An Integrative Framework for Artificial Intelligence Education

Pat Langley
Institute for the Study of Learning and Expertise,
2164 Staunton Court, Palo Alto, CA 94306 USA
Department of Computer Science, University of Auckland,
Private Bag 92019, Auckland 1142 NZ

Abstract

Modern introductory courses on AI do not train students to create intelligent systems or provide broad coverage of this complex field. In this paper, we identify problems with common approaches to teaching artificial intelligence and suggest alternative principles that courses should adopt instead. We illustrate these principles in a proposed course that teaches students not only about component methods, such as pattern matching and decision making, but also about their combination into higher-level abilities for reasoning, sequential control, plan generation, and integrated intelligent agents. We also present a curriculum that instantiates this organization, including sample programming exercises and a project that requires system integration. Participants also gain experience building knowledge-based agents that use their software to produce intelligent behavior.

1 Background and Motivation

Modern students of artificial intelligence are being poorly served. Introductory courses focus on topics that are well formalized and easy to teach, and they are usually communicated as a set of unrelated problems. These classes omit many of the discipline's most basic and important ideas, despite their continued relevance. The past 30 years have seen substantial progress, but similar conceptual advances occurred in physics, chemistry, and biology without those disciplines dropping their established content. AI is in serious danger of raising entire generations of researchers and practitioners with no training in the field's history or its many well-understood, successful methods.

Moreover, courses seldom champion the idea that some mental abilities build on others in a cumulative manner or the importance of integration in developing intelligent systems. Students typically gain experience with using existing software, but few of them ever learn how to design and implement such systems on their own. Mainstream introductory classes train people to become consumers of AI technology rather than producers, which does not bode well for the discipline's future. Increased emphasis on statistics and machine learning, which often replace traditional topics, has made the situation even worse.

In this paper, we analyze this crisis in AI instruction and propose an alternative approach to presenting content about the field. In the next section, we review the state of introductory courses in this area and drawbacks of the current style. After this, we propose some principles for introductory education in the area that address these issues. After this, we describe the structure and content for a new type of AI course, discussing how each module responds to the principles. The narrative emphasizes the cumulative, system-level character of intelligence, starting with low-level tasks like matching patterns and making decisions, and proceeding to integrated agents that combine inference, execution, and planning. We also discuss how traditional topics like natural language, vision, uncertainty, and learning relate to our framework, along with the importance of history and the target audiences for our innovative course.

2 The State of the Art

As noted earlier, we believe that existing introductory AI courses do not train students in the concepts or skills they need to develop intelligent systems. However, before we devote substantial effort to devising a new curriculum, we should first examine the way that classes are currently taught and determine their adequacy. To this end, we examined the Web sites for undergraduate AI courses at the ten major universities shown in Table 1. Most described classes offered in 2018, but one occurred the year before. In each case, we inspected the course schedule and associated exercises, although for a few the latter were unavailable. With two exceptions, they used Russell and Norvig's (2009) textbook and covered its topics, although often in different orders. Most schedules included multiple sessions on search and game playing, constraint satisfaction and logical reasoning, probabilistic inference, and varieties of statistical learning.

Analysis of information on these ten courses revealed a number of themes that, we believe, are reasons for concern:

- *Students learn about AI as a collection of isolated algorithms*, with little attention to how they can contribute to integrated intelligent systems. This gives many people the impression that all important results in the field reside at the level of these component methods, that their integration involves 'mere' application, and that the study of 'AI systems' is effectively an oxymoron.

- *There is little emphasis on representing domain content*. For example, search algorithms are presented in the abstract, with limited discussion of how to encode states or operators to generate them. Students learn about different formalisms, like logic and Bayesian networks, but get sparse practice at encoding knowledge in them, which can greatly impact effectiveness of processing.
- *Courses seldom convey the cumulative character of the field*, in which high-level representations and their associated mechanisms build directly on lower-level structures. As with representation, this comes from the bias toward abstract algorithms. For instance, problem solving builds naturally on pattern matching – to find relevant operators – and decision making – to select among choices, but the latter are usually taught *after* search methods. Integrated systems rely on layered integration of such mechanisms.
- *Exercises often rely on software packages or partial solutions* that students can treat as black boxes. In some cases, students must only download packages and run them on input files with different options; in others, they must insert a small amount of code into programs provided to them. This approach makes exercises easier to grade, but it does not provide participants with deep understanding of their operation, and it certainly does not teach them how to create such methods themselves.
- *Courses often omit important topics and key theoretical ideas* that have contributed much historically to computational accounts of intelligence. Problem areas like qualitative reasoning, analogy, and creativity are ignored in favor of ones that are more easily formalized. Even foundational AI concepts, such as list processing, satisficing, and expert systems are in danger of being forgotten.

