AI Planning

1. About this Course

Organization, and Content Overview

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Agenda

Foundations of Artificial Intelligence (FAI) Group

- **What?** Basic research in AI.
- **Who?**
  - Prof. Jörg Hoffmann (lead).
  - Bärbel Zitzmann/Angelika Scholl-Danopoulos (secretary)
  - Dr. Cosmina Croitoru (postdoc).
  - Dr. Álvaro Torralba (postdoc).
  - Rebecca Eiffler (Ph.D. student).
  - Daniel Gnad (Ph.D. student).
  - Maximilian Fickert (Ph.D. student).
  - Patrick Speicher (Ph.D. student).
  - Marcel Steinmetz (Ph.D. student).
- **Where?** E1 1, 3rd floor.

`http://fai.cs.uni-saarland.de/`

Who Does What in This Course: FAI Group

- Álvaro Torralba and Cosmina Croitoru: Lectures.
- Cosmina Croitoru: Tutoring the paper exercises.
- Daniel Gnad and Marcel Steinmetz: Tutoring the programming exercises.
  → Programming Workshops.
- Angelika Scholl-Danopoulos: Contact for administrational issues like HISPOS and certificates ("Schein") at the end in case of need.

Who Does What in This Course: You

. . . and who are you?

E.g.: Did you take the AI'18 Core Course? The course is self-contained, but it makes things easier if you have successfully completed an introductory course to AI (mine, in particular).

E.g.: Which course of studies (Studiengang) do you follow?

- Informatics BSc? Informatics MSc?
- Embedded Systems? Media Informatics? CyberSecurity? Visual Computing? Bioinformatics?
- Computational linguistics?
- Other? (Which?)

When and Where?

When?

- Lectures: Tuesdays 12:15–13:45, Wednesdays 16:15–17:45.
- Tutorials: Five of the slots on Wednesdays 16:15–17:45.

Where?

- Lectures: Here, i.e., Building E1 3, HS002.
  → Unless specified otherwise: see the red entries in the calendar at the bottom of course web page.
- Tutorials: Here, or in Building E1 1 room 3.06. (See Moodle!)
- Programming Office Hours: FAI offices (Building E1 1).

Web Resources

Course Web Page:

`http://fai.cs.uni-saarland.de/teaching/winter18-19/planning.html`

- General information about the course.
- Lecture slides hand-outs.
- Not updated at all, apart from the lecture slides!

Course Moodle Pages:

`https://fai-lecture.cs.uni-saarland.de/login/index.php`

- "Automatic Planning (Winter 2018/2019)".
- Registration (see also next section).
- Exercises material and solution submission.
- Announcements, discussion forum for technical questions, . . .

Your Questions

Questions about course organization/general:

- Cosmina Croitoru.

Technical questions about course content/exercises:

- "Technical Discussion" in Moodle pages. (Read by everybody: All students, the whole team.)
- Cosmina Croitoru, Álvaro Torralba.

Questions about how to use the Moodle pages:

- Cosmina Croitoru.

Other questions:

- Álvaro Torralba.
- Consultation hour (Sprechstunde): Wednesdays, 9:30–10:30.
- Come to the front directly after the lecture.

Exercises: What?

Programming exercises: Programming your own planning system!

- C++, starting from base code we will provide.
- You don't need to worry about parsing PDDL etc., your task will only be to implement search routines and heuristic functions.
- Groups of 3 allowed.
- Competition between your systems at end of course.

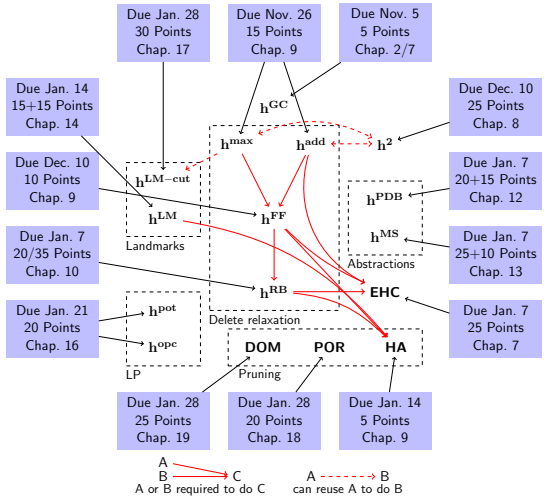
  → Competition winners get some goodies.

