents can learn from their experiences. However, in this context, learning is specifically a means to enhance rationality by updating the agent's knowledge or strategies to maximize performance.
- **Decision-making**: Rational agents make decisions based on a utility function or a set of rules designed to ensure the maximization of expected utility. This focused approach to decision-making sets them apart from intelligent agents, which may consider a broader range of factors.

Rational agents are designed to make decisions that maximize outcome. This makes it useful in a business context. Let's look at some business applications where rational AI agents can play a vital role.

Customer Success

Rational AI agents are widely used in customer success. Their goal is to ensure the customer's query is solved and they're provided a huge knowledge base of company documents to do so. Apart from this, AI agent platforms like Alltius can also use AI to predict customer behavior to allow businesses to tailor perfect pitches to the customer and thus, increase conversion rates.

Dynamic Pricing

E-commerce and retail businesses use rational AI agents for dynamic pricing strategies. These agents analyze market demand, competitor pricing, and inventory levels to adjust prices in real-time, maximizing sales and profits.

Human Resources

AI agents assist in the HR domain by automating the screening and selection process, identifying the best candidates based on the requirements of a job. They can also predict employee turnover and identify factors that influence employee satisfaction and performance.

Fraud Detection

Financial institutions use rational AI agents to detect and prevent fraud. By analyzing transaction patterns and behaviors, these agents can identify anomalous activities that may indicate fraudulent actions, reducing financial losses.

Personalized Recommendations

In digital platforms and services, rational AI agents analyze user behavior to provide personalized content, product recommendations, and services. This enhances user engagement and increases conversion rates by offering tailored experiences.

AI with Alltius

Alltius is pioneering the use of generative AI in customer success and sales to improve buyer journeys across channels. Alltius is made by AI experts from CMU, Google, Amazon and more. With alltius.ai, sales and customer success teams can sell 3X more and reduce average resolution time to 10 seconds within weeks. Access a free trial or book a demo.

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.

Mini-Max algorithm uses recursion to search through the game-tree.
Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.

In this algorithm two players play the game, one is called MAX and other is called MIN.
Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm:

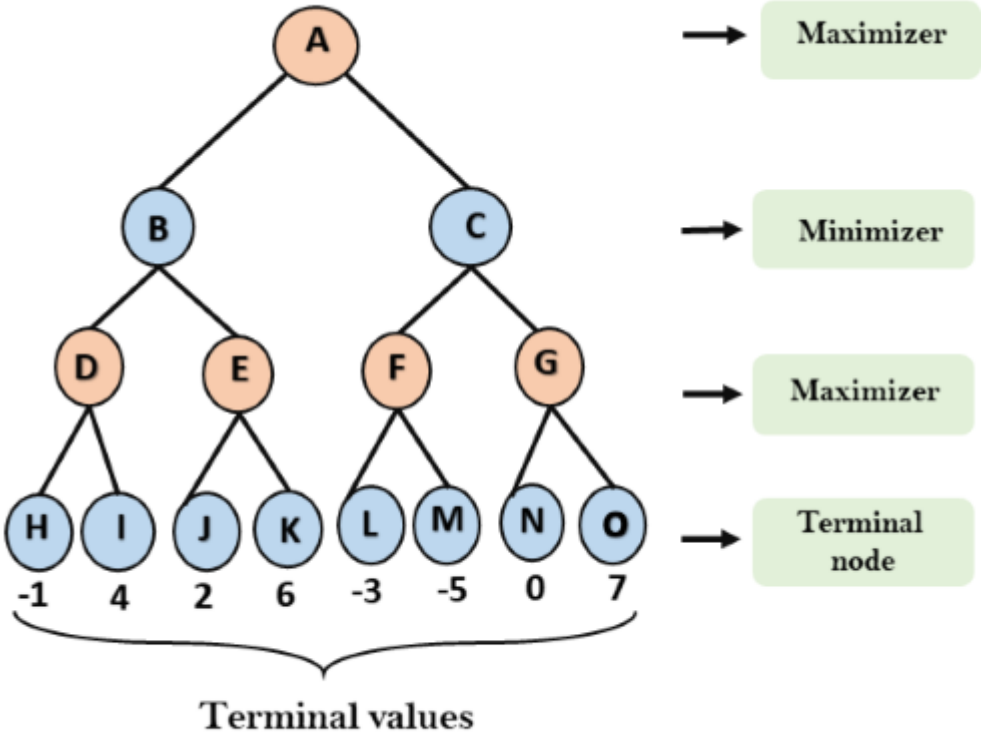
```
1. function minimax(node, depth, maximizingPlayer) is
2. if depth ==0 or node is a terminal node then
3. return static evaluation of node
4.
5. if MaximizingPlayer then      // for Maximizer Player
6. maxEva= -infinity
7.  for each child of node do
8.  eva= minimax(child, depth-1, false)
9. maxEva= max(maxEva,eva)       //gives Maximum of the values
10. return maxEva
11.
12. else                         // for Minimizer player
13.  minEva= +infinity
14.  for each child of node do
        eva= minimax(child, depth-1, true)
     minEva= min(minEva, eva)//gives minimu     m of the values
15.  return minEva
```