Table 1 summarizes how each course fares along these dimensions, based on inspection of their on-line schedules and exercises, with ○, ⊙, and ● denoting poor, medium, and good scores, respectively. The situation supports the concerns expressed earlier that introductory AI courses downplay integration, representation, cumulative presentation, programming, and breadth. One especially narrow course focused primarily on statistical learning, almost to the exclusion of other topics. Naturally, our analysis is subjective and based on limited information, but we predict others would draw similar conclusions from the content available. At the same time, we expect many AI educators would disagree that low scores on these criteria are undesirable. They are likely to believe that presenting the field as a collection of algorithms, using available software, and ignoring 'outmoded' topics are evidence of its maturity, not a cause for dismay.

There are multiple reasons why this perspective is widely adopted, the most basic being inertia. The standard textbook makes it easy to teach AI in this manner, and instructors who have done so many times are reluctant to change gears. Another is that AI's home in computer science departments, most of which grew out of mathematics, have a strong bias toward abstract analysis at the algorithm level. This history also mitigates against inclusion of topics that are associated with cognitive psychology, which is often viewed as less respectable. A third reason is that AI applications often emphasize one capability, which is integrated with other information technology but not with other aspects of intelligence. Finally, the increasing availability of software packages makes it easy for students to produce empirical results without learning how to recreate their abilities. Taken together, these trends have damaged the field, leading to graduates who have neither deep understanding of AI principles or the ability to develop integrated intelligent systems.

Table 1: Sample AI courses and their evaluations – poor (○), medium (⊙), good (●), unknown (∗) – on five criteria: integration (I), representation (R), cumulative style (C), programming (P), and breadth (B). Course sites appear in Appendix.

	I	R	C	P	B
Carnegie Mellon	○	⊙	○	○	⊙
Georgia Tech	○	○	○	○	○
MIT	○	∗	○	⊙	⊙
Stanford	○	○	○	○	○
UC Berkeley	○	○	⊙	○	○
UCLA	○	∗	○	○	○
U. Maryland	○	⊙	∗	∗	⊙
USC	○	∗	○	○	⊙
UT Austin	∗	⊙	⊙	○	⊙
U. Washington	○	○	⊙	○	○

3 Principles for AI Instruction

Now that we have identified some drawbacks of modern AI courses, we can consider ways to teach them differently. Here we propose a set of principles for selecting and organizing instructional content, each responding to a problem identified above. We maintain that AI classes should:

- *Champion a systems perspective that shows how mechanisms interact to produce intelligence*. This will combat views that AI is a collection of disconnected algorithms.
- *Give students experience with encoding representational content* that mechanisms interpret to produce behavior. This will clarify the centrality of structured representations in intelligent agents.
- *Present topics in a cumulative manner, with later material layered on earlier content*, much as calculus builds on algebra, which draws on arithmetic. This will emphasize the hierarchical character of intelligence.
- *Teach students not only how to use AI methods, but how to construct them from simpler components*. This will give them the ability to develop their own mechanisms when existing ones do not suffice.
- *Cover important abilities exhibited in human intelligence* even when they are difficult to formalize. Linking AI to psychology will remind students that the two fields address many of the same core phenomena.

We believe that organizing content in accordance with these five principles will counter the widespread belief that AI is simply a set of engineering tricks and it will better prepare students for innovative research and practical applications.

This approach to instruction is very different from that adopted by most courses on artificial intelligence. Some textbooks (e.g., Russell and Norvig, 2009) introduce the notion of intelligent agents, but they lose this message as they proceed. The standard organization focuses first on search algorithms, then treats representation and reasoning as afterthoughts, with sections on uncertainty and learning connecting weakly to earlier material. This framework dates back over two decades to outlines by Korf (1994) and by Russell and Norvig (1994). Nilsson (1994) and Kowalski (2011) have proposed narratives closer to our own, but neither of their approaches has been widely adopted.