Paper Exercises: (same groups of 3)

- Apply concepts and algorithms to examples, lead simple proofs.
- 2-week intervals for hand-out/submission.
- Hand-out in Moodle, submission deadline stated on each sheet.
- A feature, not a bug: We won't make this too hard, and solving these will prepare you well for the exam.

Programming Exercises: What Exactly?

Whatever you want!



Exams

Exams Setup:

- Exercises similar to the paper exercises (proofs simpler).
- Open book (any paper material allowed, no electronic devices).
- Each exam is a separate attempt to pass this course; the better one of the two grades counts.

Exam Dates:

- 1st Exam: Monday February 11, 14:00–17:00, HS003.
- 2nd Exam: TBA.

Exam Preparation:

- Wednesday February 6 lecture slot.
- We'll do exercises on the blackboard; opportunity for you to ask questions.

Grading Rules

For each exam, $\in \{$**1st, 2nd**$\}$**:**

- Paper exercises 100 points maximum; 50 points needed for exam admission.
- Programming exercises maximally 110 points count; 50 points needed for exam admission.
- Exam 100 points maximum; 50 points needed to pass exam.
- Overall grade is based on the following points calculation:

$$\max(exam, 0.6 * exam + 0.4 * \min(programmingpoints, 110))$$

→ The planner programming is an essential part of this course. Doing well helps you get a good overall grade.

Registration

Moodle Registration:

- On our Moodle pages, you can login with your CS department account, i.e., the one you use for e.g. the cloud.

  → Attention: If you have no such account, contact Cosmina Croitoru.
- Go to "Automatic Planning (Winter 2017/2018)"; when prompted, enter key "UDS-PLAN-18".
- Everything (announcements, discussions, exercises) happens in Moodle, so do register soon!

Registering for the course/exams etc.:

- According to your course of study. For questions, please contact (not us but) the study administration, studium@cs.uni-saarland.de

Slides Availability

→ Like in the AI Core Course, I'm gonna be asking you lots of questions, and do many examples interactively at the blackboard. Thus we have the same 2 different forms of hand-outs.

Pre-Handouts:

- Without answers to questions.
- Without details for (many) examples.

Availability:

- 1 day before chapter begins.
- 1 slide/page.
- 4 slides/page ("-4up").

Post-Handouts:

- With answers to questions and details for examples.
- Corrections, if applicable.

Availability:

- Day the chapter ended.
- 1 slide/page.
- 4 slides/page ("-4up").

→ The chapters are short, most fitting into a single lecture slot. So you'll typically have the post-handouts almost immediately after the lecture.

Course Material

That one's easy: There is none.

More seriously:

- The content we cover is way too recent to be in a text book.
- The text books available are of very limited relevance to this course: Russel & Norvig is rudimentarily relevant at best; the two books by Ghallab, Nau & Traverso contain very little of the material we're interested in here.
- The ground truth, as far as this course is concerned, are the post-handouts.
- The slides include citations for general reference. I don't expect you to read this unless you want to embark on a BSc or MSc thesis in the subject of the respective chapter.
- In each chapter under "Reading", I list a few directly relevant papers, and give a few useful hints. Reading these papers is still entirely optional, and not needed to understand the course. But if you're really into it, then this is where to start . . .

Slides Structure

- The course consists of 21 chapters, each of which consists of several sections. (I'll be using these words in cross-references.)
- E.g. right now we're in → **Chapter 1, Slides section**.
- I might, if required in the interest of time, skip a few sections; and some sections are explicitly marked "for Reference".
- This material is *not* exam-relevant. But it might be useful for you to look through these slides anyway (I'm trying to make the candidates for this very self-contained).
- Each chapter consists of an Introduction section, several sections of content, and a Conclusion section.
- Special features:
  - Questionnaire: As in AI Core Course.
  - Reading: cf. previous slide.

Beating Kasparov is great . . .

Duell Kasparow gegen Deep Blue (1997): *Demütigende Niederlage*

Beating Kasparov is great . . . but how to play Mau-Mau??

Duell Kasparow gegen Deep Blue (1997): *Demütigende Niederlage*

- You (and your brother/sister/little nephew) are better than Deep Blue at *everything* – except playing Chess.
- Is that (artificial) "Intelligence"?
- How to build machines that automatically solve **new** problems?

General Problem Solving

Ambition: Write one program that can solve all problems.

