Events

- Lectures
  - Room 301
  - Every Thursday at 14:00
  - Please enroll to the lecture on http://campus.studium.kit.edu

- Exercises
  - Room -119
  - Every second Wednesday (starting 31.10.) at 14:00

- Exams
  - Oral/Written examination
  - Bonus points for homework improve the grade
  - Dates are flexible
Homeworks and Lecture Notes

- You get homework points for doing homework and showing up with them on the exercises.
- You will have the possibility to collect at least 120 points for homeworks during the semester (plus a lot more bonus points).
- You must collect at least 60 points to pass the exercises and be allowed to participate in the oral exam.
- Extra points improve your grade:
  - $\geq 70$ points improves grade by 0.3
  - $\geq 90$ points improves grade by 0.7
  - $\geq 110$ points improves grade by 1.0
Contact

- Tomas Balyo
  - mailto:tomas.balyo@kit.edu
  - room: 208
- Dominik Schreiber
  - mailto:dominik.schreiber@kit.edu
  - room: 209
- Homepage
  - https://baldur.iti.kit.edu/plan/
  - Contains all the slides and homework assignments
Goals of this lecture

- What is planning and scheduling?
  - Why is it important?
- How to use a Planner efficiently
  - How to encode your problems into PDDL
- How do Planners work
  - Algorithms
- How to make a Planner
  - Implementation techniques
- How do scheduling algorithms work
Planning vs Scheduling

- **Planning**
  - given a description of the **current state**, a set of **possible actions**, and a **desired state** come up with a **sequence of actions = plan** that one can take to achieve the desired state.
  - belongs to the category of Artificial Intelligence.
  - high complexity, P-SPACE hard or even Undecidable.

- **Scheduling**
  - given a collection of **actions** and restricted **resources** decide how to execute **all the actions** in an efficient manner (create a schedule).
  - belongs to the category of operations research.
  - complexity typically in P and NP
Planning Problem Example – Trucking

Initial State
- There is a truck and a package in city A
- There is a package in city B

Goal
- There are two packages in city C

Possible Actions
- (Un)loading packages from/on the truck, driving between cities
Planning Problem Example – Trucking

World State
- Truck at A, packages at A and B
- Truck at A with one package ...
- Truck at B ...
- Truck at B with both packages
- Truck at C with both packages
- both packages delivered to C

The Plan
- load package at A
- drive truck to B
- load the second package
- drive truck to C
- unload both packages
Another Planning Problem Example

Sokoban

- **Initial State**
  - There is a worker and a bunch of boxes

- **Goal**
  - All the boxes must be in goal positions

- **Possible Actions**
  - Moving with the worker
  - Pushing a box

- **Forbidden**
  - To pull boxes
  - Move through walls or boxes

www.sokobanonline.com
Scheduling Problem - School Timetable

- **Actions**
  - There is given number of each kind of lesson that needs to take place

- **Constraints**
  - Teachers can teach certain subjects, only one at a time
  - Students need to take certain subjects
  - It must fit in a week

- **Optimization Goals**
  - Finish early each day (especially on fridays)
  - No holes in the schedule
Job Shop Scheduling

- Actions
  - A job is a sequence of tasks, each task has a certain duration and must be executed on a specific machine.

- Constraints
  - A machine works on one task at a time uninterrupted
  - The tasks of a job are executed in the correct order without overlap

- Optimization Goal = finish as soon as possible

Example: job0 = [(0,3),(1,2),(2,2)], job1 = [(0,2),(2,1),(1,4)], job2 = [(1,4),(2,3)]
Motivation - Aircraft Assembly

- 570 tasks, 17 resources
- A traditional approach:
  - ARTEMIS
  - 20 hours to produce a schedule
- Intelligent Planning and Scheduling:
  - ARTEMIS substituted by a CSP
  - 30 minutes to generate an optimal schedule
  - 10-15% shorter makespan
- Savings:
  - 4 to 6 days shorter schedules
  - 200,000$ – 1,000,000$ per day
Motivation - Submarine Construction

- 7000 tasks per boat, 125 resource classes
- A traditional approach:
  - ARTEMIS
  - 6 weeks to produce a schedule (very non uniform)
- Intelligent Planning and Scheduling:
  - ARTEMIS substituted by a CSP
  - 2 days per schedule
  - uniform resource profile
- Savings:
  - 30% less overtime and sub-contracts
Motivation - Gulf War 1991 Logistics

