Chapter: *Introduction to (Non-Hierarchical) Planning*

Dr. Pascal Bercher

Institute of Artificial Intelligence,
Ulm University, Germany

Winter Term 2018/2019

(Compiled on: February 20, 2019)

ulm university universität uulm

Overview:

Where and When?

Lecture	Tuesday	14:15 - 15:45 Uhr	(O27/R 2202)
	Thursday	16:15 - 17:45 Uhr	(O27/R 2202)

Exercises Replace every 4th lecture, see Moodle

Consultation Hours Right after the lectures and by appointment
`pascal.bercher@uni-ulm.de`

Exam

Grades

- Exam gives 6 ECTS if passed.
- Scoring at least 50% of the exercise points improves the exam grade by one step (0.3 or 0.4, respectively).

Exam

Grades

- Exam gives 6 ECTS if passed.
- Scoring at least 50% of the exercise points improves the exam grade by one step (0.3 or 0.4, respectively).

Dates

- The exam may be written or oral, depending on the number of participants.
- Exams: We agree on this *now* or *very soon*.

Literature

Remarks

- We provide pointers to scientific papers for most of the content.
They provide more details, but are *not* required.

Literature

Remarks

- We provide pointers to scientific papers for most of the content. **They provide more details, but are *not* required.**
- Most of the lecture's content is *state of the art*! It is way too recent to be in any textbook.

Literature

Remarks

- We provide pointers to scientific papers for most of the content. **They provide more details, but are *not* required.**
- Most of the lecture's content is *state of the art*! It is way too recent to be in any textbook.
- The suggested text books and lectures mainly provide detailed explanations as well as examples about the basics.

Literature, cont'd I

Text Books

- S. Russell, P. Norvig: *Artificial Intelligence – A Modern Approach*, Prentice Hall, 2010. (Much on non-hierarchical planning, but only approx. 10/1300 pages about hierarchical planning.)

Literature, cont'd I

Text Books

- S. Russell, P. Norvig: *Artificial Intelligence – A Modern Approach*, Prentice Hall, 2010. (Much on non-hierarchical planning, but only approx. 10/1300 pages about hierarchical planning.)
- Q. Yang: *Intelligent Planning – A Decomposition and Abstraction Based Approach*, Springer, 1997. (The entire book is about hierarchical planning.)

Literature, cont'd I

Text Books

- S. Russell, P. Norvig: *Artificial Intelligence – A Modern Approach*, Prentice Hall, 2010. (Much on non-hierarchical planning, but only approx. 10/1300 pages about hierarchical planning.)
- Q. Yang: *Intelligent Planning – A Decomposition and Abstraction Based Approach*, Springer, 1997. (The entire book is about hierarchical planning.)
- M. Ghallab, D. Nau, P. Traverso: *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004. (1+/24 chapters are on hierarchical planning.)

Literature, cont'd I

Text Books

- S. Russell, P. Norvig: *Artificial Intelligence – A Modern Approach*, Prentice Hall, 2010. (Much on non-hierarchical planning, but only approx. 10/1300 pages about hierarchical planning.)
- Q. Yang: *Intelligent Planning – A Decomposition and Abstraction Based Approach*, Springer, 1997. (The entire book is about hierarchical planning.)
- M. Ghallab, D. Nau, P. Traverso: *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004. (1+/24 chapters are on hierarchical planning.)
- M. Ghallab, D. Nau, P. Traverso: *Automated Planning and Acting*, Cambridge University Press, 2016. (Only a very few pages are on hierarchical planning.)

Literature, cont'd II

Lectures

- Automated Planning, Susanne Biundo, Ulm University. Slides available upon request. (1/6 chapters is about hierarchical planning.)

Literature, cont'd II

Lectures

- Automated Planning, Susanne Biundo, Ulm University. Slides available upon request. (1/6 chapters is about hierarchical planning.)
- Planning and Optimization, Malte Helmert, University of Basel. Slides publicly available. (Covers only non-hierarchical planning.)

Literature, cont'd II

Lectures

- Automated Planning, Susanne Biundo, Ulm University. Slides available upon request. (1/6 chapters is about hierarchical planning.)
- Planning and Optimization, Malte Helmert, University of Basel. Slides publicly available. (Covers only non-hierarchical planning.)
- Automatic Planning, Jörg Hoffmann, Saarland University. Slides publicly available. (Covers only non-hierarchical planning.)

Literature, cont'd II

Lectures

- Automated Planning, Susanne Biundo, Ulm University. Slides available upon request. (1/6 chapters is about hierarchical planning.)
- Planning and Optimization, Malte Helmert, University of Basel. Slides publicly available. (Covers only non-hierarchical planning.)
- Automatic Planning, Jörg Hoffmann, Saarland University. Slides publicly available. (Covers only non-hierarchical planning.)
- Automated Planning: Theory and Practice, Dana Nau, University of Maryland. Slides publicly available. (1/16 chapters covers hierarchical planning.)

Literature, cont'd III

Planning Research in the Scientific Landscape

Most important planning and AI conferences:

- ICAPS, the Int. Conf. on Planning and Scheduling.
  ICAPS is the fusion of two conferences in 2003. Before:
  - ECP, the Europ. Conf. on Planning (odd years).
  - AIPS, the Int. Conf. on AI Planning Systems (even years).

Literature, cont'd III

Planning Research in the Scientific Landscape

Most important planning and AI conferences:

- ICAPS, the Int. Conf. on Planning and Scheduling.
  ICAPS is the fusion of two conferences in 2003. Before:
  - ECP, the Europ. Conf. on Planning (odd years).
  - AIPS, the Int. Conf. on AI Planning Systems (even years).
- IJCAI, the Int. Joint Conf. on Artificial Intelligence.

Literature, cont'd III

Planning Research in the Scientific Landscape

Most important planning and AI conferences:

- ICAPS, the Int. Conf. on Planning and Scheduling.
  ICAPS is the fusion of two conferences in 2003. Before:
  - ECP, the Europ. Conf. on Planning (odd years).
  - AIPS, the Int. Conf. on AI Planning Systems (even years).
- IJCAI, the Int. Joint Conf. on Artificial Intelligence.
- ECAI, the Europ. Conf. on Artificial Intelligence.

Literature, cont'd III

Planning Research in the Scientific Landscape

Most important planning and AI conferences:

- ICAPS, the Int. Conf. on Planning and Scheduling.
  ICAPS is the fusion of two conferences in 2003. Before:
  - ECP, the Europ. Conf. on Planning (odd years).
  - AIPS, the Int. Conf. on AI Planning Systems (even years).
- IJCAI, the Int. Joint Conf. on Artificial Intelligence.
- ECAI, the Europ. Conf. on Artificial Intelligence.
- AAAI, the AAAI Conf. on Artificial Intelligence.
  (AAAI = Association for the Advancement of Artificial Intelligence.)

Literature, cont'd III

Planning Research in the Scientific Landscape

Most important planning and AI conferences:

- ICAPS, the Int. Conf. on Planning and Scheduling.
  ICAPS is the fusion of two conferences in 2003. Before:
  - ECP, the Europ. Conf. on Planning (odd years).
  - AIPS, the Int. Conf. on AI Planning Systems (even years).
- IJCAI, the Int. Joint Conf. on Artificial Intelligence.
- ECAI, the Europ. Conf. on Artificial Intelligence.
- AAAI, the AAAI Conf. on Artificial Intelligence.
  (AAAI = Association for the Advancement of Artificial Intelligence.)

Most important AI journals:

- Artificial Intelligence (AIJ, AI Journal).
- Journal of Artificial Intelligence Research (JAIR).

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability
  - uncertainty: probabilistic, non-deterministic, partially observable, etc.

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability
  - uncertainty: probabilistic, non-deterministic, partially observable, etc.
  - learning

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability
  - uncertainty: probabilistic, non-deterministic, partially observable, etc.
  - learning
- Logistics Robots.

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability
  - uncertainty: probabilistic, non-deterministic, partially observable, etc.
  - learning
- Logistics Robots.
- Distributed and Multi-Agent Planners.

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability
  - uncertainty: probabilistic, non-deterministic, partially observable, etc.
  - learning
- Logistics Robots.
- Distributed and Multi-Agent Planners.
- ICKEPS, the International Competition on Knowledge Engineering for Planning and Scheduling.

Literature, cont'd IV

Planning(-related) Competitions

- IPC, the International Planning Competition. Tracks:
  - satisficing vs. optimal
  - temporal
  - unsolvability
  - uncertainty: probabilistic, non-deterministic, partially observable, etc.
  - learning
- Logistics Robots.
- Distributed and Multi-Agent Planners.
- ICKEPS, the International Competition on Knowledge Engineering for Planning and Scheduling.
- For a comprehensive list, see: `http://www.icaps-conference.org/index.php/Main/Competitions`

Lecture Slides

- Access them via Moodle using the password **HP-2018!**

Lecture Slides

- Access them via Moodle using the password **HP-2018!**
- In case you find any flaws, problems, inconsistencies, or simply have ideas of how to improve the slides: *Let me know!*

Lecture Slides

- Access them via Moodle using the password **HP-2018!**
- In case you find any flaws, problems, inconsistencies, or simply have ideas of how to improve the slides: *Let me know!*
- The lecture slides will always be online *after* each lecture.

Lecture Slides

- Access them via Moodle using the password **HP-2018!**
- In case you find any flaws, problems, inconsistencies, or simply have ideas of how to improve the slides: *Let me know!*
- The lecture slides will always be online *after* each lecture.
  - Primarily, this is done to encourage active participation. Often, slides contain questions, the solutions of which are one the subsequent slides.

Lecture Slides

- Access them via Moodle using the password **HP-2018!**
- In case you find any flaws, problems, inconsistencies, or simply have ideas of how to improve the slides: *Let me know!*
- The lecture slides will always be online *after* each lecture.
  - Primarily, this is done to encourage active participation. Often, slides contain questions, the solutions of which are one the subsequent slides.
  - Secondly, this allows to fix issues that were detected during the lecture *before* they are put online.

Lecture Slides

- Access them via Moodle using the password **HP-2018!**
- In case you find any flaws, problems, inconsistencies, or simply have ideas of how to improve the slides: *Let me know!*
- The lecture slides will always be online *after* each lecture.
  - Primarily, this is done to encourage active participation. Often, slides contain questions, the solutions of which are one the subsequent slides.
  - Secondly, this allows to fix issues that were detected during the lecture *before* they are put online.
- Even after putting the slides online the first time, they might still get updated. Ensure to use the newest version when studying for the exam!

