UCLA COMPUTER SCIENCE FIELD NOTES

CS 161

Fundamentals of Artificial Intelligence

State-space search, heuristics, games, planning, logic, knowledge representation, and AI judgment

How these notes are written. I wrote each chapter the way I wish the material had been explained the first time: the real idea, the point where I usually get stuck, the simplest mental model that still stays honest, and a worked decision instead of a polished answer appearing from nowhere. Every chapter closes with practice and a fully written solution. If a sentence sounds impressive but does not help me solve or explain something, it does not belong here.

Daniel Mastick

Curriculum map, working notes, practice, and solutions

Course map

Week	Core thread	What should be solid by the end
1	**Agents, Lisp, and problem formulation**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
2	**Uninformed state-space search**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
3	**Heuristic search and A-star**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
4	**Adversarial search and games**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
5	**Constraint satisfaction and planning**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
6	**Predicate logic and inference**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
7	**Knowledge representation systems**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
8	**Uncertainty, perception, and responsible AI**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
9	**Integration workshop**	Connect at least three chapters in one end-to-end problem and explain where their contracts meet.
10	**Synthesis and project review**	Audit assumptions and explain a complete solution from interface to failure handling.

Scope anchor: https://catalog.registrar.ucla.edu/course/2025/COMSCI161. The sequence is aligned to the official catalog description and expanded into a practical ten-week study plan.

How I would use this packet

Treat each numbered section as one chapter. Read the formal explanation, pause at the student interpretation, and see whether the easy version still says the same thing. Rework the example without looking, then cover the solution in the chapter practice box. The cumulative set at the end is deliberately mixed: knowledge becomes durable when I have to choose the tool instead of being told which chapter I am in.

Contents

CHAPTER 1 Agents, Lisp, and problem formulation

The idea

An AI problem defines state, actions, transition model, initial state, goal test, and path cost. A rational agent chooses actions from percept history and objectives; Lisp makes recursive symbolic structures and functional decomposition explicit.

Rebuild the chapter from first principles

The claim I need to own is this: An AI problem defines state, actions, transition model, initial state, goal test, and path cost. A rational agent chooses actions from percept history and objectives; Lisp makes recursive symbolic structures and functional decomposition explicit. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of agents, lisp, and problem formulation. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach agents, lisp, and problem formulation on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from the course foundations to uninformed state-space search. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** An AI problem defines state, actions, transition model, initial state, goal test, and path cost.
- **Mechanism.** A rational agent chooses actions from percept history and objectives; Lisp makes recursive symbolic structures and functional decomposition explicit.
- **Boundary.** The useful test is whether I can apply agents, lisp, and problem formulation to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when agents, lisp, and problem formulation is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I spend more time defining a state that contains exactly the needed information than writing the search loop.

The easy version

An AI problem defines state, actions, transition model, initial state, goal test, and path cost.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: What makes a state representation too weak? The conclusion is Two histories that require different future decisions collapse into the same state. Add information needed by legal actions, goal tests, or costs, but avoid irrelevant

history.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. What makes a state representation too weak?
Solution. Two histories that require different future decisions collapse into the same state. Add information needed by legal actions, goal tests, or costs, but avoid irrelevant history.

Practice 2. Give a short oral explanation of agents, lisp, and problem formulation that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct agents, lisp, and problem formulation from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a agents, lisp, and problem formulation problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 2 Uninformed state-space search

The idea

Breadth-first, depth-first, uniform-cost, depth-limited, and iterative-deepening search differ in frontier order, completeness, optimality, time, and memory. Graph search additionally needs a reached-state policy.

Rebuild the chapter from first principles

The claim I need to own is this: Breadth-first, depth-first, uniform-cost, depth-limited, and iterative-deepening search differ in frontier order, completeness, optimality, time, and memory. Graph search additionally needs a reached-state policy. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of uninformed state-space search. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach uninformed state-space search on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from agents, lisp, and problem formulation to heuristic search and a-star. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Breadth-first, depth-first, uniform-cost, depth-limited, and iterative-deepening search differ in frontier order, completeness, optimality, time, and memory.
- **Mechanism.** Graph search additionally needs a reached-state policy.
- **Boundary.** The useful test is whether I can apply uninformed state-space search to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when uninformed state-space search is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I choose search by the guarantee and cost model, not by whichever traversal I remember first.

The easy version

Breadth-first, depth-first, uniform-cost, depth-limited, and iterative-deepening search differ in frontier order, completeness, optimality, time, and memory.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: When is breadth-first search optimal? The conclusion is When every step has equal positive cost, it explores states in nondecreasing path length, so the first goal has minimum cost. With unequal costs, use uniform-cost search.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. When is breadth-first search optimal?
Solution. When every step has equal positive cost, it explores states in nondecreasing path length, so the first goal has minimum cost. With unequal costs, use uniform-cost search.

