Planning for Artificial Intelligence

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Lukáš Chrpa and Stefan Edelkamp

AI CENTER

Before we start ….

- Lectures and Tutorials will be conducted in person
  - very likely for the whole semester
- Two lecturers
  - Lukáš Chrpa (first 7 lectures)
  - Stefan Edelkamp (last 6 lectures)
- Two tutors
  - Michaela Urbanovská (first 10 weeks)
  - Jan Mrkos (last 4 weeks)

Before we start ….

- Assignment (zápočet)
  - Two courseworks (classical planning and probabilistic planning)
  - Get at least **25** out of **50** points
- Exam
  - Written exam (onsite if possible)
  - Get at least **25** out of **50** points

Before we start ….

- Course website
  - https://cw.fel.cvut.cz/b212/courses/be4m36pui/start
- Course forum
  - https://cw.felk.cvut.cz/forum/forum-1778.html
- **Don’t hesitate to contact us if you need anything**

What is AI ?

- **“The science concerned with understanding intelligent behavior by attempting to create it in artificial”** (T. Smithers)

- **Intelligent behavior** can be considered as an ability to **solve problems** on which the **machine has no knowledge of a suitable algorithm**

What is Automated Planning ?

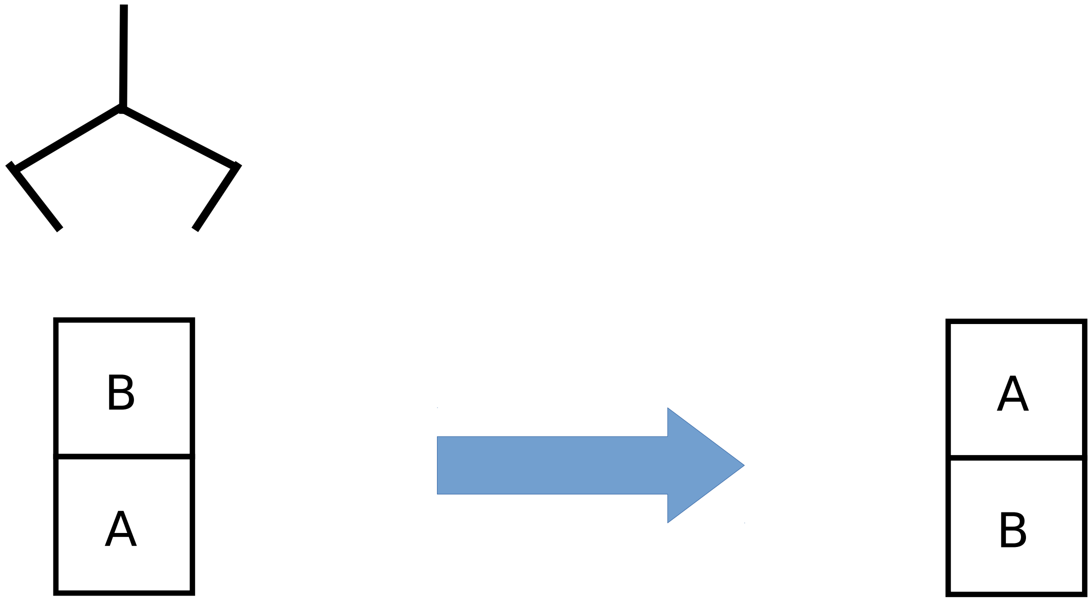
“Planning is reasoning about acting”
[Ghallab, Nau, Traverso]

An actor finds and executes a sequence of actions in order to achieve its goals

What is Automated Planning?