→ Write $X \in \{algorithms\}$: for all $Y \in \{$"problems"$\}$: X "solves" Y

→ What is a "problem"? What does it mean to "solve" it?

Ambition 2.0: Write one program that can solve a large class of problems.

Ambition 3.0: Write one program that can solve a large class of problems effectively.

(some new problem)

describe problem ↦ use off-the-shelf solver

(solution competitive with a human-made specialized program)

→ Beat humans at coming up with clever solution methods!

General Game Playing

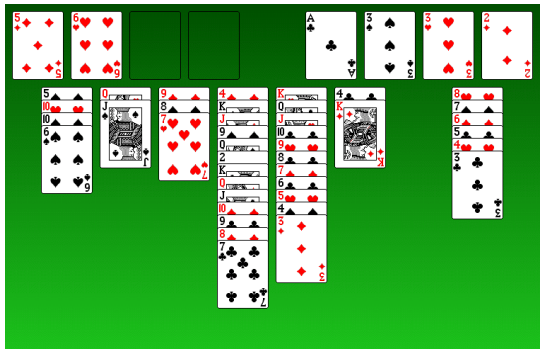
Ambition:

Write one program that can play all (2-player, 0-sum, . . .) games.

Duell Kasparow gegen Deep Blue (1997): *Demütigende Niederlage*

- Describe "game" in a declarative language (based on logics).
- Variables (board position/cards), operators (moves), win/loss.
- Chess, but also Räuberschach, and Backgammon, and . . .

Classical Search Problems



- States: Card positions (*position_Jspades=Qhearts*).
- Actions: Card moves (*move_Jspades_Qhearts_freecell4*).
- Initial state: Start configuration.
- Goal states: All cards "home".
- Solution: Card moves solving this game.

Automatic Planning

Ambition:

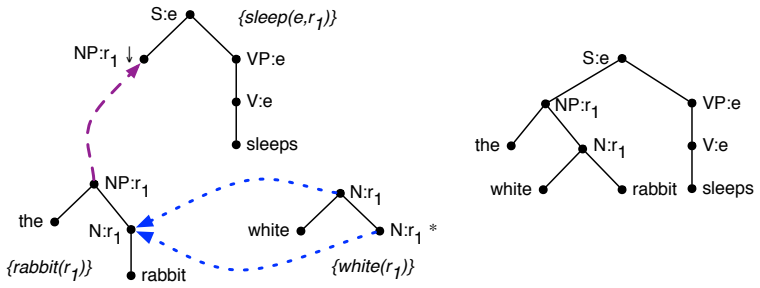
Write one program that can solve all classical search problems.

Planning (problem description) languages:

- A *logical description* of the possible states (vs. Blackbox: data structures). E.g.: predicate $Eq(.,.)$.
- A *logical description* of the initial state I (vs. data structures). E.g.: $Eq(x, 1)$.
- A *logical description* of the goal condition G (vs. a goal-test function). E.g.: $Eq(x, 2)$.
- A *logical description* of the set A of actions in terms of preconditions and effects (vs. functions returning applicable actions and successor states). E.g.: "increment x: pre $Eq(x, 1)$, eff $Eq(x, 2) \wedge \neg Eq(x, 1)$".

$\rightarrow$ Solution (plan) = sequence of actions from A, transforming I into a state that satisfies G. E.g.: "increment x".

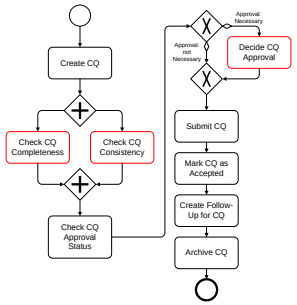
Application: Natural Language Generation



- **Input:** Tree-adjoining grammar, intended meaning.
- **Output:** Sentence expressing that meaning.