Practice 2. Give a short oral explanation of uninformed state-space search that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that

licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct uninformed state-space search from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a uninformed state-space search problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 3 Heuristic search and A-star

The idea

A heuristic estimates remaining cost. Greedy best-first uses the estimate alone; A-star uses path cost plus heuristic. Admissibility supports optimality, while consistency supports graph search without reopening settled states.

Rebuild the chapter from first principles

The claim I need to own is this: A heuristic estimates remaining cost. Greedy best-first uses the estimate alone; A-star uses path cost plus heuristic. Admissibility supports optimality, while consistency supports graph search without reopening settled states. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of heuristic search and a-star. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.

4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach heuristic search and a-star on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from uninformed state-space search to adversarial search and games. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** A heuristic estimates remaining cost.
- **Mechanism.** Greedy best-first uses the estimate alone; A-star uses path cost plus heuristic.
- **Boundary.** Admissibility supports optimality, while consistency supports graph search without re-opening settled states.
- **Decision test.** I should be able to say when heuristic search and a-star is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

A heuristic is useful only if I can explain why it never invents impossible savings.

The easy version

A heuristic estimates remaining cost.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why is straight-line distance admissible for road distance? The conclusion is Any road route is at least the Euclidean distance between endpoints, so the estimate cannot exceed the true shortest road cost under nonnegative distance.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why is straight-line distance admissible for road distance?
Solution. Any road route is at least the Euclidean distance between endpoints, so the estimate cannot exceed the true shortest road cost under nonnegative distance.

Practice 2. Give a short oral explanation of heuristic search and a-star that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct heuristic search and a-star from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a heuristic search and a-star problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 4 Adversarial search and games

The idea

Minimax models alternating rational opponents over game trees. Depth limits need evaluation functions; alpha-beta pruning removes branches that cannot change the decision without changing the minimax result.

Rebuild the chapter from first principles

The claim I need to own is this: Minimax models alternating rational opponents over game trees. Depth limits need evaluation functions; alpha-beta pruning removes branches that cannot change the decision without changing the minimax result. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of adversarial search and games. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach adversarial search and games on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from heuristic search and a-star to constraint satisfaction and planning. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Minimax models alternating rational opponents over game trees.
- **Mechanism.** Depth limits need evaluation functions; alpha-beta pruning removes branches that cannot change the decision without changing the minimax result.
- **Boundary.** The useful test is whether I can apply adversarial search and games to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when adversarial search and games is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I write whose utility is optimized at every level; otherwise max and min become arbitrary labels.

The easy version

Minimax models alternating rational opponents over game trees.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: What does an alpha-beta cutoff prove? The conclusion is The branch cannot improve the choice already available to an ancestor. Its exact value is unnecessary because a rational player already has an option sufficient to reject it.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. What does an alpha-beta cutoff prove?
Solution. The branch cannot improve the choice already available to an ancestor. Its exact value is unnecessary because a rational player already has an option sufficient to reject it.

Practice 2. Give a short oral explanation of adversarial search and games that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct adversarial search and games from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a adversarial search and games problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 5 Constraint satisfaction and planning

The idea

Constraint-satisfaction problems define variables, domains, and constraints. Backtracking improves through ordering, propagation, and consistency. Planning represents actions through preconditions and effects over a state model.

Rebuild the chapter from first principles

The claim I need to own is this: Constraint-satisfaction problems define variables, domains, and constraints. Backtracking improves through ordering, propagation, and consistency. Planning represents actions through preconditions and effects over a state model. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays

fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of constraint satisfaction and planning. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach constraint satisfaction and planning on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from adversarial search and games to predicate logic and inference. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method

valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Constraint-satisfaction problems define variables, domains, and constraints.
- **Mechanism.** Backtracking improves through ordering, propagation, and consistency.
- **Boundary.** Planning represents actions through preconditions and effects over a state model.
- **Decision test.** I should be able to say when constraint satisfaction and planning is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I propagate consequences immediately after a choice so contradictions fail close to their cause.

The easy version

Constraint-satisfaction problems define variables, domains, and constraints.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why does minimum-remaining-values help backtracking? The conclusion is It chooses the most constrained variable, exposing dead ends early and reducing wasted search. Degree and least-constraining-value heuristics complement it.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why does minimum-remaining-values help backtracking?
Solution. It chooses the most constrained variable, exposing dead ends early and reducing wasted search. Degree and least-constraining-value heuristics complement it.

Practice 2. Give a short oral explanation of constraint satisfaction and planning that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct constraint satisfaction and planning from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects

and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a constraint satisfaction and planning problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 6 Predicate logic and inference

The idea

Propositional and first-order logic distinguish syntax, models, entailment, and proof. Unification, substitution, forward chaining, backward chaining, and resolution derive consequences under explicit assumptions.