- Artificial Intelligence (sub-field)
  - (general) problem solving
- Decision Theory meets Computer Science
  - sequential decision making
  - various forms of combinatorial optimization problems
- Three approaches in AI to the problems of action selection or control
  - Learning: learn control from experience
  - Programming: specify control by hand
  - **Planning:** specify problem by hand, derive control automatically

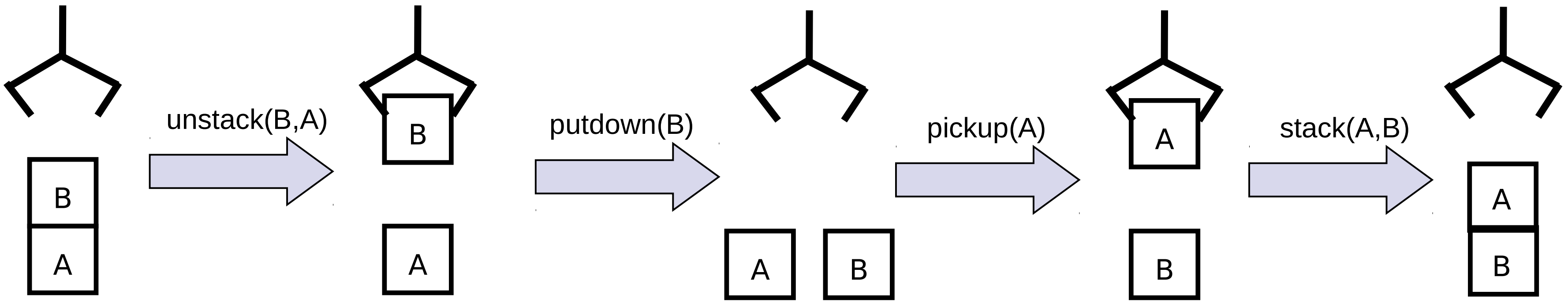
BlocksWorld Example



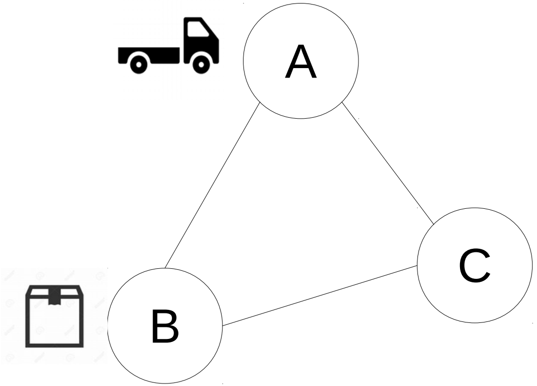
Initial state

Goal

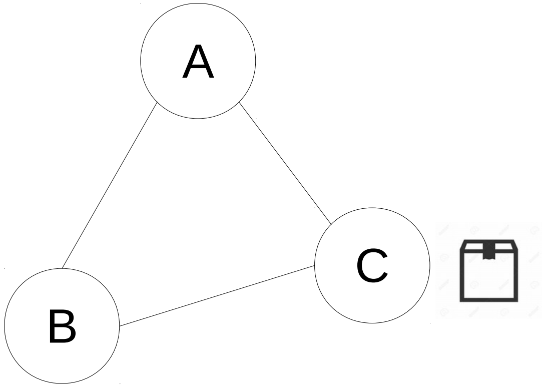
BlocksWorld Example



Logistics Example