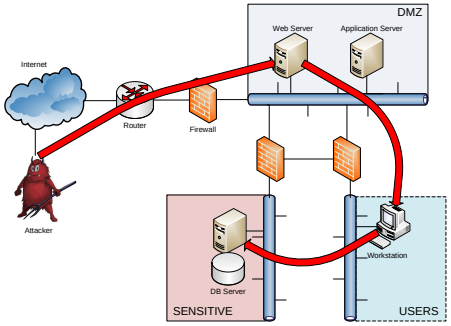
Application: Business Process Templates at SAP

Action name	precondition	effect
Check CQ Completeness	CQ.archiving:notArchived	CQ.completeness:complete OR CQ.completeness:notComplete
Check CQ Consistency	CQ.archiving:notArchived	CQ.consistency:consistent OR CQ.consistency:notConsistent
Check CQ Approval Status	CQ.archiving:notArchived AND CQ.approval:notChecked AND CQ.completeness:complete AND CQ.consistency:consistent	CQ.approval:necessary OR CQ.approval:notNecessary
Decide CQ Approval	CQ.archiving:notArchived AND CQ.approval:necessary	CQ.approval:granted OR CQ.approval:notGranted
Submit CQ	CQ.archiving:notArchived AND (CQ.approval:notNecessary OR CQ.approval:granted)	CQ.submission:submitted
Mark CQ as Accepted	CQ.archiving:notArchived AND CQ.submission:submitted	CQ.acceptance:accepted
Create Follow-Up for CQ	CQ.archiving:notArchived AND CQ.acceptance:accepted	CQ.followUp:documentCreated
Archive CQ	CQ.archiving:notArchived	CQ.archiving:archived



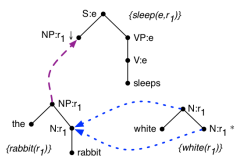
- **Input:** SAP-scale model of behavior of activities on Business Objects, process endpoint.
- **Output:** Process template leading to this point.

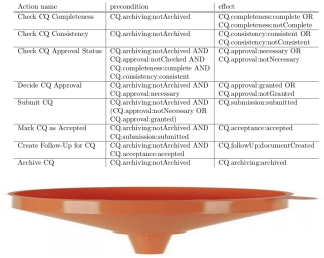
Application: Simulated Penetration Testing

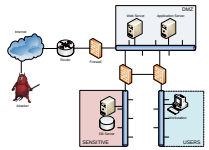


- **Input:** Network configuration, location of sensible data.
- **Output:** Sequence of exploits giving access to that data.

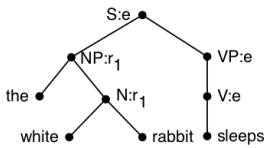
Planning!

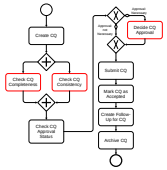


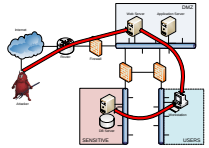




Planning Domain Definition Language (PDDL) $\mapsto$ Planning System







General Problem Solving, Pros and Cons

- **Powerful:** In some applications generality is absolutely necessary. (SAP!)
- **Quick:** Rapid prototyping: 10s lines of problem description vs. 1000s lines of C++ code. (Language generation!)
- **Flexible:** Adapt/maintain *the description*. (Network security!)
- **Intelligent:** Determines automatically how to solve a complex problem effectively! (The ultimate goal, no?!)

- **Efficiency loss:** Without any domain-specific knowledge about Chess, you don't beat Kasparov ...
  $\rightarrow$ Trade-off between "automatic and general" vs. "manual work but effective".

How to make fully automatic algorithms effective?

Motivation, p.s.

What we're going to do in this course is not "just" relevant to Artificial Intelligence!

- At the end of the day, what we're looking at is the reachability problem in compactly represented transition systems.
- We'll consider generic methods to solve that problem.
- Such problems occur in many places in CS:
  - Model Checking/Verification (state spaces of software/hardware).
  - BioInformatics (e.g. sequence alignment, genome rearrangement).
  - Diagnosis of discrete-event systems (behavior space of the system being diagnosed).
  - Game-playing algorithms (space of game positions).

$\rightarrow$ We will focus on AI Planning exclusively, but the techniques we will consider are very generic and can be applied broadly.

Course Content Overview

Chapter	Title
1.	About this Course
2.	Planning Formalisms
3.	PDDL
4.	Applications
5.	Causal Graphs
6.	Progression and Regression
7.	Search Algorithms
8.	Critical Path Heuristics
9.	Delete Relaxation Heuristics
10.	Partial Delete Relaxation
11.	Abstractions
12.	Pattern Database Heuristics
13.	Merge-and-Shrink Heuristics
14.	Landmark Heuristics
15.	Combining Heuristic Functions
16.	LP Heuristics
	Christmas Surprise
17.	Comparing Heuristic Functions
18.	Partial-Order Reduction
19.	Dominance Pruning
20.	Symmetry Reduction
21.	Planning Systems and the IPC

→ Notice something here? The word "heuristic" occurs quite a bit.