There is some evidence that AI can be taught effectively in the manner we have proposed. In the late 1980s, we designed and offered a graduate course that adopted some of these ideas. Students implemented, in sequence, a relational pattern matcher, a production-system interpreter, and a forward-chaining search engine, with later programming exercises using results from previous ones as subroutines. Moreover, after writing and testing each module, students used them as high-level programming languages to create knowledge-based systems for related tasks, including categorization, sentence parsing, and puzzle solving. The course did not follow all the principles above, but its success provides support for an alternative way to structure AI education. Over the past decade, we have used a similar cumulative approach to lectures in introductory classes, relying on an existing agent architecture for exercises. Together, these experiences suggest that the proposed curriculum is viable.

4 An Integrated Introduction to AI

In this section, we outline the contents for an introductory AI course that follows the principles introduced earlier. Such a class cannot cover all topics in such a broad field. For this reason, we will focus on a subset of the core abilities that Langley, Laird, and Rogers (2009) identified in their survey of cognitive architectures. For each topic, we specify the generic task in terms of inputs and outputs, consider representations for these structures, and note key theoretical ideas. We also describe a programming assignment that would require students to implement the basic functionality, variations on these mechanisms, and demonstrate their software on test cases. Students would also develop knowledge bases themselves and run their module on them to gain experience with its use as a high-level programming language.

4.1 Recognition and Pattern Matching

One of the most fundamental abilities of an intelligent agent is recognizing an instance of some category, concept, or pattern. We can state this task as:

- *Given:* A pattern that describes some class of situations;
- *Given:* A description of some specific situation;
- *Find:* All ways in which the pattern matches the situation.

Detecting instances of matched patterns enables many more sophisticated forms of processing, as we will see later. The key theoretical ideas here are (1) the central role of patterns and pattern matching in AI, (2) the relational character of many situations and patterns that match against them, and (3) a pattern's ability to match a situation in multiple ways.

For the programming assignment, students would implement a pattern matcher that works with a predicate logic or frame notation. Each situation would be stated as a conjunction of relational ground literals with constants as arguments. Patterns would take a similar form but might replace some constants with variables and could include negated conditions. The match process would produce zero or more pattern matches or instantiations, each specifying the matched situation elements and a set of bindings between variables and constants. Subroutines would include modules for unifying a single condition with a single situation element and another for finding all matches between a single condition and a set of elements.

Test cases would include checking conditions on moves for simple puzzles like the Tower of Hanoi, goal descriptions for simple games like Tic-Tac-Toe, and spatial relations between objects in a perceived environment. In addition, students would gain experience with using their software by developing patterns for tasks like recognizing triangles in line drawings, recognizing prepositional phrases in word sequences, and noting temporal relations in the Allen calculus. At least some of these tasks should require a mixture of symbolic and numeric descriptions, as well as patterns that incorporate negated conditions.

4.2 Decision Making and Choice

Another key feature of an intelligent system is its capacity for *choice*, that is, to decide among a number of alternatives. We can state this problem as:

- *Given:* A set of entities and associated descriptions;
- *Given:* A set of goals and/or evaluation criteria;
- *Find:* A subset of the original entities to select.

Like pattern matching, the ability to select among such alternatives forms the basis for more complex aspects of intelligence. This leads naturally to the notion of agency, since it lets one choose actions to carry out. The important theoretical ideas are that (1) agents must generate candidates from among available entities, (2) they must evaluate and select among options, and (3) such decisions are influenced both by characteristics of alternatives and by agents' objectives.

For the programming assignment, students would implement a choice mechanism that accesses or calculates one or more scores for each choice, then uses them to rank alternatives and make a final selection. We assume that candidates are encoded as objects or entities with associated relational and numeric descriptions, and that evaluation criteria are specified in terms of the latter. Subroutines would include modules for recognizing patterns to generate options, calculating one or more scores for each candidate, and deciding on a subset of these choices. Parameters would control details of the generation, ranking, and selection processes. The decision-making mechanism would produce a subset of the original entities, each with one or more associated scores.

Test cases would include selecting among categories using a numeric utility function and choosing among patterns

based on recency of matched elements and on pattern specificity. In addition, students would gain experience with uses of decision making by developing their own candidate sets and objectives. Problems would include deciding which legal move to take in a puzzle based on heuristic criteria, assigning entities to categories using a relational probabilistic classifier, and recommending items based on user profiles.