Informal Description

Patrik Haslum

Planning is the art and practice of thinking before acting.

Informal Description

Patrik Haslum

Planning is the art and practice of thinking before acting.

Jörg Hoffmann

Selecting a goal-leading course of action based on a high-level description of the world.

Informal Description

Patrik Haslum

Planning is the art and practice of thinking before acting.

Jörg Hoffmann

Selecting a goal-leading course of action based on a high-level description of the world.

Just a bit more formally...

Planning is the reasoning process required to generate a *plan* – a sequence of action that transforms a given state of a system into a desired one.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.
- Automated support:

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.
- Automated support:
  - Model can be checked for consistency (modeling support).

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.
- Automated support:
  - Model can be checked for consistency (modeling support).
  - Existing techniques for proving unsolvability can be used.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.
- Automated support:
  - Model can be checked for consistency (modeling support).
  - Existing techniques for proving unsolvability can be used.
  - Plan explanation techniques can be exploited.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.
- Automated support:
  - Model can be checked for consistency (modeling support).
  - Existing techniques for proving unsolvability can be used.
  - Plan explanation techniques can be exploited.
  - The planning systems often exist for a long time, i.e. they might be better tested for bugs than new software.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Advantages

- Cost-effective: only write a formal model, but no specialized solvers.
- Optimality guarantees.
- Automated support:
  - Model can be checked for consistency (modeling support).
  - Existing techniques for proving unsolvability can be used.
  - Plan explanation techniques can be exploited.
  - The planning systems often exist for a long time, i.e. they might be better tested for bugs than new software.
  - Verification tools exist that test solutions for their actual correctness (redundant if the planning software is definitely bug-free).

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Disadvantages

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Disadvantages

- You need a planning expert to model the domain.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Disadvantages

- You need a planning expert to model the domain.
- Potential inefficiency: a domain-specific solver is most likely more efficient than a domain-independent one.

Domain-Independent Approach

- We want to define a large *class of problems*, i.e., a range of different problems with certain properties.
- For these problems there are general problem solvers that can solve *all* possible problems of the respective class.

Disadvantages

- You need a planning expert to model the domain.
- Potential inefficiency: a domain-specific solver is most likely more efficient than a domain-independent one.

  Though differences must not always be large as shown in one real-world application: Malte Helmert and Hauke Lasinger. "The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem". In: *Proc. of the 20th Int. Conf. on Automated Planning and Scheduling (ICAPS 2010)*. AAAI Press, 2010, pp. 234–237

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:

  Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).
- Actions can take time: their effects occur at one or more time points (static) or in a continuous way.

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).
- Actions can take time: their effects occur at one or more time points (static) or in a continuous way.
- Actions might consume or produce resources.

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).
- Actions can take time: their effects occur at one or more time points (static) or in a continuous way.
- Actions might consume or produce resources.
- Actions can be deterministic, non-deterministic, or probabilistic.

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).
- Actions can take time: their effects occur at one or more time points (static) or in a continuous way.
- Actions might consume or produce resources.
- Actions can be deterministic, non-deterministic, or probabilistic.
- The environment might be fully or partially observable; epistemic knowledge might be modeled.

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).
- Actions can take time: their effects occur at one or more time points (static) or in a continuous way.
- Actions might consume or produce resources.
- Actions can be deterministic, non-deterministic, or probabilistic.
- The environment might be fully or partially observable; epistemic knowledge might be modeled.
- There can be one agent or several (see also multi-agent-planning and general game playing).

Planning in the Research Landscape

Properties of Planning Tasks

- Goals to achieve versus tasks to accomplish:
  - Ex.1 "Package 1 and 2 should be at locations 3 and 4" versus "Deliver package 1 and 2 to locations 3 and 4"
  - Ex.2 "Be at location X (starting from location X)" versus "Do a roundtrip from X to X"
- Goals might "hard" (obligatory) or "soft" (optional).
- Actions can take time: their effects occur at one or more time points (static) or in a continuous way.
- Actions might consume or produce resources.
- Actions can be deterministic, non-deterministic, or probabilistic.
- The environment might be fully or partially observable; epistemic knowledge might be modeled.
- There can be one agent or several (see also multi-agent-planning and general game playing).
- Many more...

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:
  - Detect (i.e., prove) the unsolvability.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:
  - Detect (i.e., prove) the unsolvability.
  - Provide certificates that serve as verifiable unsolvability proofs.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:
  - Detect (i.e., prove) the unsolvability.
  - Provide certificates that serve as verifiable unsolvability proofs.
- Heuristic search.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:
  - Detect (i.e., prove) the unsolvability.
  - Provide certificates that serve as verifiable unsolvability proofs.
- Heuristic search.
- Complexity analysis.

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:
  - Detect (i.e., prove) the unsolvability.
  - Provide certificates that serve as verifiable unsolvability proofs.
- Heuristic search.
- Complexity analysis.
  - How hard are the problems?

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms.
  - Reductions to other problems.
  - Heuristics.
  - Divide problem into independent sub problems.
  - Mixed-initiative plan generation with a human.
- Unsolvable planning tasks:
  - Detect (i.e., prove) the unsolvability.
  - Provide certificates that serve as verifiable unsolvability proofs.
- Heuristic search.
- Complexity analysis.
  - How hard are the problems?
  - What can be expressed by our formalisms?

Planning in the Research Landscape, cont'd I

What is being researched?

- Plan generation:
  - Algorithms. ***(covered in lecture)***
  - Reductions to other problems. ***(covered in lecture)***
  - Heuristics. ***(covered in lecture)***
  - Divide problem into independent sub problems. *(not covered)*
  - Mixed-initiative plan generation with a human. *(not covered)*
- Unsolvable planning tasks: *(not covered)*
  - Detect (i.e., prove) the unsolvability.
  - Provide certificates that serve as verifiable unsolvability proofs.
- Heuristic search. ***(covered in lecture)***
- Complexity analysis. ***(covered in lecture)***
  - How hard are the problems?
  - What can be expressed by our formalisms?

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.
- Plan and goal recognition.

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.
- Plan and goal recognition.
- Plan Repair.

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.
- Plan and goal recognition.
- Plan Repair.
- Application to real-world settings:

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.
- Plan and goal recognition.
- Plan Repair.
- Application to real-world settings:
  - Plan execution and monitoring.

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.
- Plan and goal recognition.
- Plan Repair.
- Application to real-world settings:
  - Plan execution and monitoring.
  - Convey plans to human users.

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support.
- Explainable planning.
- Plan and goal recognition.
- Plan Repair.
- Application to real-world settings:
  - Plan execution and monitoring.
  - Convey plans to human users.
  - *many* more ...

Planning in the Research Landscape, cont'd I

What is being researched?

- Modeling support. *(not covered)*
- Explainable planning. ***(covered if time)***
- Plan and goal recognition. *(not covered)*
- Plan Repair. ***(covered if time)***
- Application to real-world settings:
  - Plan execution and monitoring. ***(covered if time)***
  - Convey plans to human users. ***(covered if time)***
  - *many* more ...

Planning in the Research Landscape, cont'd II

Properties of Planning Systems

- Planning is done offline versus online (versus mixed-initiatively).

Planning in the Research Landscape, cont'd II

Properties of Planning Systems

- Planning is done offline versus online (versus mixed-initiatively).
- Solving via search: progression, regression, Partial-Order Causal-Link (POCL); ground versus lifted? or via reduction.

Planning in the Research Landscape, cont'd II

Properties of Planning Systems

- Planning is done offline versus online (versus mixed-initiatively).
- Solving via search: progression, regression, Partial-Order Causal-Link (POCL); ground versus lifted? or via reduction.
- Find some solution quickly versus the optimal or a good one. Possible quality metrics:

Planning in the Research Landscape, cont'd II

Properties of Planning Systems

- Planning is done offline versus online (versus mixed-initiatively).
- Solving via search: progression, regression, Partial-Order Causal-Link (POCL); ground versus lifted? or via reduction.
- Find some solution quickly versus the optimal or a good one. Possible quality metrics:
  - Minimize number of action.

Planning in the Research Landscape, cont'd II

Properties of Planning Systems

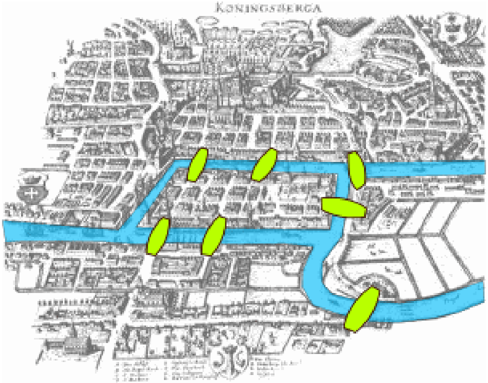
- Planning is done offline versus online (versus mixed-initiatively).
- Solving via search: progression, regression, Partial-Order Causal-Link (POCL); ground versus lifted? or via reduction.
- Find some solution quickly versus the optimal or a good one. Possible quality metrics:
  - Minimize number of action.
  - Minimize action costs.

Planning in the Research Landscape, cont'd II

Properties of Planning Systems

- Planning is done offline versus online (versus mixed-initiatively).
- Solving via search: progression, regression, Partial-Order Causal-Link (POCL); ground versus lifted? or via reduction.
- Find some solution quickly versus the optimal or a good one. Possible quality metrics:
  - Minimize number of action.
  - Minimize action costs.
  - Minimize makespan (time to execute when accounting for parallelism).

Combinatorial Problems, e.g., Seven Bridges of Königsberg



Title: *The Problem of the Seven Bridges of Königsberg*

Source: https://commons.wikimedia.org/wiki/File:Konigsberg_bridges.png

Riddles, e.g. Dining Philosophers

There are five philosophers, five meals, and five forks.

Picture:

Title: *An illustration of the dining philosophers problem*

Source: `https://commons.wikimedia.org/wiki/File:An_illustration_of_the_dining_philosophers_problem.png`

Riddles, e.g. Dining Philosophers

There are five philosophers, five meals, and five forks.

- Each philosopher is either *thinking* or *eating*, initially they are all thinking.

Picture:

Title: *An illustration of the dining philosophers problem*

Source: `https://commons.wikimedia.org/wiki/File:An_illustration_of_the_dining_philosophers_problem.png`

Riddles, e.g. Dining Philosophers

There are five philosophers, five meals, and five forks.

- Each philosopher is either *thinking* or *eating*, initially they are all thinking.
- To eat, they need their left *and* right fork,

Picture:

Title: *An illustration of the dining philosophers problem*

Source: https://commons.wikimedia.org/wiki/File:An_illustration_of_the_dining_philosophers_problem.png

Riddles, e.g. Dining Philosophers

There are five philosophers, five meals, and five forks.