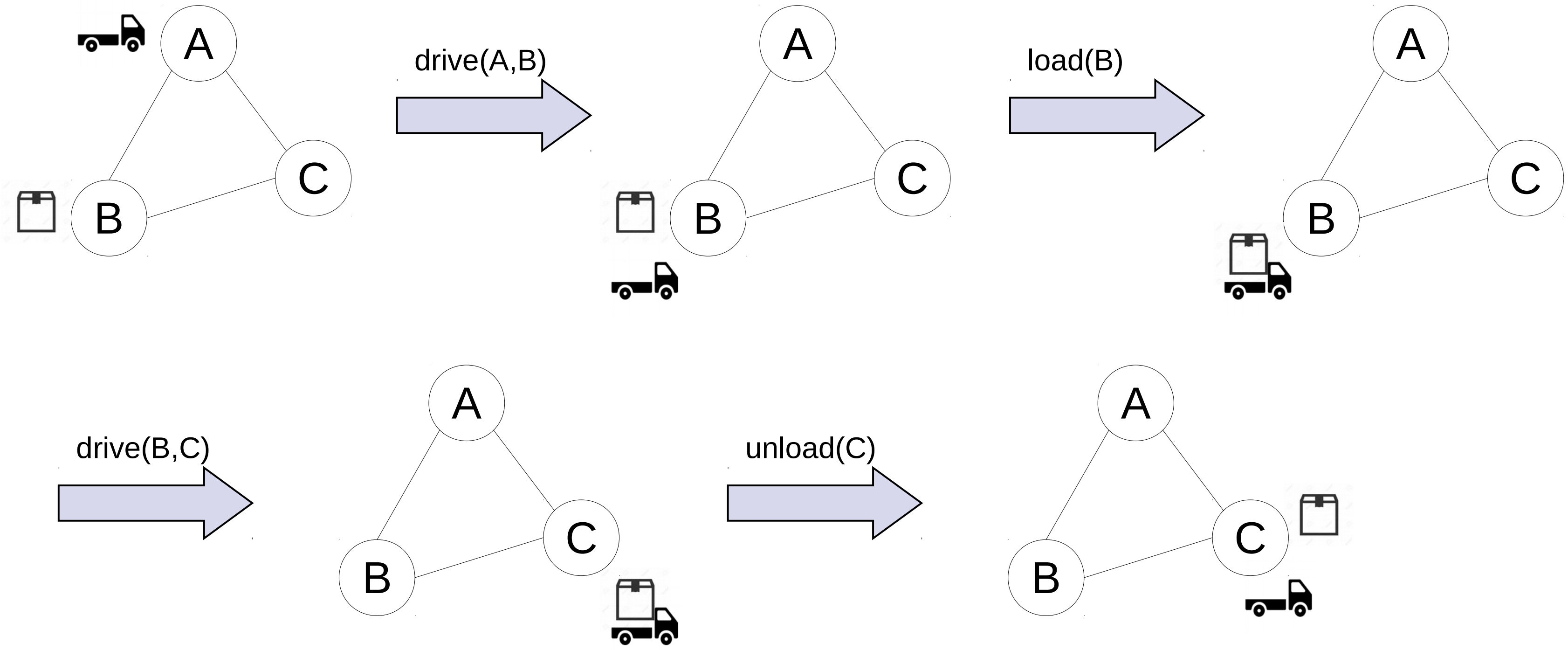


Initial state

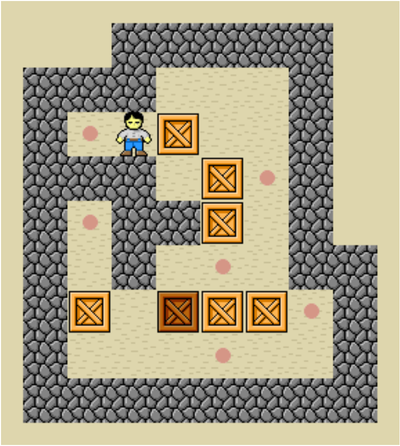


Goal

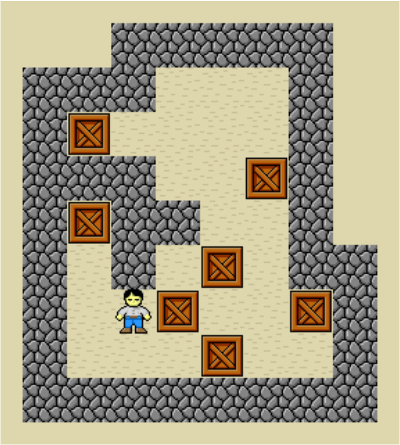
Logistics Example



Sokoban Example

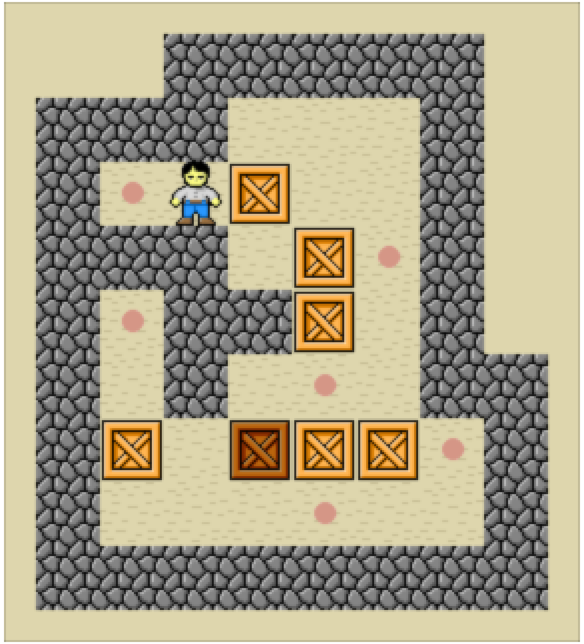


Initial state



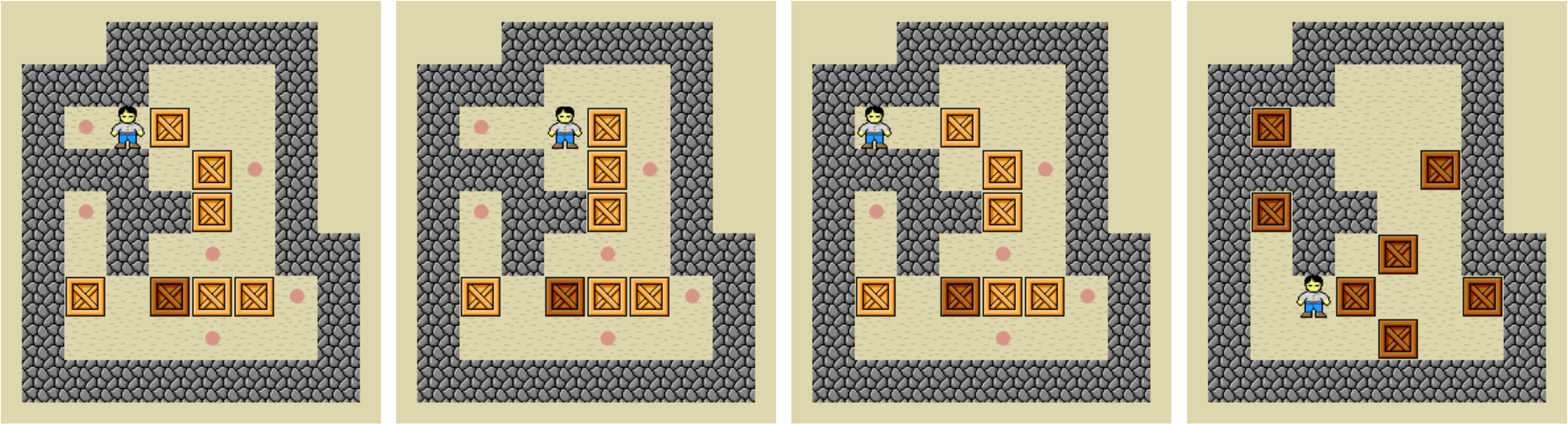
Goal

Sokoban Example