4.3 Conceptual Inference and Reasoning

Matching individual patterns and making single decisions serve as solid foundations for mental abilities, but we typically reserve the term 'intelligence' for higher-level processing. These require some form of *composition* that combines elementary structures to produce new mental or physical states. Conceptual inference and reasoning is one capability that depends on this idea. We can state this task as:

- *Given:* A set of knowledge elements encoding expertise;
- *Given:* A set of beliefs that describe some situation;
- *Given:* An optional query to answer or goal to achieve;
- *Find:* A set of reasoning chains that connect these facts (and optionally the query) through the knowledge.

The core theoretical postulates are that (1) inference constructs proof-like structures that link beliefs and queries, (2) there is a space of candidate structures, only some of them viable, and (3) one must search this space to find solutions.

For the main assignment, students would write software for deductive reasoning that, like Prolog (Clocksin and Mellish, 1981), carries out AND/OR search through a space of proof trees. Representations would be similar to those in Prolog, with knowledge encoded as relational rules. Typical rules would have a single consequent and multiple antecedents, but headless rules could specify constraints on relations that cannot occur together. Queries would be conjunctions of relations with either variables or constants as arguments. Subroutines would unify rule heads with queries and antecedents with beliefs, as well as decide which elements and rules to focus on during search. Parameters would specify criteria to use when choosing elements and rules, limits on search, and desired number of solutions.

Basic test cases would include answering queries about relationships in a kinship database and proving geometry theorems from available postulates. For more advanced demonstrations, students would create and test knowledge bases for sentence processing with context-free grammars and reasoning about complex spatial relations among objects in simulated environments. A follow-on exercise would require extending the code to support abductive reasoning and then applying it to parsing ill-formed sentences and inferring other agents' plans from their observed actions, possibly augmenting rules and beliefs with probabilities.

4.4 Execution and Sequential Control

Intelligent agents exist over time, and thus encounter a continuing sequence of choices. In each case, they must generate alternatives, describe these candidates, and select one or more for execution. One obvious approach to sequential decision making is to invoke this process repeatedly. This leads to the reactive control task, which we can state as:

- *Given:* A set of knowledge elements that describe conditional effects of candidate actions;
- *Given:* A set of goals and/or evaluation criteria;
- *Given:* A description of the agent's current situation;
- *Find:* One or more action instances to carry out and the changes they are expected to produce.

The agent applies this scheme iteratively to produce a sequence of actions over time. Most applications involve extended activity in a physical or simulated environment, but they might also involve mental simulation. The theoretical tenets here are that control (1) combines knowledge and beliefs about the situation to generate candidate actions, (2) uses knowledge about the agent's objectives to evaluate these alternatives, and (3) selects a subset of actions and carries them out before continuing the process.

For the programming assignment, students would implement a procedure that repeatedly matches, selects, and executes an action, continuing for a specified number of cycles or until no rules apply.[1] Subroutines would draw from earlier units on pattern matching and decision making, with parameters controlling details of these component processes. Situations would be encoded as sets of relations that specify the agent's beliefs about the environment, along with goals or tasks it wants to pursue. Each action would have an associated rule that describes expected effects when taken under specified conditions (Fikes and Nilsson, 1972; McDermott et al., 1998). Conditions and effects would be symbolic structures, but might include quantitative information (Fox and Long, 2003). On each cycle, the system would select and execute a matched rule based on specified criteria.

Test cases would include simple physical tasks, such as controlling a first-person game agent and letting a simulated robot approach objects while avoiding obstacles. A follow-up assignment would involve extending the software to modulate reactive control using inference and task-directed processing. Here students would use hierarchical task networks (Nau et al., 2003) to produce more sophisticated behavior for games and simulated robotic tasks, such as collecting and assembling a set of objects. For a related problem, they would handcraft hierarchical plans that the agent executes in its environment. These would clarify that control is not limited to purely reactive methods and can benefit from knowledge.

4.5 Planning and Problem Solving

Executing complex behaviors over time is an essential part of agency, but it also occurs in many animals that we are reluctant to call intelligent. Humans can also solve novel problems and generate innovative plans that achieve their goals, which involves another variety of compositional processing. We can state this important task as:

- *Given:* Knowledge about conditional effects of actions;
- *Given:* A set of goals and/or evaluation criteria;

[1] In essence, this would involve creating a production system interpreter (Newell, 1973; Brownston et al., 1985).