- Each philosopher is either *thinking* or *eating*, initially they are all thinking.
- To eat, they need their left *and* right fork,
- They can only take back forks after they have eaten (and they will not take a fork if they have eaten already)

Picture:

Title: *An illustration of the dining philosophers problem*

Source: `https://commons.wikimedia.org/wiki/File:An_illustration_of_the_dining_philosophers_problem.png`

Riddles, e.g. Dining Philosophers

There are five philosophers, five meals, and five forks.

- Each philosopher is either *thinking* or *eating*, initially they are all thinking.
- To eat, they need their left *and* right fork,
- They can only take back forks after they have eaten (and they will not take a fork if they have eaten already)
- They all want to have eaten

Picture:

Title: *An illustration of the dining philosophers problem*

Source: https://commons.wikimedia.org/wiki/File:An_illustration_of_the_dining_philosophers_problem.png

Games, e.g., Solitaire

Games, e.g., Sliding Tile Puzzle, 15 Puzzle, n^2-1 Puzzle

2	1	4	8
9	7	11	10
6	5	15	3
13	14	12	

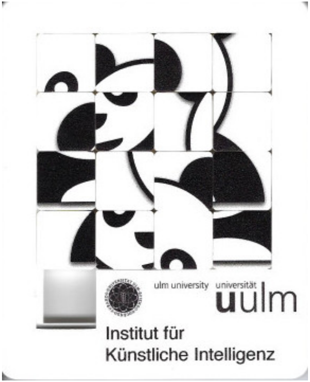
Problem

→

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Solution

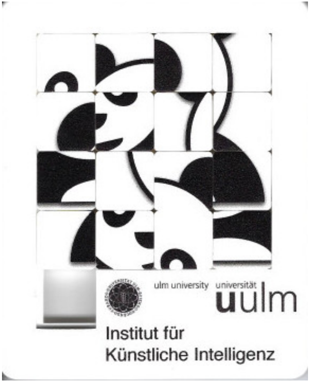
Games, e.g., Sliding Tile Puzzle, 15 Puzzle, n^2-1 Puzzle



$\longrightarrow$

Problem

Games, e.g., Sliding Tile Puzzle, 15 Puzzle, n^2-1 Puzzle

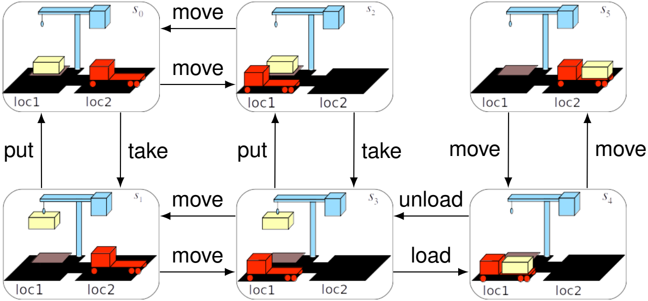


Problem

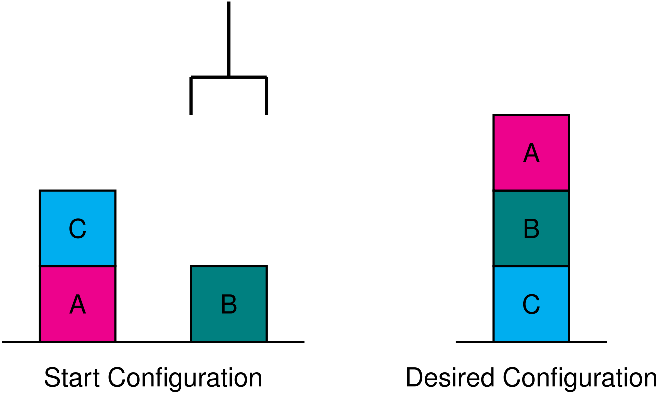
$\longrightarrow$

Solution

Cranes in a Harbor

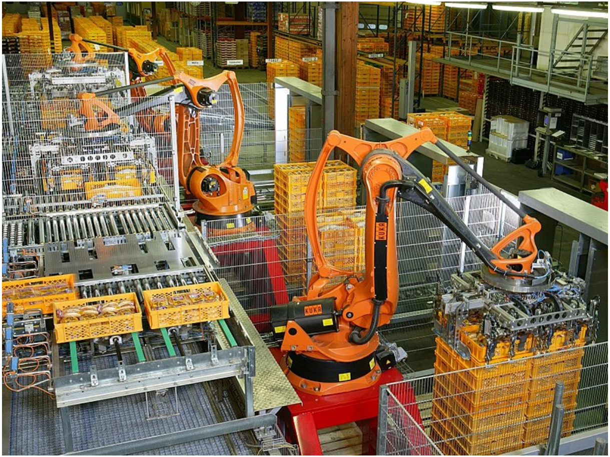


Blocksworld



Standard Planning Benchmark in the IPC and *every* planning lecture.

Automatic Factories, Industry 4.0



Source: https://en.wikipedia.org/wiki/File:Factory_Automation_Robotics_Palettizing_Bread.jpg

Greenhouse

Source: `https://www.lemnatec.com/`

Further reading:

- Malte Helmert and Hauke Lasinger. "The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem". In: *Proc. of the 20th Int. Conf. on Automated Planning and Scheduling (ICAPS 2010)*. AAAI Press, 2010, pp. 234–237
- The IPC Scanalizer Domain in PDDL (see paper above).

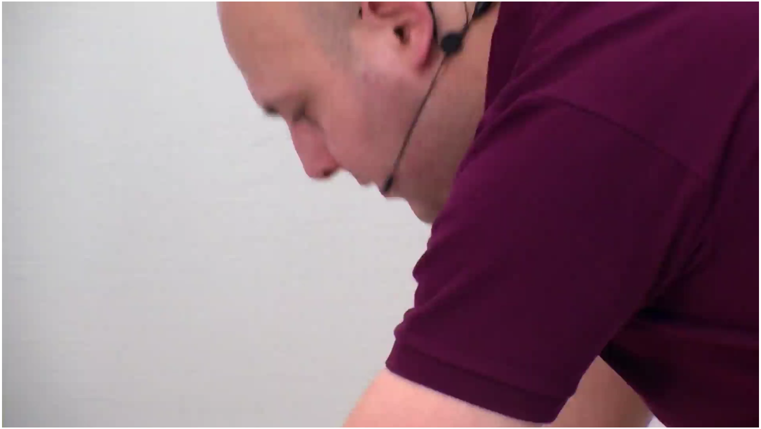
Home Theater Assembly Assistant

Four devices:

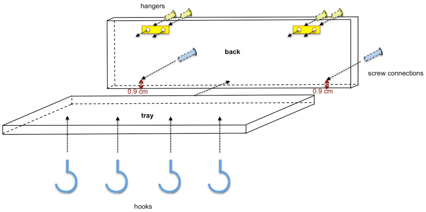
- Television (requires video)
- Blu-ray player
- Satellite receiver
- audio/video receiver (requires audio)

Home Theater Assembly Assistant, cont'd I (Step-by-Step Instructions)

Home Theater Assembly Assistant, cont'd II (Explanations)



Do-It-Yourself (DIY) Assistant

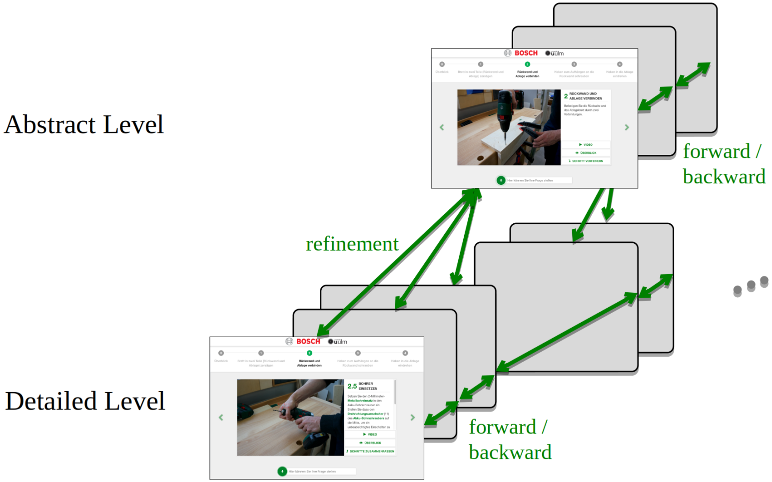


The material:

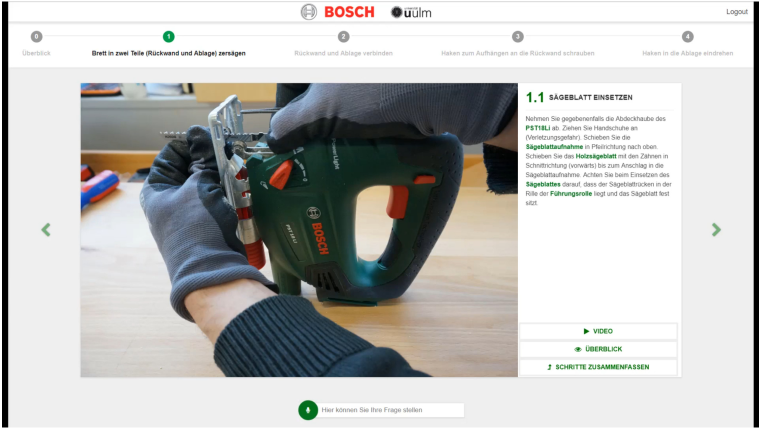
- Boards (need to be cut first)
- Electrical devices like drills and saws
- Attachments like drill bits and materials like nails

Do-It-Yourself (DIY) Assistant, cont'd I (Different Levels of Abstraction)

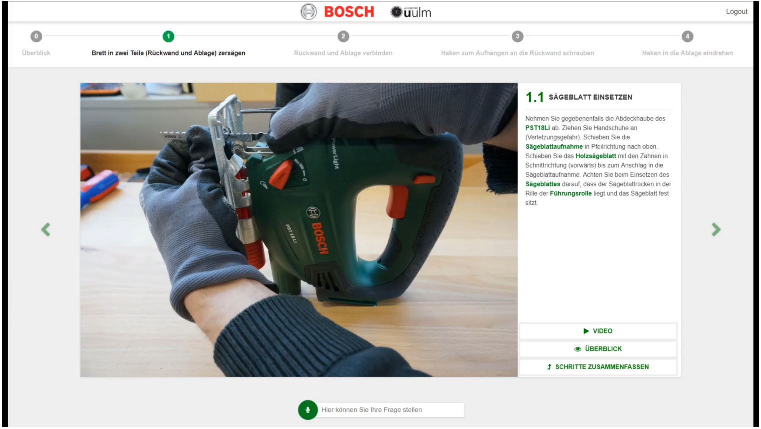
Presentation of instructions on different levels of abstraction:



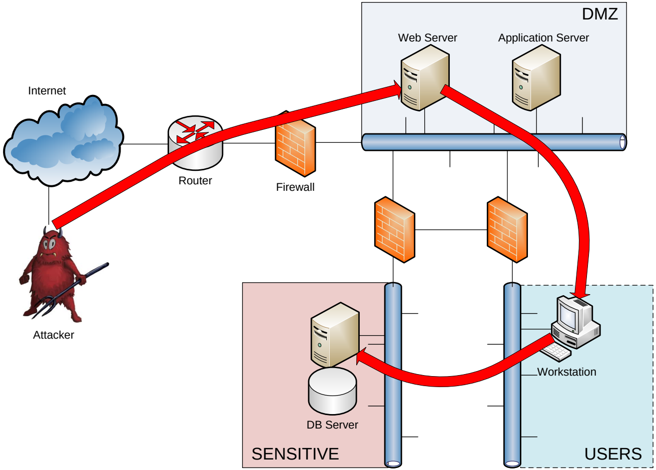
Do-It-Yourself (DIY) Assistant, cont'd II (Step-by-Step Instructions)



Do-It-Yourself (DIY) Assistant, cont'd III (Explanations)



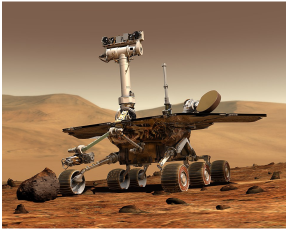
Network Intrusion



Source: https://fai.cs.uni-saarland.de/teaching/winter17-18/planning-material/planning01-about-this-course-post-handout.pdf

Mars Rover





Source:
- left https://commons.wikimedia.org/wiki/File:KSC-03PD-0786.jpg
- middle https://commons.wikimedia.org/wiki/File:Curiosity_Self-Portrait_at_%27Big_Sky%27_Drilling_Site.jpg
- right https://commons.wikimedia.org/wiki/File:NASA_Mars_Rover.jpg

Further reading:
- Pascal Bercher and Daniel Höller. "Interview with David E. Smith". In: *Künstliche Intelligenz* 30.1 (2016). Special Issue on Companion Technologies, pp. 101–105. DOI: 10.1007/s13218-015-0403-y
- Every paper about *MAPGEN* (for references, see also article above).

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation
- Manufacturing

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation
- Manufacturing
- Scheduling

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— http://sig-aps.org/

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation
- Manufacturing
- Scheduling
- Robotics & Motion Planning

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."
— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation
- Manufacturing
- Scheduling
- Robotics & Motion Planning
- E-Learning

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation
- Manufacturing
- Scheduling
- Robotics & Motion Planning
- E-Learning
- (Web) Service Composition

Not Enough?

SiGAPS

"The Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS) aims to widen awareness of AI P&S technology, promote its application outside academia, and provide resources for researchers interested in tackling application problems. This web site is our first initiative."

— `http://sig-aps.org/`

For example, see link *Success stories* for lists of deployed planning & scheduling applications within the following areas:

- Space Application
- Logistics/Transportation
- Manufacturing
- Scheduling
- Robotics & Motion Planning
- E-Learning
- (Web) Service Composition
- and more!

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.
- Fully observable.

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.
- Fully observable.
- Single-agent.

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.
- Fully observable.
- Single-agent.

More formally, a classical planning problem consists of:

- A finite set of (deterministic and discrete) actions.

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.
- Fully observable.
- Single-agent.

More formally, a classical planning problem consists of:

- A finite set of (deterministic and discrete) actions.
- A (fully known) initial state.

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.
- Fully observable.
- Single-agent.

More formally, a classical planning problem consists of:

- A finite set of (deterministic and discrete) actions.
- A (fully known) initial state.
- A set of (fully known) goal states.

What is Classical Planning?

Classical planning is the “base case”:

- Discrete (no time).
- Deterministic.
- Fully observable.
- Single-agent.

More formally, a classical planning problem consists of:

- A finite set of (deterministic and discrete) actions.
- A (fully known) initial state.
- A set of (fully known) goal states.

A *solution* (or *plan*) is any sequence of actions transforming the initial state into a goal state.

Historical Remarks

The best-known classical planning formalism is the *STRIPS formalism*.

Historical Remarks

The best-known classical planning formalism is the *STRIPS formalism*.

STRIPS = Stanford Research Institute Problem Solver

Historical Remarks

The best-known classical planning formalism is the *STRIPS formalism*.

STRIPS = Stanford Research Institute Problem Solver

That is, back in 1971, STRIPS was a planning *system*.

- Richard E. Fikes and Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". In: *Artificial Intelligence* 2 (1971), pp. 189–208.
- It was used to control the robot Shakey: `https://www.youtube.com/watch?v=qXdn6ynwpiI`

Historical Remarks

The best-known classical planning formalism is the *STRIPS formalism*.

STRIPS = Stanford Research Institute Problem Solver

That is, back in 1971, STRIPS was a planning *system*.

- Richard E. Fikes and Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". In: *Artificial Intelligence* 2 (1971), pp. 189–208.
- It was used to control the robot Shakey: `https://www.youtube.com/watch?v=qXdn6ynwpiI`

Today, STRIPS ordinarily refers to (a restricted fragment of) STRIPS' description language.

The STRIPS Formalism

The STRIPS formalism bases upon a first-order, function- and quantifier-free predicate logic.

The STRIPS Formalism

The STRIPS formalism bases upon a first-order, function- and quantifier-free predicate logic.

For the moment (we come back to this later), we furthermore assume:

The STRIPS Formalism

The STRIPS formalism bases upon a first-order, function- and quantifier-free predicate logic.

For the moment (we come back to this later), we furthermore assume:

- The model is *ground* (in contrast to *lifted*), i.e., there are no free variables: All variables (i.e., action parameters) are replaced by constants.

The STRIPS Formalism

The STRIPS formalism bases upon a first-order, function- and quantifier-free predicate logic.

For the moment (we come back to this later), we furthermore assume:

- The model is *ground* (in contrast to *lifted*), i.e., there are no free variables: All variables (i.e., action parameters) are replaced by constants.
- We have only *positive* preconditions, i.e., actions do not pose constraints on what properties do *not* hold in the state of their application.

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).
  - *States* are collections of state variables.

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).
  - *States* are collections of state variables.
  - We assume the *closed world assumption*, i.e., all variables not mentioned in a state s do not hold in that state (in contrast to: it's not known whether they hold or not).

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).
  - *States* are collections of state variables.
  - We assume the *closed world assumption*, i.e., all variables not mentioned in a state s do not hold in that state (in contrast to: it's not known whether they hold or not).
  - $S = 2^V$ is called the *state space*.

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).
  - *States* are collections of state variables.
  - We assume the *closed world assumption*, i.e., all variables not mentioned in a state s do not hold in that state (in contrast to: it's not known whether they hold or not).
  - $S = 2^V$ is called the *state space*.
- A is a finite set of actions. Each action $a \in A$ is a tuple $(pre, add, del, c) \in 2^V \times 2^V \times 2^V \times \mathbb{R}_0^+$ consisting of a *precondition*, *add and delete list*, and action costs. (We often only give a 3-tuple if there are no action costs.)

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).
  - *States* are collections of state variables.
  - We assume the *closed world assumption*, i.e., all variables not mentioned in a state s do not hold in that state (in contrast to: it's not known whether they hold or not).
  - $S = 2^V$ is called the *state space*.
- A is a finite set of actions. Each action $a \in A$ is a tuple $(pre, add, del, c) \in 2^V \times 2^V \times 2^V \times \mathbb{R}_0^+$ consisting of a *precondition*, *add and delete list*, and action costs. (We often only give a 3-tuple if there are no action costs.)
- $s_I \in S$ is the initial state (complete state description).

The STRIPS Formalism, cont'd I

A STRIPS planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ consists of:

- V is a finite set of *state variables* (also called: *facts* or *propositions*).
  - *States* are collections of state variables.
  - We assume the *closed world assumption*, i.e., all variables not mentioned in a state s do not hold in that state (in contrast to: it's not known whether they hold or not).
  - $S = 2^V$ is called the *state space*.
- A is a finite set of actions. Each action $a \in A$ is a tuple $(pre, add, del, c) \in 2^V \times 2^V \times 2^V \times \mathbb{R}_0^+$ consisting of a *precondition*, *add and delete list*, and action costs. (We often only give a 3-tuple if there are no action costs.)
- $s_I \in S$ is the initial state (complete state description).
- $g \subseteq V$ is the goal description (encodes a set of goal states).

The STRIPS Formalism, cont'd II

Action application:

- An action $a \in A$ is called *applicable* (or executable) in a state $s \in S$ if and only if $pre(a) \subseteq s$. Often, this is given by a function: $\tau(a, s) \Leftrightarrow pre(a) \subseteq s$.

The STRIPS Formalism, cont'd II

Action application:

- An action $a \in A$ is called *applicable* (or executable) in a state $s \in S$ if and only if $pre(a) \subseteq s$. Often, this is given by a function: $\tau(a, s) \Leftrightarrow pre(a) \subseteq s$.
- If $\tau(a, s)$ holds, its application results into the successor state $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$. $\gamma : A \times S \to S$ is called the *state transition function*.

The STRIPS Formalism, cont'd II

Action application:

- An action $a \in A$ is called *applicable* (or executable) in a state $s \in S$ if and only if $pre(a) \subseteq s$. Often, this is given by a function: $\tau(a, s) \Leftrightarrow pre(a) \subseteq s$.
- If $\tau(a, s)$ holds, its application results into the successor state $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$. $\gamma : A \times S \to S$ is called the *state transition function*.
- An action sequence $\bar{a} = a_0, \ldots, a_{n-1}$ is applicable in a state s_0 if and only if for all $0 \leq i \leq n-1$ a_i is applicable in s_i, where for all $1 \leq i \leq n$ s_i is the resulting state of applying $a_0, \ldots, a_i$ to $s_0 = s_I$. Often, the state transition function is extended to work on action sequences as well $\gamma : A^* \times S \to S$.