Rebuild the chapter from first principles

The claim I need to own is this: Propositional and first-order logic distinguish syntax, models, entailment, and proof. Unification, substitution, forward chaining, backward chaining, and resolution derive consequences under explicit assumptions. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of predicate logic and inference. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach predicate logic and inference on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from constraint satisfaction and planning to knowledge representation systems. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Propositional and first-order logic distinguish syntax, models, entailment, and proof.
- **Mechanism.** Unification, substitution, forward chaining, backward chaining, and resolution derive consequences under explicit assumptions.
- **Boundary.** The useful test is whether I can apply predicate logic and inference to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when predicate logic and inference is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I separate what is true in every model from what one proof procedure happened to find.

The easy version

Propositional and first-order logic distinguish syntax, models, entailment, and proof.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: What is the difference between entailment and implication syntax? The conclusion is Entailment is semantic: every model of the premises satisfies the conclusion. Implication is a formula inside the language that can itself be true or false in a model.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. What is the difference between entailment and implication syntax?
Solution. Entailment is semantic: every model of the premises satisfies the conclusion. Implication is a formula inside the language that can itself be true or false in a model.

Practice 2. Give a short oral explanation of predicate logic and inference that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct predicate logic and inference from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a predicate logic and inference problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 7 Knowledge representation systems

The idea

Production rules, semantic networks, frames, scripts, ontologies, and logical facts encode different structures and defaults. Representation determines which queries are easy, which exceptions are awkward, and where ambiguity hides.

Rebuild the chapter from first principles

The claim I need to own is this: Production rules, semantic networks, frames, scripts, ontologies, and logical facts encode different structures and defaults. Representation determines which queries are easy, which exceptions are awkward, and where ambiguity hides. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of knowledge representation systems. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach knowledge representation systems on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from predicate logic and inference to uncertainty, perception, and responsible ai. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Production rules, semantic networks, frames, scripts, ontologies, and logical facts encode different structures and defaults.
- **Mechanism.** Representation determines which queries are easy, which exceptions are awkward, and where ambiguity hides.
- **Boundary.** The useful test is whether I can apply knowledge representation systems to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when knowledge representation systems is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

A knowledge structure is an interface for reasoning, so I test it with the questions the agent must answer.

The easy version

Production rules, semantic networks, frames, scripts, ontologies, and logical facts encode different structures and defaults.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why can default inheritance create

conflict? The conclusion is An entity may inherit incompatible defaults from multiple categories or an exception may override a general rule. The system needs precedence, defeasibility, or explicit context.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why can default inheritance create conflict?
Solution. An entity may inherit incompatible defaults from multiple categories or an exception may override a general rule. The system needs precedence, defeasibility, or explicit context.

Practice 2. Give a short oral explanation of knowledge representation systems that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct knowledge representation systems from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a knowledge representation systems problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 8 Uncertainty, perception, and responsible AI

The idea

Real AI systems combine search and knowledge with uncertain sensing, learning, planning, and human oversight. Evaluation must cover utility, calibration, robustness, distribution shift, misuse, explainability limits, and unequal error costs.

Rebuild the chapter from first principles

The claim I need to own is this: Real AI systems combine search and knowledge with uncertain sensing, learning, planning, and human oversight. Evaluation must cover utility, calibration, robustness, distribution shift, misuse, explainability limits, and unequal error costs. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of uncertainty, perception, and responsible ai. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach uncertainty, perception, and responsible ai on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from knowledge representation systems to the cumulative course model. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Real AI systems combine search and knowledge with uncertain sensing, learning, planning, and human oversight.
- **Mechanism.** Evaluation must cover utility, calibration, robustness, distribution shift, misuse, explainability limits, and unequal error costs.
- **Boundary.** The useful test is whether I can apply uncertainty, perception, and responsible ai to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when uncertainty, perception, and responsible ai is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

My shorthand for this chapter is: a benchmark score is one measurement in one environment, not a certificate of intelligence or safety.

The easy version

Real AI systems combine search and knowledge with uncertain sensing, learning, planning, and human oversight.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: An agent succeeds in simulation but fails after deployment. What should be audited? The conclusion is Compare state and observation distributions, simulator assumptions, sensor noise, action latency, reward alignment, feedback loops, subgroup failures, monitoring, and safe fallback behavior.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. An agent succeeds in simulation but fails after deployment. What should be audited?
Solution. Compare state and observation distributions, simulator assumptions, sensor noise, action latency, reward alignment, feedback loops, subgroup failures, monitoring, and safe fallback

behavior.

Practice 2. Give a short oral explanation of uncertainty, perception, and responsible ai that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct uncertainty, perception, and responsible ai from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a uncertainty, perception, and responsible ai problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

Final study check

I should be able to close this packet and reconstruct the core models, not merely recognize their names. My final check is to explain one complete problem aloud: what the inputs mean, which invariant or contract controls the work, why the method is legal, what it costs, how it can fail, and what evidence would convince another engineer.