- *Given:* A description of the agent's current situation;
- *Find:* A set of plans (action sequences) that transform the current state into one satisfying the agent's goals.

The planning task has much in common with sequential control, but there are three key differences. These include the theoretical ideas that plan generation (1) involves mental simulation of action sequences (i.e., plans) rather than execution in the environment, (2) requires the agent to search through a space of alternative plans, and (3) uses heuristics to guide this search and make it tractable.

Programming exercises would require students to implement a planning system that searches through a space of candidate plans. This would select a node (partial plan) in the search tree, select an operator to extend it, check to see whether it fails or solves the problem, and iterate until finding enough solutions or exhausting allocated resources. Subroutines would include modules developed previously for pattern matching, decision making, inference, and execution. Parameters would control the heuristics used to select nodes and operators, the details of backtracking, and the number of solutions desired. The module would assume the same formalism as sequential control, with relations describing situations and goals, and with rules characterizing actions. The latter may be organized in a hierarchy that states how to decompose complex tasks into simpler ones.

Test cases would repeat the tasks used for sequential control, but with weaker heuristic guidance to illustrate the role of search. Additional problems would come from classic tasks on which first-principles planning is effective. Students would also use the hierarchical task networks they wrote for the control module to constrain the search process. As before, advanced problems would include controlling an agent in a first-person game and having a mobile robot collect and assemble objects. The latter task would involve an interesting mixture of symbolic and numeric processing.

4.6 Integrated Intelligent Agents

Humans combine each of the abilities discussed earlier, and developing integrated intelligent agents was originally one of the field's primary objectives. Research on cognitive architectures (Langley et al., 2009) has emphasized this idea, but so has a parallel body of work on integrated robotics systems. We cannot specify this task in terms of inputs and outputs because it relies on feedback between modules, but the basic aim is an agent that interacts with its environment over an extended period in a goal-directed manner. The important theoretical ideas are that (1) such intelligent agents draw on inference, planning, and execution, (2) they monitor the progress of plans as they are carried out, and (3) they detect environmental anomalies when they arise and revise their plans in response. There are many variations on this underlying theme, but they share core assumptions.

For this module's programming assignment – actually a small project – students would create an agent architecture that supports such integrated, goal-directed processing. Subroutines would include earlier software for inference, plan generation, and sequential execution. Parameters would include those from previous assignments, along with criteria that drive detection of anomalies (pattern matching) and choosing when to replan (decision making). The project would require no new representational formalisms, as it would build directly on those introduced earlier for various components. Instead, participants would focus on integrating previous capabilities to support extended behavior in pursuit of the agent's goals, which themselves might change over time (Aha, Cox, and Muñoz-Avila, 2013).

Students would choose from challenge domains that involve cognitive robotics, diagnosis and repair of machinery, and task-oriented dialogue. Agents would operate in simulated environments that, unlike those for planning, can violate their expectations. One example would involve controlling a simulated robot in settings where objects appear or move unexpectedly, where the robotic platform malfunctions, or where other surprises occur. Similar challenges would arise in the diagnostic domain, where machine components break down, and in task-oriented dialogue, where another participant introduces new topics. In each case, students would provide the architecture with knowledge that lets the agent pursue and achieve goals over extended periods in environments that do not always behave as it predicts.

5 Course Schedule and Assignments

We have used this organization to develop a course schedule, shown in Table 2, that lists the topics covered in each session. These are grouped into five sections – introductory concepts, inference and reasoning, execution and control, planning and problem solving, integrated systems, and advanced subjects. Each topic emphasizes abstract computational tasks, such as deductive reasoning and heuristic search, and methods for solving them, but lectures will include examples of specific applications, such as parsing sentences and playing games. Ideas in later sections build on ones presented earlier, conveying the cumulative character of cognition. The section on integrated systems brings this content together and illustrates them in the context of three important types of application.