Initial call:
Minimax(node, 3, true)
Working of Min-Max Algorithm:

- **The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.**
- **In this example, there are two players one is called Maximizer and other is called Minimizer.**
- **Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.**

- **This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.**
- **At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:**

Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.

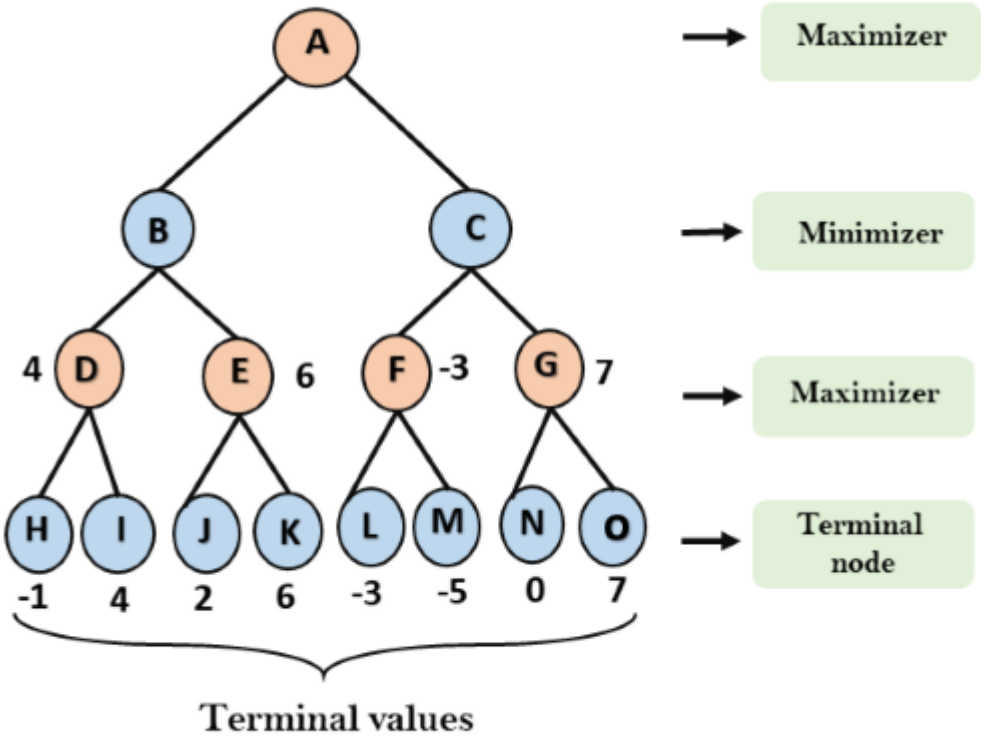


Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

Backward Skip 10sPlay VideoForward Skip 10s

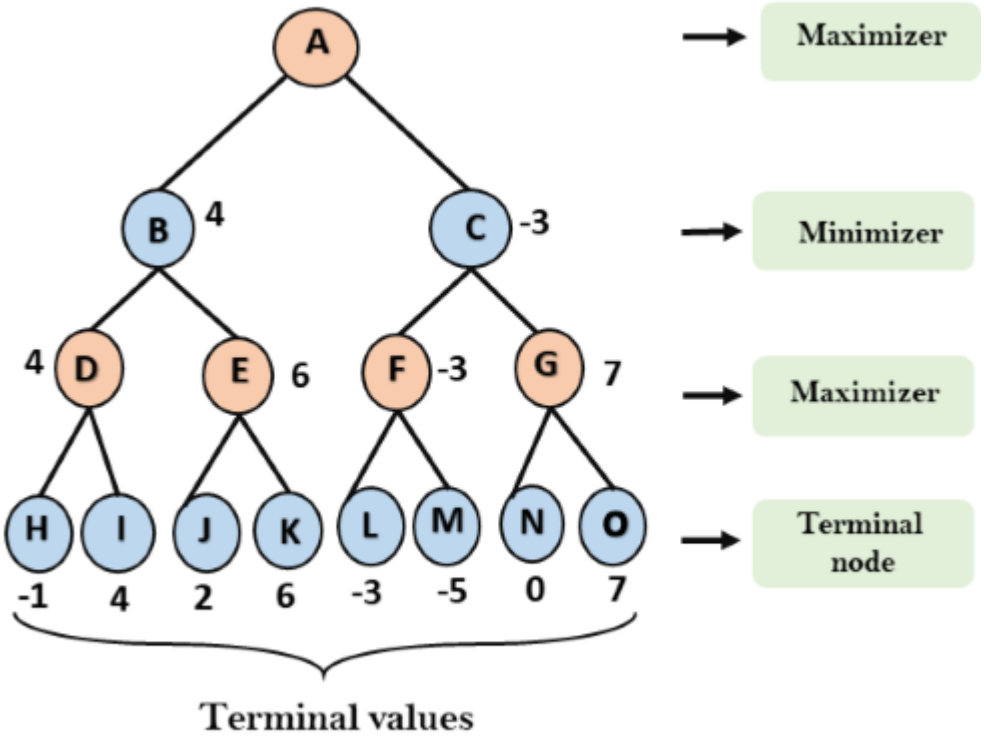
ADVERTISEMENT

- **For node D max(-1,- -∞) => max(-1,4)= 4**
- **For Node E max(2, -∞) => max(2, 6)= 6**
- **For Node F max(-3, -∞) => max(-3,-5) = -3**
- **For node G max(0, -∞) = max(0, 7) = 7**



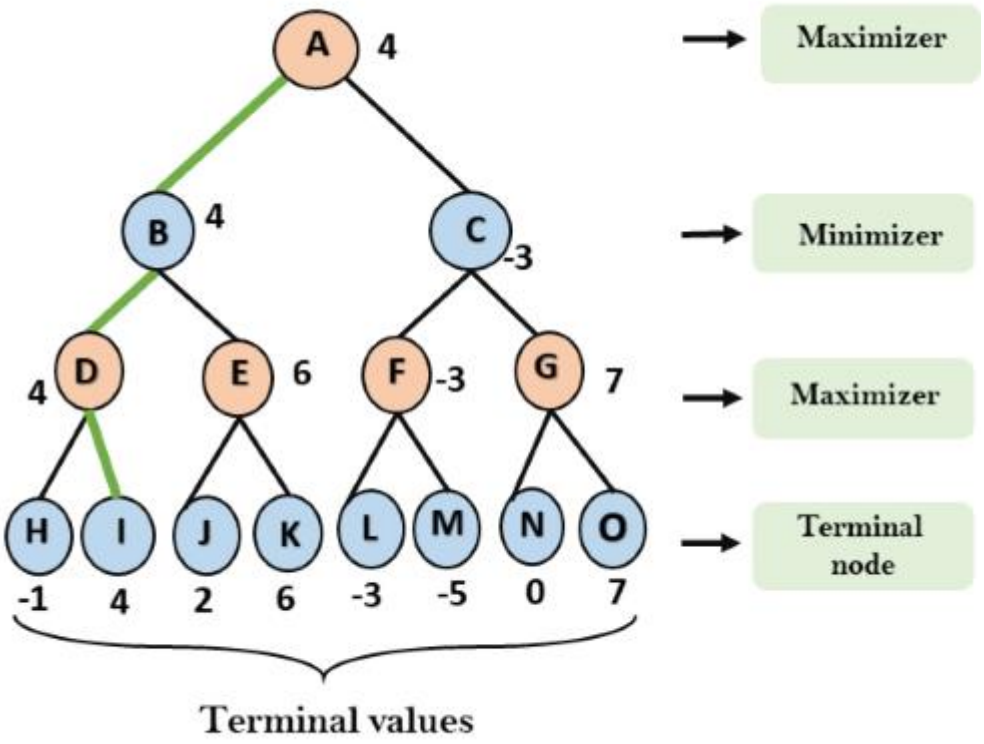
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

- **For node B= min(4,6) = 4**
- **For node C= min (-3, 7) = -3**



Step 4: Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- **For node A max(4, -3)= 4**



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

Complete- Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.

Optimal- Min-Max algorithm is optimal if both opponents are playing optimally.

Time complexity- As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is $O(b^m)$, where b is branching factor of the game-tree, and m is the maximum depth of the tree.

Space Complexity- Space complexity of Mini-max algorithm is also similar to DFS which is O(bm).

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from alpha-beta pruning which we have discussed in the next topic.

Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.

As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two

threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.

Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

The two-parameter can be defined as:

. Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.

. Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Note: To better understand this topic, kindly study the minimax algorithm.

Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

1. **α>=β**

Key points about alpha-beta pruning:

- **The Max player will only update the value of alpha.**
- **The Min player will only update the value of beta.**
- **While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.**
- **We will only pass the alpha, beta values to the child nodes.**

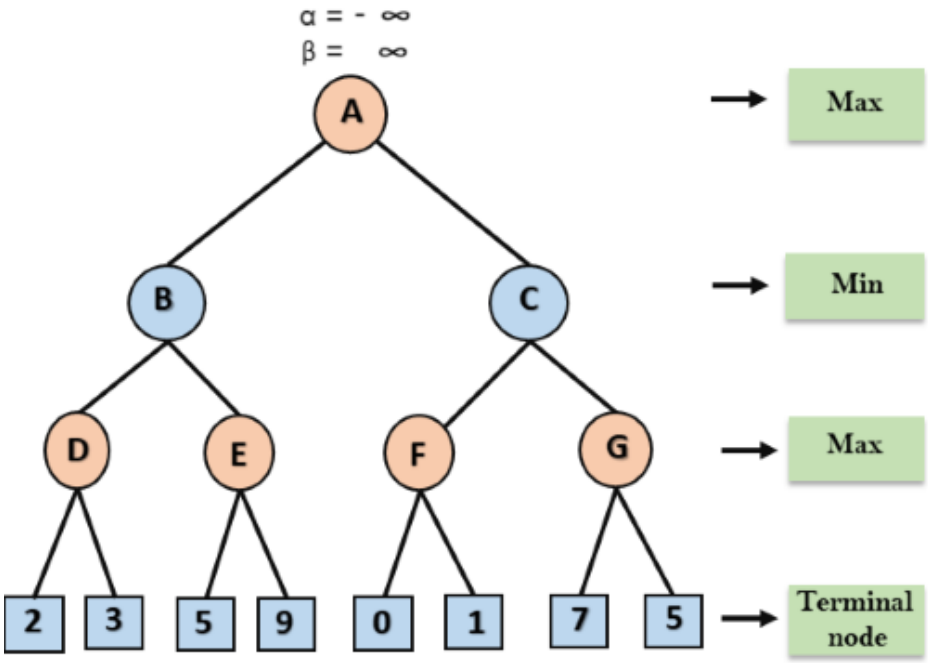
Pseudo-code for Alpha-beta Pruning:

```
1. function minimax(node, depth, alpha, beta, maximizingPlayer) is
2. if depth ==0 or node is a terminal node then
3. return static evaluation of node
4.
5. if MaximizingPlayer then      // for Maximizer Player
6.    maxEva= -infinity
7.    for each child of node do
8.    eva= minimax(child, depth-1, alpha, beta, False)
9.   maxEva= max(maxEva, eva)
10.  alpha= max(alpha, maxEva)
11.   if beta<=alpha
12.  break
13.  return maxEva
14.
15. else                    // for Minimizer player
16.   minEva= +infinity
17.   for each child of node do
18.   eva= minimax(child, depth-1, alpha, beta, true)
19.   minEva= min(minEva, eva)
20.   beta= min(beta, eva)
21.    if beta<=alpha
22.   break
23.  return minEva
```

Working of Alpha-Beta Pruning:

Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

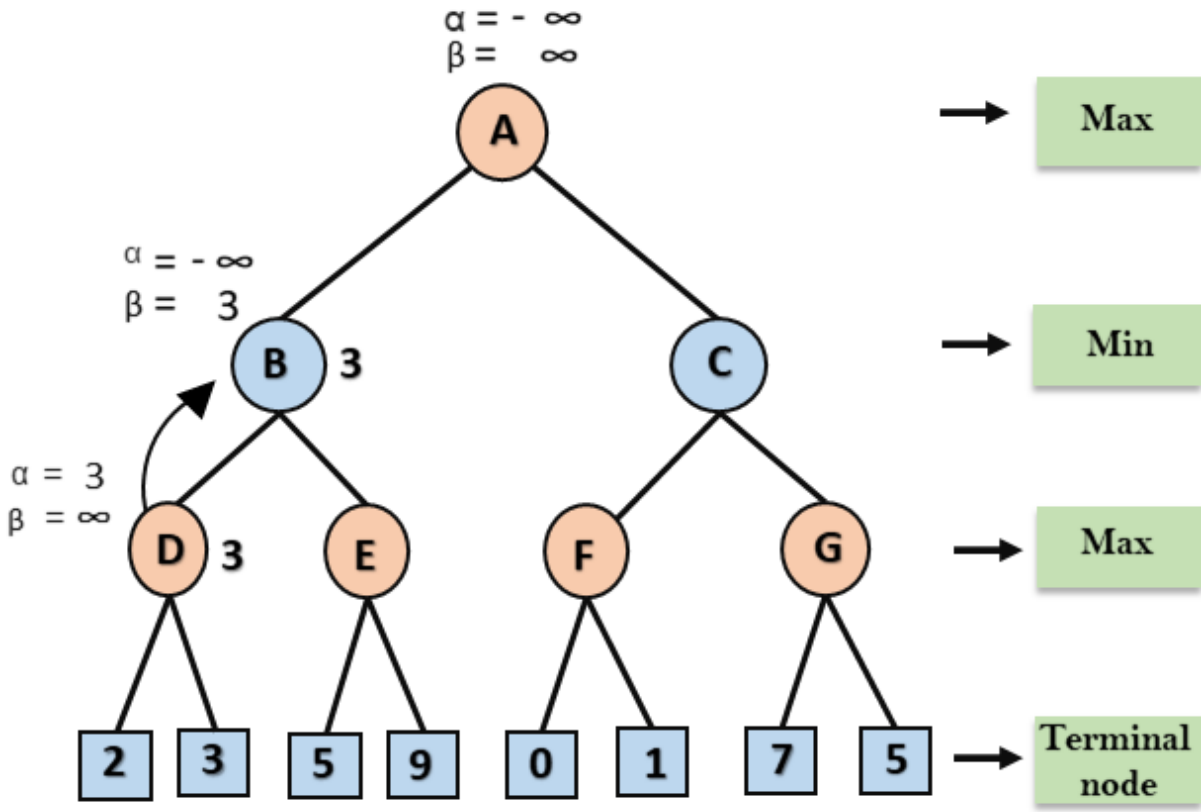
Step 1: At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

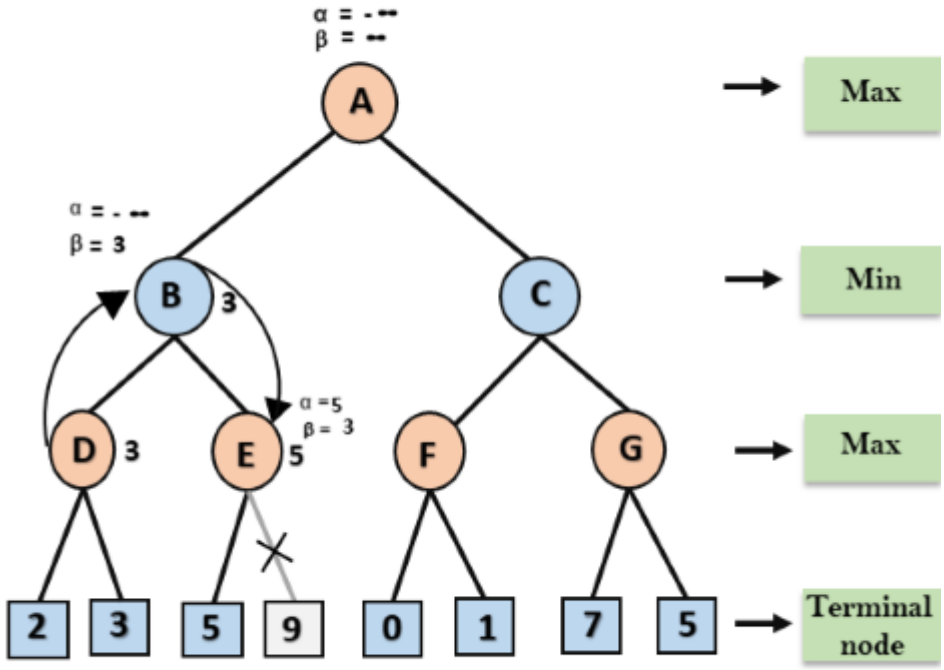
Backward Skip 10sPlay VideoForward Skip 10s

Step 3: Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.