- A traditional approach:
  - hundreds of human planners
  - months to generate the plans

- Intelligent Planning and Scheduling:
  - O-PLAN2 (a system combining planning and scheduling)

- Savings:
  - faster background creation
  - less flight missions
  - Financial savings that highly exceed all AI research supported by US government since 1956
Motivation - Mars Rovers

- Why automated planning and scheduling?
  - Communication delay 3-22 minutes one way
  - Comm. opportunity via orbiter only once a sol for about 8 minutes
  - Direct comm. to Earth only possible for few hours a day

- CASPER (Continuous Activity Scheduling, Planning, Execution and Re-planning) is the NASA’s automated planning and scheduling component of OASIS (Onboard Autonomous Science Investigation System)

- Capabilities:
  - generation of a rover activity plan based on science priorities
  - handling of opportunistic science
  - modifying the rover activity plan in response to problems or other state and resource changes
Restrictive assumptions of Planning

Compared to the “Real World” in Classical Planning

- There are finitely many states and actions
- The world state is fully observable, the agent knows the current state
- Actions are deterministic, they only have one outcome
- The world is static, it only changes by the agents actions
- Goal is defined as a set of states
- Plan is defined as a sequence of actions
Three Kinds of Planners

- Domain–specific
  - A planner designed and developed for a specific planning domain.
  - Won’t work well or at all for other planning domains.
  - Examples: Path finding algorithms, sokoban puzzle solver

- Domain–independent
  - A planner that works on any planning domain (given the restrictions on the previous slide).
  - Correctness and completeness is guaranteed, but performance may be worse than a domain-specific planner on its respective domain.

- Configurable
  - Domain independent engine, input includes info about efficient solving.
  - One example of this HTN (Hierarchical Task Network) Planning.
A classical planning problem \( \pi = (S, A, s_I, s_G) \) is a tuple where
- \( S \) represents the final set of world states
- \( A \) represents the final set of actions
- \( s_I \in S \) represents the initial state
- \( s_G \subset S \) represents the set of goal states

A plan \( P = [a_1, a_2, \ldots, a_n] \) is a sequence of actions where \( a_i \in A \) that transforms the world state from \( s_I \) to a state \( s \in s_G \).
Example: pathfinding in a graph

\[ \pi = (S, A, s_I, s_G) \] Explicit representation:

- \( S = \{ at_1, at_2, at_3, at_4, at_5, at_6 \} \)
- \( A = \{ \text{move}(at_6, at_4), \text{move}(at_4, at_6), \text{move}(at_4, at_5), \ldots \} \)
- \( s_I = at_1 \)
- \( s_G = \{ at_6 \} \)

Plan \( P = [\text{move}(at_1, at_5), \text{move}(at_5, at_4), \text{move}(at_4, at_6)] \)
Representation of World States

- In general far too many states to represent explicitly
- We represent states as a set of features
  - a set of propositions that are true (PR – Propositional rep.)
  - vector of values of finite domain variables (FDR – finite dom. rep.)

Example

- PR propositions: Tr@A, Tr@B, Tr@C, P1@A, P2@A, P1@B, P2@B, P1@C, P2@C, P1-in-Truck, P1-in-Truck
- FDR variables: TruckLocation = \{A|B|C\}, Package1Location = \{A|B|C|T\}, Package2Location = \{A|B|C|T\}
Representation of Initial State

Representing the Initial state is same as any other state

- A full description of a world state
- PR: set of all true propositions
- FDR: each variable has a value assigned from its domain

Example

- PR: Tr@A, P1@A, P2@B
- FDR: TruckLocation = A, Package1Location = A, Package2Location = B
Representation of Goal Conditions

Goal conditions are represented as a partial state

- **PR:** set of true propositions – same as any state
- **FDR:** some variables have a value assigned from their domains
- A state $s$ is goal state if $s \subseteq s_G$

Example

- **PR:** $P1@C$, $P2@C$
- **FDR:** Package1Location = $C$, Package2Location = $C$
Representation of an Action