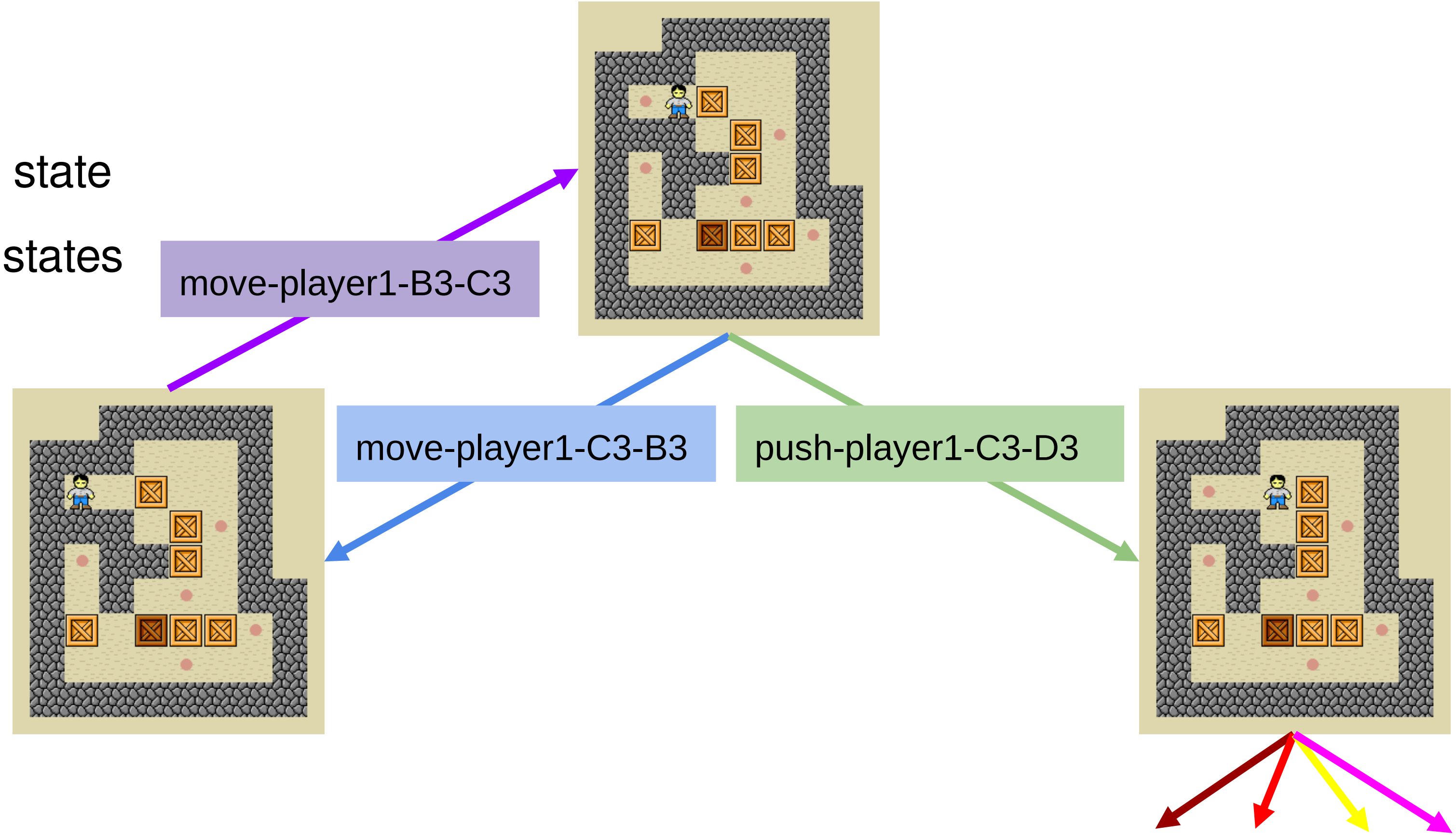
Classical Planning Elements

- **States**
  - Initial state
  - Goal states
- Actions



Classical Planning Elements

- States
  - Initial state
  - Goal states
- **Actions**

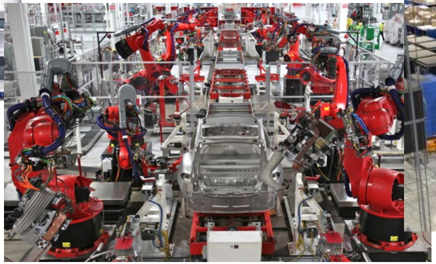


Real-World Environment is not that simple ...