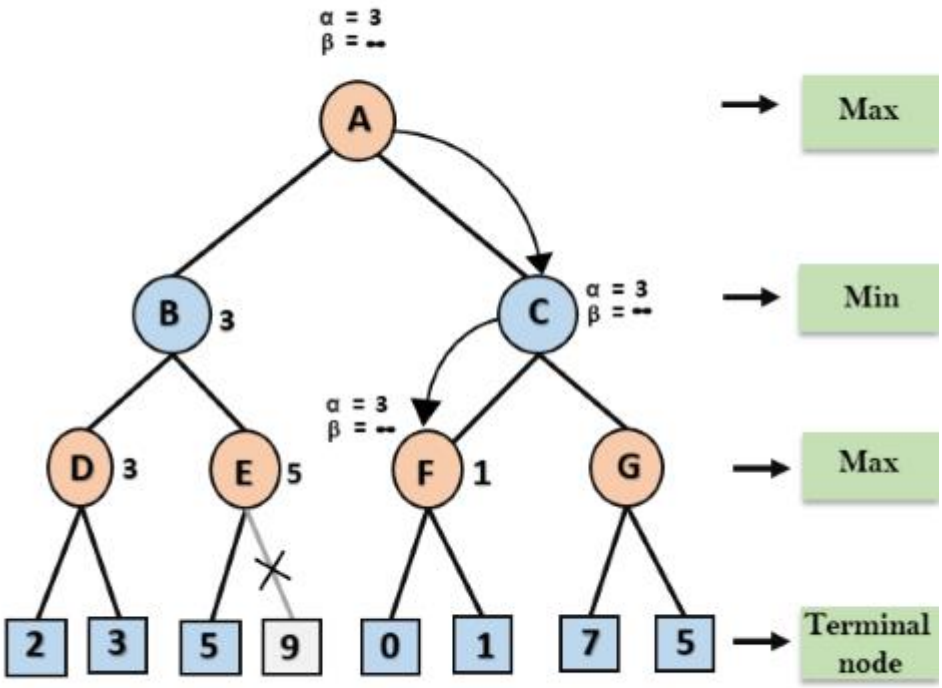


In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

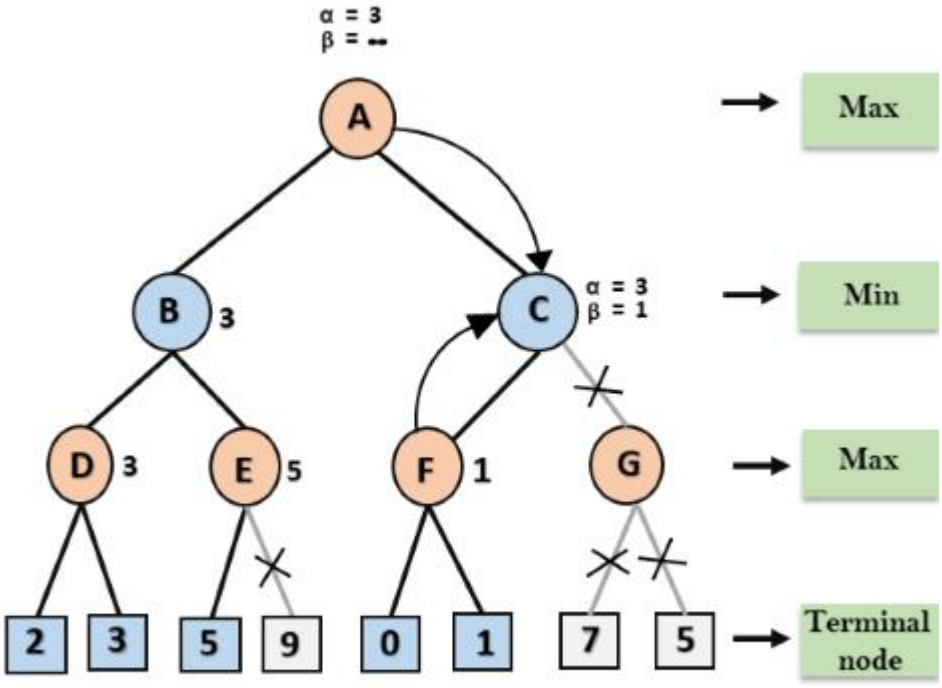
Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