International Planning Competition (IPC)

Reminder from AI Core Course:

- **IPC 2000:** Winner heuristic search.
- **IPC 2002:** Winner heuristic search.
- **IPC 2004:** Winner satisficing heuristic search; optimal compilation to SAT.
- **IPC 2006:** Winner satisficing heuristic search; optimal compilation to SAT.
- **IPC 2008:** Winner satisficing heuristic search; optimal symbolic search.
- **IPC 2011:** Winner satisficing heuristic search; optimal heuristic search.
- **IPC 2014:** Winner satisficing heuristic search; optimal symbolic search.
- **IPC 2018:** Winner satisficing heuristic search; portfolio/symbolic search/heuristic search.

→ The course focuses exclusively on the heuristic search approach to planning.

→ That approach has revolutionized the area in the last 15 years. We cover its technical core, mostly recent research results close to the research frontier.

Course Content Break-Down and Motivation

Getting started:

Chapter	Title
1.	About this Course
2.	Planning Formalisms
3.	PDDL
4.	Applications

- I won't give an introduction other than this one, as I already introduced planning in the AI Core Course (if you weren't there, look here).
- Chapters 2 and 3 are to get you up to speed with the problems addressed in automatic planning, more specifically in this course.

  → Anybody remember what PDDL is? The de-facto standard input language for planning systems today. (Used in the planning competitions, in particular.)
- Chapter 4 describes some recent and successful applications of planning, in the hope that this motivates you to hear the rest of the course out :-)

Course Content Break-Down and Motivation, ctd.

Our basic tool set:

Chapter	Title
5.	Causal Graphs
6.	Progression and Regression
7.	Search Algorithms

- Causal graphs are a very useful notion to capture certain structural properties of planning problems. They'll be used in various later chapters.
- Progression and regression are the most basic approaches to organizing the search for a solution.
- Search algorithms (like A^*) define how we navigate the search space. This is partly a reminder, and partly an extension, of the material I covered in the AI Core Course.

Course Content Break-Down and Motivation, ctd.

The "meat" of the course:

Chapter	Title
8.	Critical Path Heuristics
9.	Delete Relaxation Heuristics
10.	Partial Delete Relaxation
11.	Abstractions
12.	Pattern Database Heuristics
13.	Merge-and-Shrink Heuristics
14.	Landmark Heuristics
16.	LP Heuristics

- Essentially four families of methods for automatically generating heuristic functions: critical paths, delete relaxation, abstractions, landmarks. (Pattern databases and merge-and-shrink heuristics are abstractions.)
- In the AI Core Course, we only did delete relaxation heuristics. Chapter 9 presents them from a more general perspective, Chapter 10 generalizes them substantially.
- LP Heuristics: Heuristics based on Linear Programming. New chapter this year.

Course Content Break-Down and Motivation, ctd.

What to do with all these heuristics??

Chapter	Title
15.	Combining Heuristic Functions
17.	Comparing Heuristic Functions

- These two chapters are concerned with research issues that arise from the presence of an entire arsenal of heuristics.
- Chapter 15 details the most powerful technique known for combining heuristic functions, and overviews the currently known methods to implement that technique.
- Chapter 17 summarizes ground-breaking results regarding the theoretical comparison of heuristic functions, based on identifying whether or not they can simulate each other efficiently.

Course Content Break-Down and Motivation, ctd.

Admissible pruning methods:

Chapter	Title
18.	Partial-Order Reduction
19.	Dominance Pruning
20.	Symmetry Reduction

- There is life beyond heuristic search ...
  - 3 families of methods for reducing the search space without losing completeness/optimality.
  - Chapter 18: Determine subset of actions ("strong stubborn set") branching over which suffices to preserve optimality.
  - Chapter 19: Find a dominance relation over states, and prune t if we have already seen s which dominates t.
  - Chapter 20: Find symmetry relation over states, and prune t if we have already seen s symmetric to t.

Course Content Break-Down and Motivation, ctd.

Wrapping it up:

Chapter	Title
21.	Planning Systems and the IPC

- Chapter 21 concludes the course with a detailed summary of the development of implemented planning systems during the last decade, and the results of the international planning competitions.

So: Welcome!

That's it! Enjoy the course!