The STRIPS Formalism, cont'd III

Solution:

An action sequence $\bar{a}$ consisting of 0 (empty sequence) or more actions is called a *plan* or *solution* to a STRIPS planning problem if and only if:

The STRIPS Formalism, cont'd III

Solution:

An action sequence $\bar{a}$ consisting of 0 (empty sequence) or more actions is called a *plan* or *solution* to a STRIPS planning problem if and only if:

- $\bar{a}$ is applicable in s_I.

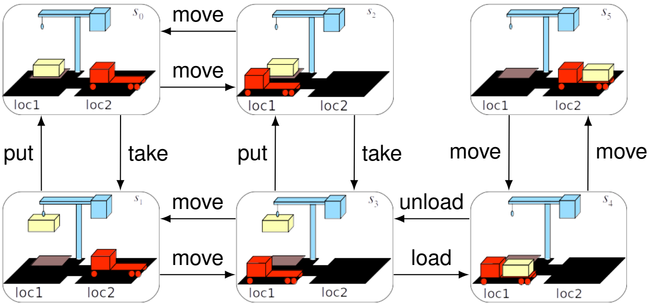
The STRIPS Formalism, cont'd III

Solution:

An action sequence $\bar{a}$ consisting of 0 (empty sequence) or more actions is called a *plan* or *solution* to a STRIPS planning problem if and only if:

- $\bar{a}$ is applicable in s_I.
- $\bar{a}$ results into a goal state, i.e., $\gamma(\bar{a}, s_I) \supseteq g$.

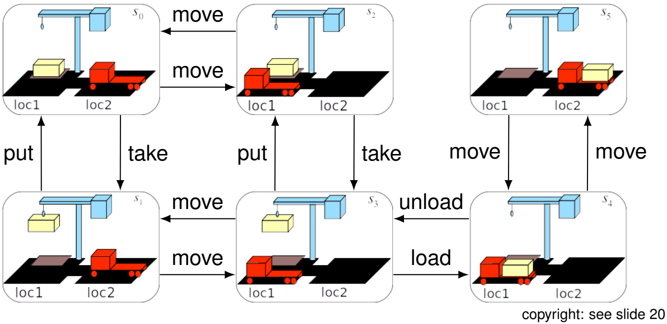
Example – Exercise!



copyright: see slide 20

Exercise:
Model a classical planning problem $\mathcal{P} = \langle V, A, s_I, g \rangle$ with the actions and states as indicated above.

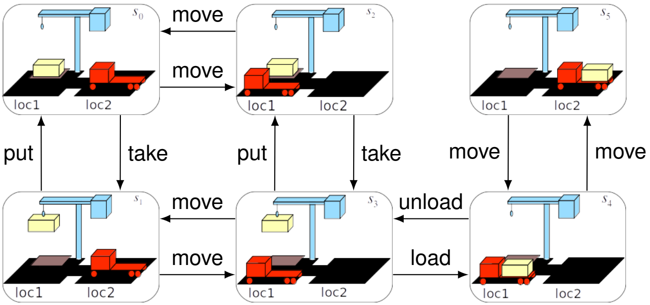
Example – Exercise!



STRIPS planning problem:

- V: {CrateAtLoc1, HoldCrate, TruckAtLoc1, TruckAtLoc2, CrateInTruck}
- A: {take, put, moveLeft, moveRight, load, unload} (see following slides)
- s_I: {CrateAtLoc1, TruckAtLoc2}
- g: {CrateInTruck, TruckAtLoc2}

Example – Exercise!



copyright: see slide 20

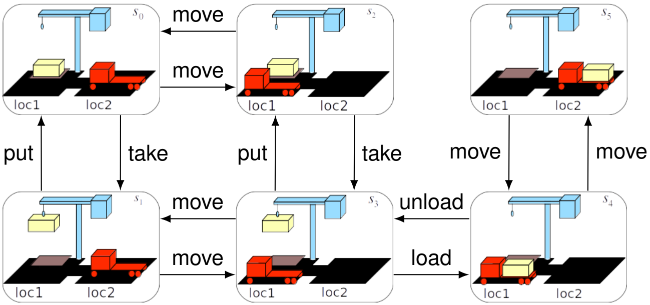
take

pre: {CrateAtLoc1}
add: {HoldCrate}
del: {CrateAtLoc1}

put

pre: {HoldCrate}
add: {CrateAtLoc1}
del: {HoldCrate}

Example – Exercise!

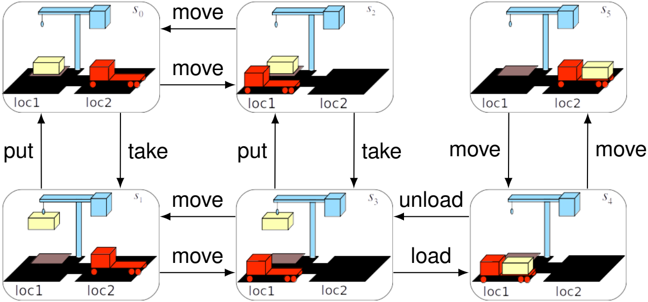


copyright: see slide 20

moveLeft
pre: {TruckAtLoc2}
add: {TruckAtLoc1}
del: {TruckAtLoc2}

moveRight
pre: {TruckAtLoc1}
add: {TruckAtLoc2}
del: {TruckAtLoc1}

Example – Exercise!



load

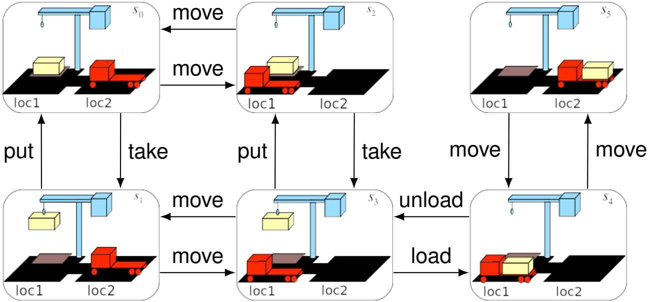
pre: {HoldCrate, TruckAtLoc1}
add: {CrateInTruck}
del: {HoldCrate}

unload

pre: {CrateInTruck, TruckAtLoc1}
add: {HoldCrate}
del: {CrateInTruck}

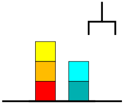
State Transition System

Every classical planning problem is a compact representation of a *state transition system*, i.e., of how states are transformed into each other.



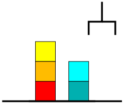
copyright: see slide 20

Size Increase of the State Space in Blocks World



- *n* blocks, 1 gripper.
- A single action either takes a block with the gripper or puts a block we are holding onto some other block/the table.

Size Increase of the State Space in Blocks World



- *n* blocks, 1 gripper.
- A single action either takes a block with the gripper or puts a block we are holding onto some other block/the table.

blocks	states
1	1
2	3
3	13
4	73
5	501
6	4051
7	37633
8	394353
9	4596553

blocks	states
10	58941091
11	824073141
12	12470162233
13	202976401213
14	3535017524403
15	65573803186921
16	1290434218669921
17	26846616451246353
18	588633468315403843

State Transition System, cont'd I

Definition (State Transition System)

A state transition system is a 6-tuple (S, L, c, T, I, G), where

- S is a finite set of states.

State Transition System, cont'd I

Definition (State Transition System)

A state transition system is a 6-tuple (S, L, c, T, I, G), where

- S is a finite set of states.
- L is a finite set of transition labels.

State Transition System, cont'd I

Definition (State Transition System)

A state transition system is a 6-tuple (S, L, c, T, I, G), where

- S is a finite set of states.
- L is a finite set of transition labels.
- $c : L \to \mathbb{R}_0^+$ is a cost function.

State Transition System, cont'd I

Definition (State Transition System)

A state transition system is a 6-tuple (S, L, c, T, I, G), where

- S is a finite set of states.
- L is a finite set of transition labels.
- $c : L \to \mathbb{R}_0^+$ is a cost function.
- $T \subseteq S \times L \times S$ is the transition relation.

State Transition System, cont'd I

Definition (State Transition System)

A state transition system is a 6-tuple (S, L, c, T, I, G), where

- S is a finite set of states.
- L is a finite set of transition labels.
- $c : L \to \mathbb{R}_0^+$ is a cost function.
- $T \subseteq S \times L \times S$ is the transition relation.
- $I \in S$ is the initial state.

State Transition System, cont'd I

Definition (State Transition System)

A state transition system is a 6-tuple (S, L, c, T, I, G), where

- S is a finite set of states.
- L is a finite set of transition labels.
- $c : L \to \mathbb{R}_0^+$ is a cost function.
- $T \subseteq S \times L \times S$ is the transition relation.
- $I \in S$ is the initial state.
- $G \subseteq S$ is the set of goal states.

State Transition System, cont'd II

Some further terminologies:

- A transition system is called *deterministic* if for all states s and labels l there is at most one state s', such that $(s, l, s') \in T$. Otherwise, it's called *non-deterministic*.

State Transition System, cont'd II

Some further terminologies:

- A transition system is called *deterministic* if for all states s and labels l there is at most one state s', such that $(s, l, s') \in T$. Otherwise, it's called *non-deterministic*.
- A state s' is called *reachable from a state s* if there is a sequence of transitions from s to s'.

State Transition System, cont'd II

Some further terminologies:

- A transition system is called *deterministic* if for all states s and labels l there is at most one state s', such that $(s, l, s') \in T$. Otherwise, it's called *non-deterministic*.
- A state s' is called *reachable from a state s* if there is a sequence of transitions from s to s'.
- A state s is called *reachable* (without mentioning another state) if it is reachable from I.

State Transition System, cont'd II

Some further terminologies:

- A transition system is called *deterministic* if for all states s and labels l there is at most one state s', such that $(s, l, s') \in T$. Otherwise, it's called *non-deterministic*.
- A state s' is called *reachable from a state s* if there is a sequence of transitions from s to s'.
- A state s is called *reachable* (without mentioning another state) if it is reachable from I.
- A state s is called *dead-end* if no goal state is reachable from s.

State Transition System, cont'd II

Some further terminologies:

- A transition system is called *deterministic* if for all states s and labels l there is at most one state s', such that $(s, l, s') \in T$. Otherwise, it's called *non-deterministic*.
- A state s' is called *reachable from a state s* if there is a sequence of transitions from s to s'.
- A state s is called *reachable* (without mentioning another state) if it is reachable from I.
- A state s is called *dead-end* if no goal state is reachable from s.
- A transition system is called *solvable* if some goal state is reachable.