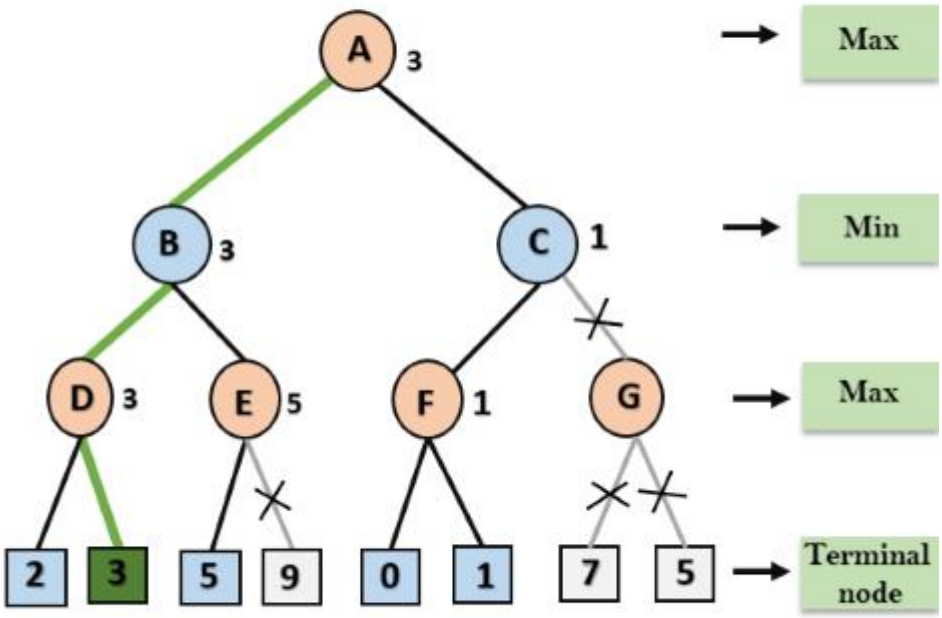
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C. At node C, α=3 and β= +∞, and the same values will be passed on to node F.
Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



Step 7: Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



ADVERTISEMENT

Move Ordering in Alpha-Beta pruning:

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering: In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is $O(b^m)$.**
- **Ideal ordering: The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is $O(b^{m/2})$.**

Rules to find good ordering:

Following are some rules to find good ordering in alpha-beta pruning:

- **Occur the best move from the shallowest node.**
- **Order the nodes in the tree such that the best nodes are checked first.**
- **Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.**
- **We can bookkeep the states, as there is a possibility that states may repeat.**

The constraint propagation algorithm is straightforward in principle. When the domain of a decision variable is modified, the constraints containing that variable are examined

to determine whether any values in the domains of other decision variables are now inconsistent.

To see how the propagation algorithm works with constraints, consider the constraint x <= y, written with the following fragment of code:

```
IloCPEngine cp;
...
IlcIntVar x(cp,0,3), y(cp,0,2);
cp.add( x <= y );
```

When this constraint is added to the optimizer, the invariant expressed in the previous section becomes active and reduces the domain of x by removing the value 3. For the moment, that is all that can be deduced from the constraint. Since the constraint has to be taken into account by CP Optimizer every time one of the variables in it is modified, the constraint itself is physically attached to these two variables.

CP Optimizer has been designed to automate and to optimize the reduction of the domains of decision variables. The CP Optimizer propagation algorithm for that purpose is straightforward in principle. When the domain of a decision variable is modified, the constraints containing that variable are examined to determine whether any values in the domains of other decision variables are now inconsistent. If this is the case, necessary domain reductions are carried out in turn.

The examination of the constraints incident on a variable is triggered by any modification of that variable. There are several different kinds of modifications, depending on the class of variable under consideration. Those modifications are referred to as propagation events.

For the class of decision variables, there are, in fact, these propagation events:

- **value means that a value has been assigned to the decision variable, that is, the variable has been fixed;**
- **range indicates that the minimum of the domain has increased or the maximum of the domain has decreased;**
- **domain indicates that the domain of the decision variable has been modified.**

When you define a new class of constraint, you must also define the propagation events for that class of constraint. You do so by means of the pure virtual member function, post.

Sometimes more than one event can be triggered after a variable modification. Specifically, the value event is always accompanied by the range and domain events. Likewise, a range event is always accompanied by a domain event.

Consider a search decision variable, *var*, with a domain containing only two values, *value1* and *value2*, where *value1* < *value2*. If you add a constraint that *var != value1*, three events are triggered:

- **the domain event is triggered since *value1* is actually removed from the domain of *var*;**
- **the range event is triggered since the minimal boundary of the domain of *var* has been increased;**
- **the value event is triggered since the variable is fixed by the reduction of its domain to a single value.**

Commutativity

CSPs are commutative.

- The order of any given set of actions has no effect on the outcome.
- Example: choose colors for Australian territories one at a time
  - [WA=red then NT=green] same as [NT=green then WA=red]
  - All CSP search algorithms consider a single variable assignment at a time ⇒ there are d^n leaves.

Backtracking search

Cfr. Depth-first search

Chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.