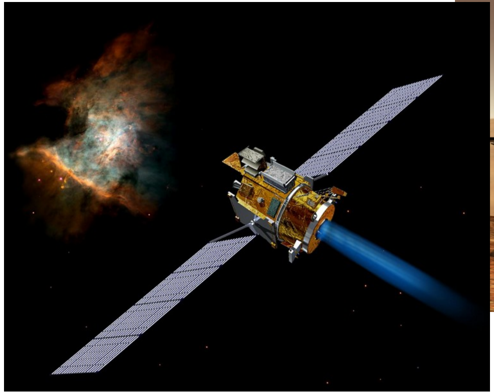
- Static vs Dynamic Environment
  - Environment cannot/can change without actor's consent
- Full vs Partial Observability
- Deterministic vs Non-deterministic action effects
- Discretized vs Continuous environment representation
- Instantaneous vs Durative and/or continuous action effects

- Classical Planning (~7 lectures)
- Temporal Planning (1 lecture)
- Planning under uncertainty (~4 lectures)

Domains



puzzles; computer, board, card games;
production planning and logistics;
humanitarian and military missions;
various-scale robotics; space missions



Task Planning for AUVs [Chrpa et al., 2015]

- Necessity to control multiple heterogeneous Autonomous Underwater Vehicles (AUVs)
- An operator (human) specifies high-level tasks (e.g. “sample an object with ctd camera”)
- Task assignment to each AUV should be automatized

How task assignment can be automatized ?

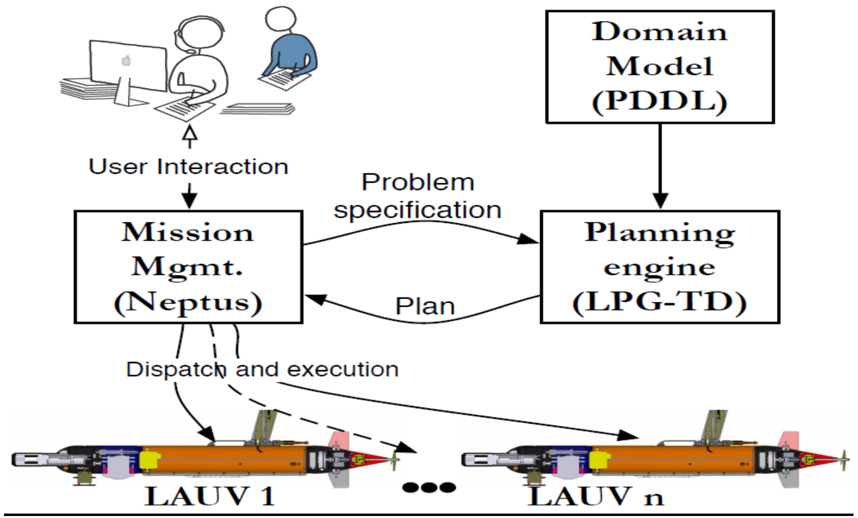
- Each **task** has specific **requirements**
- Each **vehicle** has specific **capabilities**
- For completing tasks AUVs have to perform certain sequences of **actions**
- Hence, we need to find **a plan** that if executed, the AUVs will complete all given tasks

Available “Machinery”

- In LSTS, AUVs are controlled via **NEPTUS** (a decision support tool with GUI) and **DUNE** (onboard vehicle control) → **“low-level” control**
- Domain-independent **AI planning** (i.e., finding a sequence of actions that achieves a defined goal) → **“high-level” task planning**
  - **PDDL**, a language for specifying planning domain models and problem instances
  - **LPG-td**, a planning engine accepting domain and problem descriptions in PDDL and returning a plan (if exists)

Integrating Planning and Control

- **User specifies tasks** in NEPTUS
- NEPTUS **generates a planning problem** and sends it to LPG-td
- LPG-td **returns a plan** to NEPTUS
- NEPTUS **distributes the plan** to each of the vehicles



Why is Automated Planning useful ?