Programming assignments also play a major role in the course. Table 3 summarizes eight exercises, approximately one per week, associated with the first four sections. These require writing generic software (e.g., deductive reasoning) and demonstrating it on specific domains (e.g., parsing sentences). This is an aggressive schedule, but later exercises build on the results from earlier ones, letting students reuse existing code. Also, exercises come in pairs, with the second involving straightforward extensions of its predecessor. During the final weeks, participants use this software base to create and demonstrate an integrated system for cognitive robotics, diagnosis and repair, or task-oriented dialogue. This takes place in parallel with sessions on advanced topics that have no associated exercises.

6 Supporting Discussion

We have argued that our approach to introductory AI education is both more integrative and more inclusive than standard courses, but these claims merit discussion. Of course, many readers will maintain that standalone treatment of al-

Table 2: Proposed schedule for the introductory AI course.

Introductory Concepts

1. Intelligence in Humans and Machines
2. Representation, Reasoning, Search, and Knowledge
3. Recognition and Pattern Matching
4. Decision Making and Choice

Conceptual Inference and Reasoning

5. Multi-Step Inference
6. Deductive Reasoning
7. Satisfying Constraints
8. Qualitative and Causal Reasoning
9. Abduction and Explanation
10. Analogical Reasoning

Execution and Sequential Control

11. Reactive Control
12. Cognitive and Hierarchical Control
13. Executing and Monitoring Plans

Planning and Problem Solving

14. Problem Solving as Search
15. Heuristic Guidance
16. Generating Plans
17. Adversarial Problem Solving

Integrated Systems

18. Cognitive Architectures / Integrated Agents
19. Application: Cognitive Robotics
20. Application: Automated Diagnosis and Repair
21. Application: Task-Oriented Dialogue Systems

Advanced Topics

22. Episodic Memory and Self Explanation
23. Creativity and Discovery
24. Emotion and Personality
25. Moral Reasoning
26. Review and Summary

Table 3: Proposed programming assignments for the course.

1a. Implement a pattern matcher for predicate logic or frame notation. Test on patterns for puzzles and games, spatial and temporal relations, and word sequences.
1b. Extend the pattern matcher to generate, evaluate, and choose candidates for some decision. Test on selecting moves in games and assigning entities to categories.
2a. Implement a query-driven deductive reasoner for predicate logic. Test on answering queries about kinship, proving geometry theorems, and parsing sentences.
2b. Extend the deductive reasoning engine to support abductive inference. Test on parsing ill-formed sentences, line drawings, and understanding others' plans.
3a. Implement a reactive controller that matches rules, selects candidates, and executes them in an environment. Test on simulated game agents and mobile robots.
3b. Extend the reactive system to include inference about situations and top-down hierarchical control. Test on controlling simulated game agents and mobile robots.
4a. Implement a system that carries out search to find for plans that achieve goals. Test on classic planning tasks, game agents, and simulated mobile robots.
4b. Extend the planning system to use inference about situations and hierarchical task knowledge. Test on classic tasks, game agents, and simulated mobile robots.

gorithms is natural and follows a long tradition in other areas of computer science. Some will even argue that the notion of 'AI systems' is a distraction that has no place in formal instruction. Yet the field's most visible success stories – such as SHRDLU (Winograd, 1972), TRAINS (Allen et al., 1996), TacAir-Soar (Jones et al., 1999), Façade (Mateas and Stern, 2005), Watson (Ferrucci et al. 2010), and AlphaGo (Silver et al., 2016) – all revolve around integrated intelligent systems. We believe it is important to train students in the design and construction of such computational artifacts.

Some readers may be concerned that the curriculum ignores progress in AI over the past two decades. Indeed, there have certainly been important breakthroughs in reasoning, such as answer set programming (e.g., Baral, 2003) and statistical relational inference (e.g., Richardson and Domingos, 2006), as well as in planning and game playing, such as Fast-Forward (Hoffmann, 2001) and Monte Carlo tree search (Coulom, 2006). However, this does not mean they are desirable topics for an introductory course, where novices will gain more insight from classic methods precisely because they are conceptually simpler. More sophisticated AI techniques are better reserved for advanced classes, just as physics students learn about Einsteinian relativity after they have mastered Newtonian mechanics.

Other readers will note that we have not included major modules on language or vision, both of which receive entire chapters in Russell and Norvig (2009), and this appears to counter our claims for inclusiveness. However, our course content does include these topics, just not as top-level categories. We hold that language understanding is best viewed as a form of conceptual inference, while its generation is naturally handled as a variety of sequential execution and planning. Dialogue is a more complex process that combines understanding and generation, as well as extending common ground over time, which we address in the section on integrated intelligent systems.