An action \( A = (pre, eff) \) is 2-tuple where

- \( pre \) represents the **preconditions** of the action, i.e., the conditions that must hold before the action is executed.
- \( eff \) represents the **effects** of the action, i.e., the conditions that must hold after the action is executed.
- PR: both \( pre \) and \( eff \) are sets of propositions
- FDR: both \( pre \) and \( eff \) are partial assignments to the FD variables

Example: action load package 1 in the truck at location A

- PR: \( \text{Load1A} = (\{\text{Tr@A, P1@A}\}, \{\text{P1-in-Truck, } \neg \text{P1@A}\}) \)
- FDR: \( \text{Load1A} = (\{\text{TruckLocation = A, Package1Location = A}\}, \{\text{Package1Location = T}\}) \)
Valid Plan

Let \( \pi = (S, A, s_i, s_G) \) be a planning problem. A plan \( P = [a_1, a_2, \ldots, a_n] \) is a valid plan for \( \pi \) if:

- \( \forall i : a_i \in A \)
- let \( s_1 = s_i \) and \( s_{i+1} = apply(s_i, a_i) \)
- \( \forall i : a_i \) is applicable in \( s_i \), i.e., \( pre(a_i) \subseteq s_i \)
- \( s_G \subseteq s_n \)

where \( s' = apply(s, a) \) is the world state after applying action \( a \) in state \( s \). The state \( s' \) is identical to \( s \) except for changes implied by the effects of \( a \).

Organization
Tomáš Balyo, Dominik Schreiber – Planning and Scheduling
October 18, 2018 25/34
Representation of all possible Actions

- List all the actions – Explicit Representation
  - Can be a huge amount, but most of the time it’s fine
  - Requires some script to generate the list, cannot be done by hand
  - Usually used with FDR

- Operators – Action Templates – Implicit Representation
  - Using objects, types and predicates define Action Templates

Example: operator for loading a package at some location

Types: location, package

Objects: P1, P2 – package, A, B, C – location

Predicates: at(P – package, L – location), TruckAt(L – location), InTruck(P – package)

Operator:

loadPackage(P – package, L – location) = ({at(P, L), TruckAt(L)}, {¬at(P, L), InTruck(P)})
Actions from Operators

Types: location, package
Objects: P1, P2 - package, A,B,C - location
Predicates: PackageAt(P - package, L - location), TruckAt(L - location), InTruck(P - package)
Operator:
loadPackage(P - package, L - location) =

\[
\begin{array}{l}
\{\text{PackageAt}(P,L), \text{TruckAt}(L)\}, \{-\text{PackageAt}(P,L), \text{InTruck}(P)\}
\end{array}
\]

The operator and objects generate these actions:

\begin{align*}
\text{loadPackage}(P1,A), \text{loadPackage}(P1,B), \text{loadPackage}(P1,C), \\
\text{loadPackage}(P2,A), \text{loadPackage}(P2,B), \text{loadPackage}(P2,C),
\end{align*}
(define (domain trucking)
  (:requirements :strips :typing)
  (:types
    location - object
    package - object
  )
  (:constants )
  (:predicates
    (road ?l1 - location ?l2 - location)
    (truck-at ?l - location)
    (package-at ?p - package ?l - location)
    (in-truck ?p - package)
  )
)
(:action loadPackage
   :parameters (?l - location ?p - package)
   :precondition (and
      (package-at ?p ?l)
      (truck-at ?l)
   )
   :effect(and
      (not (package-at ?p ?l))
      (in-truck ?p)
   )
)
(:action unloadPackage
  :parameters (?l - location ?p - package)
  :precondition (and
    (in-truck ?p)
    (truck-at ?l)
  )
  :effect(and
    (not (in-truck ?p))
    (package-at ?p ?l)
  )
)
(:action drive
  :parameters (?l1 ?l2 - location)
  :precondition (and
    (road ?l1 ?l2)
    (truck-at ?l1)
  )
  :effect(and
    (not (truck-at ?l1))
    (truck-at ?l2)
  )
)
(define (problem trucking-3-cities-2-packages)
  (:domain trucking)
  (:requirements :strips :typing)
  (:objects
   l1 l2 l3 - location
   p1 p2 - package
  )
)
(:init
 (road l1 l2)
 (road l2 l3)
 (truck-at l1)
 (package-at p1 l1)
 (package-at p2 l2)
 )
 (:goal (and
 (package-at p1 l3)
 (package-at p2 l3)
 )))
The End

Next Week: Complexity of Planning