- NASA's vision of space-exploratory systems
  - **Low cost** control, low cost and rapid development
  - **Long-term** autonomous operations
  - Operations **must guarantee success** (under resource and time constraints)
- **Deep Space One** (1998) and the **Mars Rover** mission (2004) are some of the successes
- Other successes: **Aircraft manufacturing** (Boeing), **Crisis management** (Schlumberger) and others !

Domain-independent and Domain-specific

Domain-independent:

- fundamental
- flexible
- reusable

Domain-specific:

- rigid
- efficient
- specialized

Domain-independent and Domain-specific

Domain-independent:

- **fundamental**
- flexible
- reusable

Domain-specific:

- rigid
- efficient
- specialized

Until we get the fundamental principles ...

Domain-independent and Domain-specific

Domain-independent:

- fundamental
- **flexible**
- **reusable**

Domain-specific:

- rigid
- efficient
- specialized

… we cannot be flexible and we cannot reuse ...

Domain-independent and Domain-specific

Domain-independent:

- fundamental
- flexible
- reusable

Domain-specific:

- **rigid**
- **efficient**
- specialized

… we cannot optimize or ...

Domain-independent and Domain-specific

Domain-independent:

- fundamental
- flexible
- reusable

Domain-specific:

- rigid
- efficient
- **specialized**

… specialize.

Let’s plan!

- **Models** for defining, classifying, and understanding problems
  - what is a planning problem
  - what is a solution (plan), and
  - what is an optimal solution
- **Languages** for representing problems (e.g. PDDL)
- **Algorithms** for solving them
- **Executing** the plans

Possible MSc Thesis Topics (not an exhaustive list)

- Modeling and Reformulation in non-classical planning
  - Numerical and Temporal Planning
  - Non-deterministic Planning
  - Continuous Planning
- Learning Domain Control Knowledge
- Reasoning with Agent Planning Programs
- Planning and acting in dynamic environments
- Contact Lukas Chrpa (chrpaluk@fel.cvut.cz) or Stefan Edelkamp (stefan.edelkamp@gmail.com) if interested

Classical Planning

State Model for Classical Planning

- Let **S** be a set of **states**
- Let **A** be a set of **actions**
- Let $\boldsymbol{\gamma}$:S×A→S be a **transition function**
- Let $\boldsymbol{\gamma^*}$:S×A*→S be a **generalized transition function**
- Let $\mathbf{s_I}$∈S be an **initial state**
- Let $\mathbf{S_G}$⊆S be a set of **goal states**

- A sequence of actions π is a **solution plan** iff $\gamma^*(s_I,\pi) \in S_G$

Transition systems

- A **transition system** is a 5-tuple $\mathcal{T}$=(S,L,T,I,G), where
  - **S** is a finite set of **states**
  - **L** is a finite set of **labels**
  - **T**$\subseteq$S$\times$L$\times$S is a **transition relation**
  - **I**$\subseteq$S is a set of **initial states**
  - **G**$\subseteq$S is a set of **goal states**
- We say that $\mathcal{T}$ **has a transition** (s,l,s') iff (s,l,s')$\in$T

Transition systems for Classical Planning State Model

- Sets of **states** correspond to each other
- A set of **labels** correspond to the set of **actions**
- A transition system has a transition (s,a,s') iff $\gamma(s,a)=s'$
- There is a **single initial state**
- Sets of **goal states** correspond to each other

Planning in Transition Systems

- A transition system is a **directed graph**
- To **solve** a planning problem, one has to find a **path** from an initial state to any of the goal states
- **Dijkstra's** algorithm can do the job in $\mathcal{O}(|S|\log(|S|)+|T|)$
- So are we done here ??

Planning in Transition Systems

- A transition system is a **directed graph**
- To **solve** a planning problem, one has to find a **path** from an initial state to any of the goal states
- **Dijkstra's** algorithm can do the job in $\mathcal{O}(|S|\log(|S|)+|T|)$
- So are we done here ??