In the same way, image understanding involves multi-level inference that aims to explain sensory data, while active vision relies on a form of sequential control. The exercises include test problems for basic versions of these abilities, but the course does not treat them as standalone topics that require entirely new mechanisms. Rather, students use their own software for inference and execution to create knowledge-based systems that demonstrate these capacities. We have not included modules for early image processing, including object detection, as these arguably involve low-level pattern recognition rather than high-level intelligence.

We have also downplayed two other topics – uncertainty and learning – to which Russell and Norvig (2009) devote entire sections of their book, as do other authors. Our argument for this decision is somewhat different. We maintain that probabilistic reasoning, and other responses to uncertainty, are best treated as modulations of classical symbolic approaches. Thus, appropriate exercises might ask students to extend basic techniques, from pattern matching to planning, to incorporate probabilities, but we should not present them as standalone abilities, as that would only buttress views that AI is a collection of isolated techniques.

The reasoning against distinct modules for machine learning is similar. As Langley (1996) has argued, one cannot describe or understand a learning mechanism until one has first described and understood the representations over which it operates or the performance processes it aims to improve. Learning methods can aid each cognitive function that our modules address, but they are best treated in that context, not as separate topics in their own right. This follows from our principle that instruction should be cumulative, as learning methods build directly on more basic performance elements. Realistically, this might be accomplished best in a follow-on course, organized along the same lines, that reviews the main topics and covers approaches to learning for each of them.

Another important issue concerns the background and origin of ideas covered in the course. We believe it is important to present problems and solutions in their historical context, and to give credit to other fields that have influenced AI's trajectory. Some textbooks acknowledge this intellectual debt, especially to mathematical disciplines like logic and probability theory, but they often ignore the crucial role played by cognitive psychology in AI's first three decades (Langley, 2012). Discussing the origin of key ideas, and giving credit where it is due, need not take much time, but it can enrich considerably the overall educational experience.

We should also discuss the target audience for the introductory course we have outlined. Although the basic organization and content are appropriate for an intensive upper-division class, there is no reason why an instructor could not offer the same basic material to lower-division undergraduates. The content's cumulative character means the only prerequisites would be an ability to design, write, and debug programs in a language like Lisp, Python, or Java that supports list processing. We can even imagine a version for undergraduates not majoring in computer science, or high-school students, that provides code for each module and uses them to create knowledge-based systems. In summary, the course's structure and content can be easily adapted to a wide range of audiences, which seems highly desirable.

7 Closing Remarks

In this paper, we identified problems with standard introductions to artificial intelligence and proposed some principles that, if adopted, would overcome them. These included championing a systems perspective that emphasizes integration, presenting topics in a cumulative manner, focusing on representational issues, teaching students not only how to use AI methods but how to construct them, and covering a broad range of abilities associated with human intelligence. We also described a novel introductory course that would follow these principles. Early assignments on pattern matching and decision making would support later ones on conceptual inference, sequential control, and plan generation, which in turn would enable integrated goal-directed agents that operate over extended periods. Students would write generic interpreters for each cognitive ability and then use them to construct knowledge-based systems that demonstrate their generality on different applications.

The course organization would differ substantially from mainstream treatments of the field, including those in the most popular textbooks. Topics like natural language and vision would be treated as special cases of more general abilities like inference and control, while approaches to dealing with uncertainty would be handled as modulations of classic symbolic methods. Material about machine learning would be reserved for a follow-on course that students could take after they had mastered more basic issues related to representation and performance. The course would discuss problems and solutions in their historical context, noting cases in which ideas originated in other fields, and its content could be easily adapted for presentation to nonmajors and high-school students by stressing the use of generic software rather than its generation. We predict that such a system-level, cumulative approach to AI instruction will better prepare future researchers and practitioners than the algorithm-centered schemes that are currently in vogue.

Acknowledgements

This research was supported in part by Grants N00014-15-1-2517 and N00014-17-1-2434 from the Office of Naval Research, which are not responsible for its contents.