Questions

Questions

Which of the following transition systems has a *reachable* dead-end? (We assume that the states of the transition system are *world states* encoding the respective current situation.)

1. FreeCell?
2. 15-Puzzle?
3. Cranes in the Harbor?
4. Dining Philosophers?

Questions

Questions

Which of the following transition systems has a *reachable* dead-end? (We assume that the states of the transition system are *world states* encoding the respective current situation.)

1. FreeCell?
2. 15-Puzzle?
3. Cranes in the Harbor?
4. Dining Philosophers?

1. Yes. Choices can be wrong and cannot be taken back.

Questions

Questions

Which of the following transition systems has a *reachable* dead-end? (We assume that the states of the transition system are *world states* encoding the respective current situation.)

1. FreeCell?
2. 15-Puzzle?
3. Cranes in the Harbor?
4. Dining Philosophers?

1. Yes. Choices can be wrong and cannot be taken back.
2. No. All actions can be inverted. There are, however, combinations of states (initial/current state and goal state) that do not admit a solution. That is, for every initial state one can construct unreachable states.

Questions

Questions

Which of the following transition systems has a *reachable* dead-end? (We assume that the states of the transition system are *world states* encoding the respective current situation.)

1. FreeCell?
2. 15-Puzzle?
3. Cranes in the Harbor?
4. Dining Philosophers?

1. Yes. Choices can be wrong and cannot be taken back.
2. No. All actions can be inverted. There are, however, combinations of states (initial/current state and goal state) that do not admit a solution. That is, for every initial state one can construct unreachable states.
3. No. All actions are invertible.

Questions

Questions

Which of the following transition systems has a *reachable* dead-end? (We assume that the states of the transition system are *world states* encoding the respective current situation.)

1. FreeCell?
2. 15-Puzzle?
3. Cranes in the Harbor?
4. Dining Philosophers?

1. Yes. Choices can be wrong and cannot be taken back.
2. No. All actions can be inverted. There are, however, combinations of states (initial/current state and goal state) that do not admit a solution. That is, for every initial state one can construct unreachable states.
3. No. All actions are invertible.
4. Yes. If all philosophers take their right fork, they starve do death.

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: *fillTank* $<$ *drive*). Some issues that need to be solved:

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: *fillTank* $<$ *drive*). Some issues that need to be solved:

- How to "identify" a certain action? Assume an action, say *drive*, occurs multiple times within a plan.

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: *fillTank* $<$ *drive*). Some issues that need to be solved:

- How to "identify" a certain action? Assume an action, say *drive*, occurs multiple times within a plan.
  - In action sequences: no problem! (trivial)

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: *fillTank* $<$ *drive*). Some issues that need to be solved:

- How to "identify" a certain action? Assume an action, say *drive*, occurs multiple times within a plan.
  - In action sequences: no problem! (trivial)
  - In partial plans: constraints like *fillTank* $<$ *drive* do not work anymore, since it's unclear which *drive* is meant.

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: $\mathit{fillTank} < \mathit{drive}$). Some issues that need to be solved:

- How to "identify" a certain action? Assume an action, say *drive*, occurs multiple times within a plan.
  - In action sequences: no problem! (trivial)
  - In partial plans: constraints like $\mathit{fillTank} < \mathit{drive}$ do not work anymore, since it's unclear which *drive* is meant.

  $\rightarrow$ We introduce labels! Labeled actions are called *plan steps*.

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: $\mathit{fillTank} < \mathit{drive}$). Some issues that need to be solved:

- How to "identify" a certain action? Assume an action, say *drive*, occurs multiple times within a plan.
  - In action sequences: no problem! (trivial)
  - In partial plans: constraints like $\mathit{fillTank} < \mathit{drive}$ do not work anymore, since it's unclear which *drive* is meant.
  - $\rightarrow$ We introduce labels! Labeled actions are called *plan steps*.
- How to define executability if, due to the partial order, there are many possible linearizations?

Introduction

So far, plans (i.e., solutions) are *action sequences*, i.e., totally ordered. We now introduce a generalization to partially ordered actions, which are based upon *partial plans*.

Informally, a *partial plan* is a set of partially ordered actions. Instead of specifying a sequence of actions, we have a set of ordering constraints (like: $\textit{fillTank} < \textit{drive}$). Some issues that need to be solved:

- How to "identify" a certain action? Assume an action, say *drive*, occurs multiple times within a plan.
  - In action sequences: no problem! (trivial)
  - In partial plans: constraints like $\textit{fillTank} < \textit{drive}$ do not work anymore, since it's unclear which *drive* is meant.
  - → We introduce labels! Labeled actions are called *plan steps*.
- How to define executability if, due to the partial order, there are many possible linearizations?
  - → In terms of *causal links* or by checking the linearizations.

Additional Definitions

Let $ps = l{:}a$ be plan step, i.e., l is a label and $a = (pre, add, del)$ an action.

Additional Definitions

Let $ps = l{:}a$ be plan step, i.e., l is a label and $a = (pre, add, del)$ an action.

Then, in addition to writing $pre(a)$, $add(a)$, and $del(a)$ for an action a, we also write $pre(ps)$, $add(ps)$, and $del(ps)$ to refer to these action elements.

Additional Definitions

Let $ps = l{:}a$ be plan step, i.e., l is a label and $a = (pre, add, del)$ an action.

Then, in addition to writing $pre(a)$, $add(a)$, and $del(a)$ for an action a, we also write $pre(ps)$, $add(ps)$, and $del(ps)$ to refer to these action elements.

Let $\prec \subseteq X \times X$ be a strict partial order. Then, a linearization of $\prec$ is a sequence $x_1, \ldots, x_n$ of X's elements that does not contradict the ordering constraints.

Partial Plans (Definition)

Definition (Partial Plan)

Given a planning problem $\langle V, A, s_I, g\rangle$, a *partial plan* is a tuple $(PS, \prec, CL)$ consisting of:

- PS, the finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a *label* (an arbitrary symbol) unique in PS and $a \in A$ is an action.

[1] Strict partial orders are: irreflexive, transitive, and asymmetric.

Partial Plans (Definition)

Definition (Partial Plan)

Given a planning problem $\langle V, A, s_I, g \rangle$, a *partial plan* is a tuple $(PS, \prec, CL)$ consisting of:

- PS, the finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a *label* (an arbitrary symbol) unique in PS and $a \in A$ is an action.
- $\prec \subseteq PS \times PS$ is a strict partial order[1] on PS.

[1] Strict partial orders are: irreflexive, transitive, and asymmetric.

Partial Plans (Definition)

Definition (Partial Plan)

Given a planning problem $\langle V, A, s_I, g \rangle$, a *partial plan* is a tuple $(PS, \prec, CL)$ consisting of:

- PS, the finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a *label* (an arbitrary symbol) unique in PS and $a \in A$ is an action.
- $\prec \subseteq PS \times PS$ is a strict partial order[1] on PS.
- CL is a finite set of causal links. A causal link $ps \xrightarrow{v} ps' \in CL$ ...

[1] Strict partial orders are: irreflexive, transitive, and asymmetric.

Partial Plans (Definition)

Definition (Partial Plan)

Given a planning problem $\langle V, A, s_I, g \rangle$, a *partial plan* is a tuple $(PS, \prec, CL)$ consisting of:

- PS, the finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a *label* (an arbitrary symbol) unique in PS and $a \in A$ is an action.
- $\prec \subseteq PS \times PS$ is a strict partial order[1] on PS.
- CL is a finite set of causal links. A causal link $ps \xrightarrow{v} ps' \in CL$ ...
  - consists of two plan steps $ps, ps' \in PS$ and a state variable $v \in V$ that occurs both in $add(ps)$ and in $pre(ps')$.

[1] Strict partial orders are: irreflexive, transitive, and asymmetric.

Partial Plans (Definition)

Definition (Partial Plan)

Given a planning problem $\langle V, A, s_I, g \rangle$, a *partial plan* is a tuple $(PS, \prec, CL)$ consisting of:

- PS, the finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a *label* (an arbitrary symbol) unique in PS and $a \in A$ is an action.
- $\prec \subseteq PS \times PS$ is a strict partial order[1] on PS.
- CL is a finite set of causal links. A causal link $ps \xrightarrow{v} ps' \in CL$ ...
  - consists of two plan steps $ps, ps' \in PS$ and a state variable $v \in V$ that occurs both in $add(ps)$ and in $pre(ps')$.
  - implies $(ps, ps') \in \prec$ and there is no ps'' with $ps'' \xrightarrow{v} ps' \in CL$ (unique supporter).

[1] Strict partial orders are: irreflexive, transitive, and asymmetric.

Recap on Partial Orders

If $\prec \subseteq X \times X$ is a strict partial order, then for all $x, y, z \in X$ holds:

irreflexive not $x \prec x$,
i.e., $(x, x) \notin \prec$

Recap on Partial Orders

If $\prec \subseteq X \times X$ is a strict partial order, then for all $x, y, z \in X$ holds:

irreflexive: not $x \prec x$,
i.e., $(x, x) \notin \prec$

transitive: if $x \prec y$ and $y \prec z$, then $x \prec z$,
i.e., $\{(x, y), (y, z)\} \subseteq \prec$ implies $(x, z) \in \prec$

Recap on Partial Orders

If $\prec \subseteq X \times X$ is a strict partial order, then for all $x, y, z \in X$ holds:

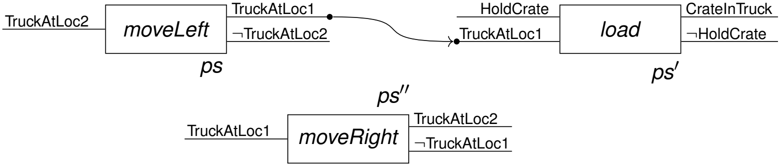
irreflexive not $x \prec x$,
i.e., $(x, x) \notin \prec$

transitive if $x \prec y$ and $y \prec z$, then $x \prec z$,
i.e., $\{(x, y), (y, z)\} \subseteq \prec$ implies $(x, z) \in \prec$

antisymmetric if $x \prec y$, then not $y \prec x$,
i.e., $(x, y) \in \prec$ implies $(y, x) \notin \prec$

Example for Causal Links and Causal Threats

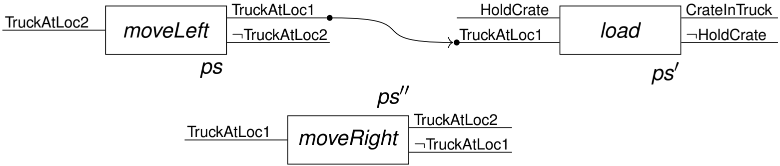
Consider the following partial plan from the *Cranes in the Harbor* domain. The causal link $ps \xrightarrow{\text{TruckAtLoc1}} ps'$ documents the achievement of the precondition of ps' by ps.