- Not really …..

Let’s count

- Blocksworld (simplified)
  - A block can be on table or stacked on another block
  - Let $g(n,k)$ be a function, where n and k stand for ungrounded and grouded towers respectively, $g(0,k)=1$ and $g(n+1,k)=g(n,k+1)+(n+k)g(n,k)$
  - Then, $|S|=g(n,0)$ (e.g. for $n=30$, $|S|\sim 2*10^{35}$)
- Logistics
  - Each truck can be at some location, each package can be at some location or in some truck
  - $|S|=l^t*(l+t)^{p}$, where t,l,p is the number of trucks, locations and packages respectively
  - With $t=10$, $l=100$, $p=100$, we get $|S|>10^{200}$

What now ?

- Such large state spaces cannot be enumerated
- Yet solving such problems is not hopeless !

- We need compact representation !

How to represent a state of the environment in Classical Planning

- By **propositions**
  - e.g. on-A-B, at-truck-A, in-package-truck
  - A **state** is a set of **propositions** such that a proposition **belonging** to a state is considered as being **true** while a proposition **not belonging** to a state is considered as being **false**
- By **state variables**
  - e.g. on-A=B, at-truck=A, loc-package=truck
  - A **state** is a set of **assignments of all variables**

STRIPS Planning Task

- A **planning task** in **STRIPS** is a quadruple (P,A,I,G), where
  - **P** is a finite set of **atoms** (or facts or propositions)
  - **A** is a finite set of **actions**, where each action $a \in A$ is a triple (pre(a),del(a),add(a)), all subsets of P, where
    - pre(a) is a **precondition** of a
    - del(a) is a set of **delete effects** of a
    - add(a) is a set of **add effects** of a
  - $\mathbf{I} \subseteq P$ is an **initial state**
  - $\mathbf{G} \subseteq P$ is a **goal**

STRIPS Planning Task cont.

- **States** are **collections of atoms**, i.e., $S \subseteq 2^P$
- An action a is **applicable** in a state s iff **pre(a)⊆s**
  - (otherwise a is **inapplicable** in s)
- A state s’ is the **result** of application of an applicable action a in a state s iff **s’=(s\del(a))**∪**add(a)**

SAS Planning Task

- A **planning task** in **SAS** is quadruple (V,A,I,G), where
  - **V** is a set of **variables**, where each variable v∈V has its own domain dom(v)
  - **A** is a set of **actions**, where each action a∈A is a pair (pre(a),eff(a)), both partial assignments over V, where
    - pre(a) is a **precondition** of a
    - eff(a) stands for **effects** of a
  - **I** is an **initial state** (a complete assignment over V)
  - **G** is a **goal** (partial assignment over V)

SAS Planning Task cont.

- Let q[v] denote the value of a variable v in a (partial) assignment q
- **States** are complete assignments over V
- An action a is **applicable** in a state s iff **pre(a)[v]=s[v]** whenever pre(a)[v] is specified
  - (otherwise a is **inapplicable** in s)
- A state s' is the **result** of application of an applicable action a in a state s iff **s'[v]=eff(a)[v]** whenever eff(a)[v] is specified or **s'[v]=s[v]** otherwise

Solution Plans

- Let $\gamma(s,a)=s'$ iff s' is the result of application of an action a in a state s (a is applicable in s)
  - $\gamma(s,a)$ is undefined iff a is inapplicable in s
- Let γ^* be defined recursively
  - $\gamma^*(s,\langle\rangle)=s$
  - $\gamma^*(s,\langle a_1,a_2,\ldots,a_n\rangle)=\gamma^*(\gamma(s,a_1),\langle a_2,\ldots,a_n\rangle)$
- We say that π, a sequence of actions over A, is a **solution plan** (or a **plan**) of the planning task iff $\gamma^*(I,\pi) \models G$ ($G\subseteq\gamma^*(I,\pi)$ for STRIPS)