Appendix: Sample AI Courses

The analysis in Section 2 comes from inspection of the Web sites for these ten recently offered AI courses:

Carnegie Mellon University
http://www.cs.cmu.edu/∼./15381/
Georgia Institute of Technology
https://www.cc.gatech.edu/∼riedl/classes/2017/cs3600/
Massachusetts Institute of Technology
https://ai6034.mit.edu/wiki/index.php?title=Main_Page
Stanford University
http://web.stanford.edu/class/cs221/
University of Maryland, College Park
http://www.cs.umd.edu/class/spring2018/cmsc421/
University of California, Berkeley
https://inst.eecs.berkeley.edu/∼cs188/fa18/
University of California, Los Angeles
web.cs.ucla.edu/∼guyvdb/teaching-service/cs161/2018w/
University of Southern California
http://idm-lab.org/wiki/360-Fall18/
University of Texas at Austin
https://www.cs.utexas.edu/users/risto/cs343/
University of Washington
https://courses.cs.washington.edu/courses/cse473/18au/

References

Aha, D. A.; Cox, M. T.; and Muñoz-Avila, H. eds. 2013. *Goal Reasoning: Papers from the ACS Workshop*. Baltimore, MD.

Allen, J. F.; Miller, B. W.; Ringger, E. K.; and Sikorski, T. 1996. A Robust System for Natural Spoken Dialogue. *Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics*, 62–70.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge, UK: Cambridge University Press.

Brownston, L.; Farrell, R.; Kant, E.; and Martin, N. 1985. *Programming Expert Systems in OPS5*. Boston: Addison-Wesley.

Clocksin, W. F., and Mellish, C. S. 1981. *Programming in Prolog*. Berlin: Springer-Verlag.

Coulom, R. 2006. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Proceedings of the Fifth International Conference on Computers and Games*, 72–83. Turin, Italy: Springer.

Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J.; Schlaefer, N.; and Welty, C. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine* 31: 59–79.

Fikes, R. E., and Nilsson, N. J. 1972. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2: 189–208.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20: 61–124.

Hoffmann, J. 2001. FF: The Fast-Forward Planning System. *AI Magazine* 22: 57–62.

Jones, R. M.; Laird, J. E.; Nielsen P. E.; Coulter, K.; Kenny, P.; and Koss, F. 1999. Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine* 20: 27–42.

Korf, R. E. 1994. An Undergraduate Introductory AI Course. *Improving Instruction of Introductory Artificial Intelligence: Papers from the 1994 Fall Symposium*, 5–7. New Orleans: AAAI Press.

Kowalski, R. 2011. *Computational Logic and Human Thinking: How to Be Artificially Intelligent*. New York: Cambridge University Press.

Langley, P. 1996. *Elements of Machine Learning*. San Francisco: Morgan Kaufmann.

Langley, P. 2012. Intelligent Behavior in Humans and Machines. *Advances in Cognitive Systems* 2: 3–12.

Langley, P.; Laird, J. E.; and Rogers, S. 2009. Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research* 10: 141–160.

Mateas, M., and Stern, A. 2005. Structuring Content in the Façade Interactive Drama Architecture. *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*. Marina del Rey, CA: AAAI Press.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. *PDDL—The Planning Domain Definition Language*. Technical Report CVC TR98003, Center for Computational Vision and Control, Yale University, New Haven, CT.

Nau, D.; Au, T.; Hghami, O.; Kuter, U.; Murdock, J. Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research* 20: 379–404.

Newell, A. 1973. Production Systems: Models of Control Structures. In W. G. Chase, ed., *Visual Information Processing.* New York: Academic Press.

Newell, A., and Simon, H. A. 1976. Computer Science as Empirical Enquiry: Symbols and Search. *Communications of the ACM* 19: 113–126.

Nilsson, N. J. 1994. Evolutionary Artificial Intelligence. *Improving Instruction of Introductory Artificial Intelligence: Papers from the 1994 Fall Symposium*, 19–21. New Orleans: AAAI Press.

Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 62: 107–136.

Russell, S. J., and Norvig, P. 1994. A Modern, Agent-Oriented Approach to Introductory Artifical Intelligence. *Improving Instruction of Introductory Artificial Intelligence: Papers from the 1994 Fall Symposium*, 15–18. New Orleans: AAAI Press.

Russell, S. J., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach* (3rd ed). Upper Saddle River, NJ: Prentice Hall.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529: 484–489.

Winograd, T. 1972. Understanding Natural Language. *Cognitive Psychology* 3: 1–191.