What can we observe?

Example for Causal Links and Causal Threats

Consider the following partial plan from the *Cranes in the Harbor* domain. The causal link $ps \xrightarrow{\text{TruckAtLoc1}} ps'$ documents the achievement of the precondition of ps' by ps.

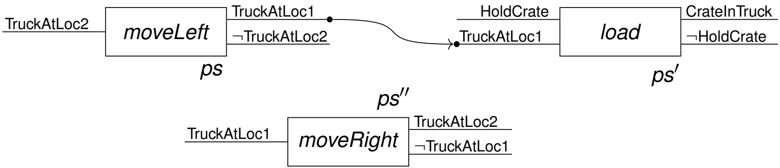


What can we observe?

- The plan step ps'' is yet unordered with respect to the interval between ps and ps'.

Example for Causal Links and Causal Threats

Consider the following partial plan from the *Cranes in the Harbor* domain. The causal link $ps \xrightarrow{\text{TruckAtLoc1}} ps'$ documents the achievement of the precondition of ps' by ps.



What can we observe?

- The plan step ps'' is yet unordered with respect to the interval between ps and ps'.

→ ps'' could be ordered between ps and ps'.
Thus: it *threatens* the causal link.

Causal Threats

Definition (Causal Threat)

Let $(PS, \prec, CL)$ be a partial plan. A *causal threat* consists of the plan steps $ps, ps' \in PS$, a causal link $ps \xrightarrow{v} ps'$, and the *threatening plan step* $ps'' \in PS$ if and only if

Causal Threats

Definition (Causal Threat)

Let $(PS, \prec, CL)$ be a partial plan. A *causal threat* consists of the plan steps $ps, ps' \in PS$, a causal link $ps \xrightarrow{v} ps'$, and the *threatening plan step* $ps'' \in PS$ if and only if

- $v \in del(ps'')$

Causal Threats

Definition (Causal Threat)

Let $(PS, \prec, CL)$ be a partial plan. A *causal threat* consists of the plan steps $ps, ps' \in PS$, a causal link $ps \xrightarrow{v} ps'$, and the *threatening plan step* $ps'' \in PS$ if and only if

- $v \in del(ps'')$
- The ordering constraints allow ps'' to be ordered between ps and ps', i.e., $(\prec \cup \{(ps, ps''), (ps'', ps')\})^*$ is a strict partial order. (* denotes the transitive closure.)

Causal Threats

Definition (Causal Threat)

Let $(PS, \prec, CL)$ be a partial plan. A *causal threat* consists of the plan steps $ps, ps' \in PS$, a causal link $ps \xrightarrow{v} ps'$, and the *threatening plan step* $ps'' \in PS$ if and only if

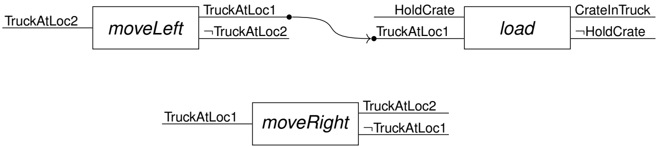
- $v \in del(ps'')$
- The ordering constraints allow ps'' to be ordered between ps and ps', i.e., $(\prec \cup \{(ps, ps''), (ps'', ps')\})^*$ is a strict partial order. (* denotes the transitive closure.)

Note:

In case we allow negative preconditions, the definition of causal threats needs to be extended.

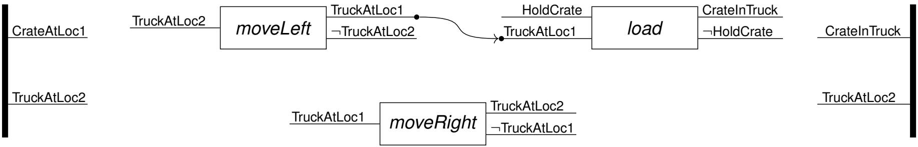
Encoding Initial State and Goal Description

How to represent the initial state and goal description?



Encoding Initial State and Goal Description

How to represent the initial state and goal description?



- We represent these states as artificial actions:
  - *Initial State.* For s_I introduce an action $a_I \notin A$ with $pre(a_I) = del(a_I) = \emptyset$ and $add(a_I) = s_I$.
  - *Goal Description.* For g introduce an action $a_g \notin A$ with $pre(a_g) = g$ and $add(a_g) = del(a_I) = \emptyset$.
  - *Ordering Constraints.* Enforce that a_I is the very first action and a_g is the very last.

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps *PS* is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps *PS* is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why?

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps *PS* is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why? Because $a_I \notin A$ and $a_g \notin A$, but $init{:}a_I \in PS$ and $goal{:}a_g \in PS$.

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps *PS* is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why? Because $a_I \notin A$ and $a_g \notin A$, but $init{:}a_I \in PS$ and $goal{:}a_g \in PS$.
- Thus, in principle, we had to alter the respective definition as follows: Let $\mathcal{P} = \langle V, A, s_I, g\rangle$ be a planning problem. Then, a *partial plan* is a tuple $P = (PS, \prec, CL)$, where:

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps PS is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why? Because $a_I \notin A$ and $a_g \notin A$, but $init{:}a_I \in PS$ and $goal{:}a_g \in PS$.
- Thus, in principle, we had to alter the respective definition as follows: Let $\mathcal{P} = \langle V, A, s_I, g \rangle$ be a planning problem. Then, a *partial plan* is a tuple $P = (PS, \prec, CL)$, where:
  - PS' is a finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a unique label and $a \in A$ is an action.

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps *PS* is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why? Because $a_I \notin A$ and $a_g \notin A$, but $init{:}a_I \in PS$ and $goal{:}a_g \in PS$.
- Thus, in principle, we had to alter the respective definition as follows: Let $\mathcal{P} = \langle V, A, s_I, g \rangle$ be a planning problem. Then, a *partial plan* is a tuple $P = (PS, \prec, CL)$, where:
  - PS' is a finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a unique label and $a \in A$ is an action.
  - $PS = PS' \cup \{init{:}a_I, goal{:}a_g\}$.

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps PS is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why? Because $a_I \notin A$ and $a_g \notin A$, but $init{:}a_I \in PS$ and $goal{:}a_g \in PS$.
- Thus, in principle, we had to alter the respective definition as follows: Let $\mathcal{P} = \langle V, A, s_I, g \rangle$ be a planning problem. Then, a *partial plan* is a tuple $P = (PS, \prec, CL)$, where:
  - PS' is a finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a unique label and $a \in A$ is an action.
  - $PS = PS' \cup \{init{:}a_I, goal{:}a_g\}$.
  - $\prec \subseteq PS \times PS$ is a strict partial order on PS with $(init{:}a_I, ps) \in \prec$ for all $ps \neq init{:}a_I$ and $(ps, goal{:}a_g) \in \prec$ for all $ps \neq goal{:}a_g$.

Encoding Initial State and Goal Description, cont'd

How to represent the initial state and goal description?

- Please note that according to the definition of partial plans (cf. sl. 46) the set of plan steps *PS* is defined as a set of plan steps $l{:}a$, where l is a unique *label* symbol and $a \in A$ is an action.
- This is, regarded *very technically*, not correct anymore. Why? Because $a_I \notin A$ and $a_g \notin A$, but $init{:}a_I \in PS$ and $goal{:}a_g \in PS$.
- Thus, in principle, we had to alter the respective definition as follows: Let $\mathcal{P} = \langle V, A, s_I, g \rangle$ be a planning problem. Then, a *partial plan* is a tuple $P = (PS, \prec, CL)$, where:
  - PS' is a finite set of *plan steps*. Each plan step $ps \in PS$ is a pair $l{:}a$, where l is a unique label and $a \in A$ is an action.
  - $PS = PS' \cup \{init{:}a_I, goal{:}a_g\}$.
  - $\prec \subseteq PS \times PS$ is a strict partial order on PS with $(init{:}a_I, ps) \in \prec$ for all $ps \neq init{:}a_I$ and $(ps, goal{:}a_g) \in \prec$ for all $ps \neq goal{:}a_g$.
  - The rest of the definition does not change.

Flaws

To specify whether a given partial plan is a solution, we use the concept of *flaws*. That is, flaws are plan deficiencies that show why a partial plan *is not* a solution.

Flaws

To specify whether a given partial plan is a solution, we use the concept of *flaws*. That is, flaws are plan deficiencies that show why a partial plan *is not* a solution.

There are two kinds of flaws:

Flaws

To specify whether a given partial plan is a solution, we use the concept of *flaws*. That is, flaws are plan deficiencies that show why a partial plan *is not* a solution.

There are two kinds of flaws:

- *Causal Threats* (as defined above).

Flaws

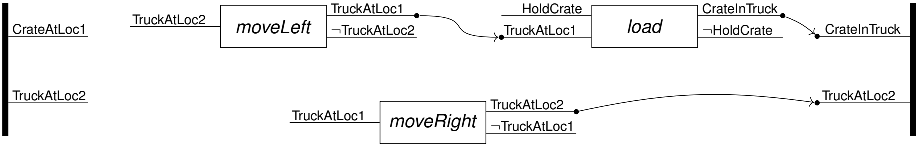
To specify whether a given partial plan is a solution, we use the concept of *flaws*. That is, flaws are plan deficiencies that show why a partial plan *is not* a solution.

There are two kinds of flaws:

- *Causal Threats* (as defined above).
- *Open Preconditions.* We call a pair (v, ps) consisting of a plan step $ps \in PS$ and its precondition $v \in pre(ps)$ an *open precondition* if and only if there does not exist a causal link $ps' \xrightarrow{v} ps \in CL$.

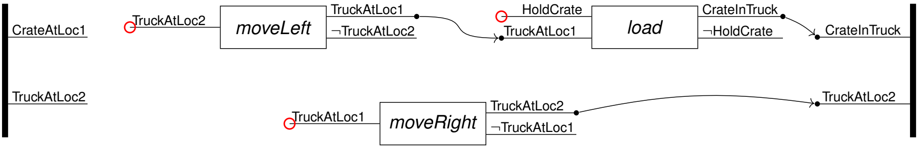
Flaws, cont'd

Which flaws does this partial plan possess?