Uninformed algorithm

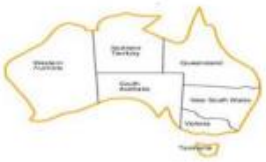
- No good general performance (see table p. 143)

Backtracking search

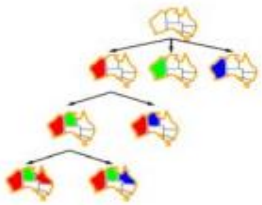
```
function BACKTRACKING-SEARCH(csp) return a solution or failure
    return RECURSIVE-BACKTRACKING({} , csp)

function RECURSIVE-BACKTRACKING(assignment, csp) return a solution or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp],assignment,csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment according to CONSTRAINTS[csp] then
            add {var=value} to assignment
            result ← RRECURSIVE-BACTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var=value} from assignment
    return failure
```

Backtracking example



Backtracking example



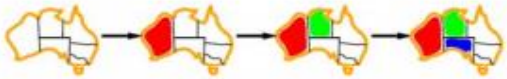
Improving backtracking efficiency

Previous improvements → introduce heuristics

General-purpose methods can give huge gains in speed:

- Which variable should be assigned next?
- In what order should its values be tried?
- Can we detect inevitable failure early?
- Can we take advantage of problem structure?

Minimum remaining values



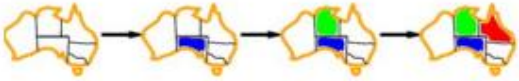
var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp],assignment,csp)

A.k.a. most constrained variable heuristic

Rule: choose variable with the fewest legal moves

Which variable shall we try first?

Degree heuristic



Use degree heuristic

Rule: select variable that is involved in the largest number of constraints on other unassigned variables.

Degree heuristic is very useful as a tie breaker.

In what order should its values be tried?

-

Constraint propagation is the process of communicating the domain reduction of a decision variable to all of the constraints that are stated over this variable.

Constraint propagation **is the process of communicating the domain reduction of a decision variable to all of the constraints that are stated over this variable. This process can result in more domain reductions. These domain reductions, in turn, are communicated to the appropriate constraints. This process continues until no more variable domains can be reduced or when a domain becomes empty and a failure occurs. An empty domain during the initial constraint propagation means that the model has no solution.**
For example, consider the decision variables *y* with an initial domain [0..10], *z* with an initial domain [0..10] and *t* with an initial domain [0..1], and the constraints

```
y + 5*z <= 4
t != z
t != y
```

over these three variables.
The domain reduction of the constraint *y + 5*z <= 4* reduces the domain of *y* to [0..4] and *z* to [0]. The variable z is thus fixed to a single value. Constraint propagation invokes domain reduction of every constraint involving *z*. Domain reduction is invoked again for the constraint *y + 5*z <= 4*, but the variable domains cannot be reduced further. Domain reduction of the constraint *t != z* is invoked again, and because *z* is fixed to 0, the constraint removes the value 0 from the domain of *t*. The variable *t* is now fixed to the value 1, and constraint propagation invokes domain reduction of every constraint

involving *t*, namely *t != z* and *t != y*. The constraint that can reduce domains further is *t != y*. Domain reduction removes the value 1 from the domain of *y*.
Constraint propagation is performed on constraints involving y; however, no more domain reduction can be achieved and the final domains are:

- ***y = [0 2..4]*,**
- ***z = [0]* and**
- ***t = [1]*.**

To invoke the constraint propagation process in CP Optimizer, the propagate function of the optimizer object is called. In the C++ API, this function is IloCP::propagate; in the Java™ API, this function is IloCP.propagate; and in the C# API, this function is CP.Propagate. This function invokes domain reduction on every constraint of the model and propagates the domain reductions. It returns true (IloTrue in the C++ API) if propagation succeeds; in other words, if no empty domains result. It returns false (IloFalse in the C++ API) otherwise.
As an example using the C++ API, a code that invokes propagation on the model above is;

```
IloIntVar y(env, 0, 10);
IloIntVar z(env, 0, 10);
IloIntVar t(env, 0, 1);
IloModel model(env);
model.add(y + 5*z <= 4);
model.add(t != z);
model.add(t != y);
IloCP cp(model);
if (cp.propagate()){
  cp.out() << " Domains reduced: " << std::endl;
  cp.out() << " Domain of y = " << cp.domain(y) << std::endl;
  cp.out() << " Domain of z = " << cp.domain(z) << std::endl;
```

```
  cp.out() << "Domain of t = " << cp.domain(t) << std::endl;
}else{
  cp.out() << " Model has no solution." << std::endl;
}
```

The call to the method IloCP::domain(IloIntVar x) is directed to an output stream to display the current domain of the decision variable *x*. Running this code produces the output:

```
Domains reduced:
Domain of y = [0 2..4]
Domain of z = [0]
Domain of t = [1]
```