Flaws, cont'd

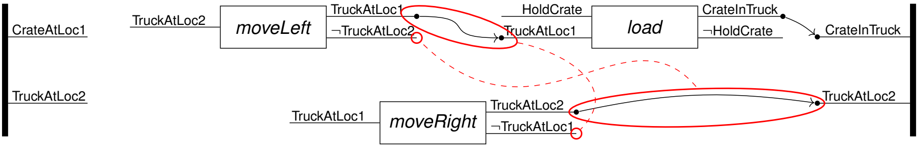
Which flaws does this partial plan possess?



- Three open preconditions.

Flaws, cont'd

Which flaws does this partial plan possess?



- Three open preconditions.
- Two causal treats.

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

- P_I is the initial partial plan and contains an encoding of the initial state s_I and the goal description g (as described before).

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

- P_I is the initial partial plan and contains an encoding of the initial state s_I and the goal description g (as described before).
- $\langle V, A, s_I, g \rangle$ is a classical planning problem.

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

- P_I is the initial partial plan and contains an encoding of the initial state s_I and the goal description g (as described before).
- $\langle V, A, s_I, g \rangle$ is a classical planning problem.

The set of solutions of $\mathcal{P}$ is defined as the set of all partial plans $P \supseteq P_I$ without flaws.

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

- P_I is the initial partial plan and contains an encoding of the initial state s_I and the goal description g (as described before).
- $\langle V, A, s_I, g \rangle$ is a classical planning problem.

The set of solutions of $\mathcal{P}$ is defined as the set of all partial plans $P \supseteq P_I$ without flaws.

Note:

POCL problems differ from classical problems in only two ways:

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

- P_I is the initial partial plan and contains an encoding of the initial state s_I and the goal description g (as described before).
- $\langle V, A, s_I, g \rangle$ is a classical planning problem.

The set of solutions of $\mathcal{P}$ is defined as the set of all partial plans $P \supseteq P_I$ without flaws.

Note:

POCL problems differ from classical problems in only two ways:

- In addition to the initial state and goal descriptions, intermediate actions (and causal links) can be specified.

Problem Definition

Definition (POCL Planning Problem)

A tuple $\mathcal{P} = \langle V, A, P_I \rangle$ is called a Partial-Order Causal-Link (POCL) planning problem, where

- P_I is the initial partial plan and contains an encoding of the initial state s_I and the goal description g (as described before).
- $\langle V, A, s_I, g \rangle$ is a classical planning problem.

The set of solutions of $\mathcal{P}$ is defined as the set of all partial plans $P \supseteq P_I$ without flaws.

Note:

POCL problems differ from classical problems in only two ways:

- In addition to the initial state and goal descriptions, intermediate actions (and causal links) can be specified.
- Solutions are partial plans rather than action sequences.

Linearizations of POCL Solutions

Theorem

Let $\mathcal{P} = \langle V, A, P_I \rangle$ be a POCL planning problem with initial state s_I and goal description g. Let $P = (PS, \prec, CL)$ be a plan for $\mathcal{P}$.

Then, each linearization of P is a solution in the classical sense, i.e., they are all executable in s_I and generate a goal state $s' \supseteq g$.

Linearizations of POCL Solutions

Theorem

Let $\mathcal{P} = \langle V, A, P_I \rangle$ be a POCL planning problem with initial state s_I and goal description g. Let $P = (PS, \prec, CL)$ be a plan for $\mathcal{P}$.

Then, each linearization of P is a solution in the classical sense, i.e., they are all executable in s_I and generate a goal state $s' \supseteq g$.

Proof:

Linearizations of POCL Solutions

Theorem

Let $\mathcal{P} = \langle V, A, P_I \rangle$ be a POCL planning problem with initial state s_I and goal description g. Let $P = (PS, \prec, CL)$ be a plan for $\mathcal{P}$.

Then, each linearization of P is a solution in the classical sense, i.e., they are all executable in s_I and generate a goal state $s' \supseteq g$.

Proof:
Exercise!

Linearizations of POCL Solutions, cont'd I

Theorem

Let $\mathcal{P} = \langle V, A, P_I \rangle$ be a POCL planning problem.
Then, there exist partial plans, for which all linearizations solve the planning problem, but there is no *plan* that possesses the same set of linearizations.

Linearizations of POCL Solutions, cont'd I

Theorem

Let $\mathcal{P} = \langle V, A, P_I \rangle$ be a POCL planning problem.
Then, there exist partial plans, for which all linearizations solve the planning problem, but there is no *plan* that possesses the same set of linearizations.

Proof:

Linearizations of POCL Solutions, cont'd I

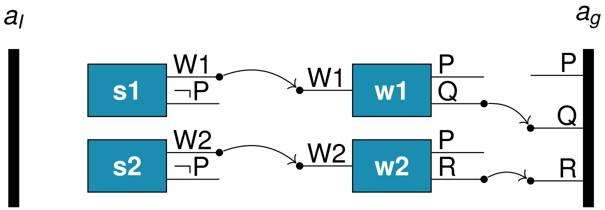
Theorem

Let $\mathcal{P} = \langle V, A, P_I \rangle$ be a POCL planning problem.
Then, there exist partial plans, for which all linearizations solve the planning problem, but there is no *plan* that possesses the same set of linearizations.

Proof:
By example (next slide), which was provided by Kambhampati, published by: David McAllester and David Rosenblitt. "Systematic Nonlinear Planning". In: *Proc. of the 9th National Conf. on Artificial Intelligence (AAAI 1991)*. AAAI Press, 1991, pp. 634–639

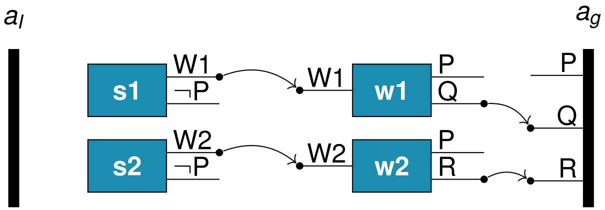
Linearizations of POCL Solutions, cont'd II

Proof:



Linearizations of POCL Solutions, cont'd II

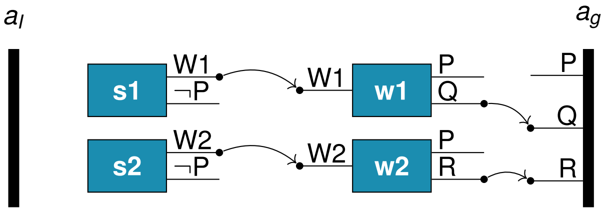
Proof:



- All six linearizations (s1,w1,s2,w2 ; s1,s2,w1,w2 ; s1,s2,w2,w1 ; s2,w2,s1,w1 ; s1,s2,w2,w1 ; s2,s1,w1,w2) are classical solutions:

Linearizations of POCL Solutions, cont'd II

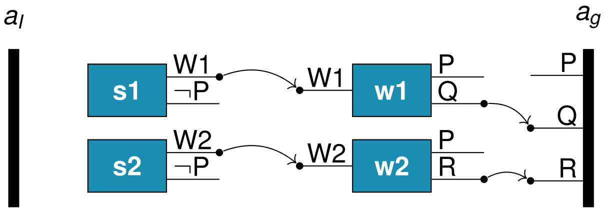
Proof:



- All six linearizations (s1,w1,s2,w2 ; s1,s2,w1,w2 ; s1,s2,w2,w1 ; s2,w2,s1,w1 ; s1,s2,w2,w1 ; s2,s1,w1,w2) are classical solutions:
  - The preconditions $W1$, $W2$, Q, R cannot possibly be problematic.

Linearizations of POCL Solutions, cont'd II

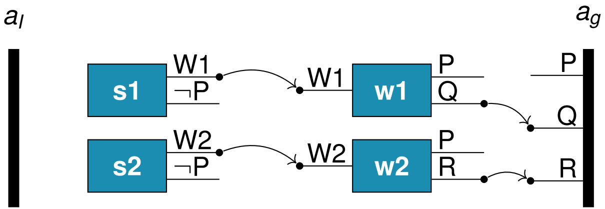
Proof:



- All six linearizations (s1,w1,s2,w2 ; s1,s2,w1,w2 ; s1,s2,w2,w1 ; s2,w2,s1,w1 ; s1,s2,w2,w1 ; s2,s1,w1,w2) are classical solutions:
  - The preconditions $W1$, $W2$, Q, R cannot possibly be problematic.
  - P will be produced by either w1 or w2.

Linearizations of POCL Solutions, cont'd II

Proof:



- All six linearizations (s1,w1,s2,w2 ; s1,s2,w1,w2 ; s1,s2,w2,w1 ; s2,w2,s1,w1 ; s1,s2,w2,w1 ; s2,s1,w1,w2) are classical solutions:
  - The preconditions $W1$, $W2$, Q, R cannot possibly be problematic.
  - P will be produced by either w1 or w2.
- However, adding a causal link involving precondition (P, a_g) will introduce a threat and therefore an ordering.

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and
  - can be expressed with the STRIPS formalism – the best-known formalization for classical planning.

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and
  - can be expressed with the STRIPS formalism – the best-known formalization for classical planning.
- Partial-Order Causal-Link (POCL) planning problems are a closely related problem class. They:

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and
  - can be expressed with the STRIPS formalism – the best-known formalization for classical planning.
- Partial-Order Causal-Link (POCL) planning problems are a closely related problem class. They:
  - originate from the POCL *planning algorithm* for classical problems,

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and
  - can be expressed with the STRIPS formalism – the best-known formalization for classical planning.
- Partial-Order Causal-Link (POCL) planning problems are a closely related problem class. They:
  - originate from the POCL *planning algorithm* for classical problems,
  - base upon partially ordered plans, and

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and
  - can be expressed with the STRIPS formalism – the best-known formalization for classical planning.
- Partial-Order Causal-Link (POCL) planning problems are a closely related problem class. They:
  - originate from the POCL *planning algorithm* for classical problems,
  - base upon partially ordered plans, and
  - on causal links for annotating established action preconditions.

Summary

- Planning is concerned with achieving goals or tasks by reasoning about actions (and, possibly, action hierarchies).
- Planning research goes far beyond just plan generation.
- Classical planning problems:
  - are discrete, deterministic, fully observable, single-agent,
  - can be considered the base case of planning,
  - can compactly and conveniently represent huge transition systems, and
  - can be expressed with the STRIPS formalism – the best-known formalization for classical planning.
- Partial-Order Causal-Link (POCL) planning problems are a closely related problem class. They:
  - originate from the POCL *planning algorithm* for classical problems,
  - base upon partially ordered plans, and
  - on causal links for annotating established action preconditions.
  - They can compactly represent many totally ordered linearizations.