PlanViz

AI Planning Graph Visualization

von Elia Pfletschinger, Florian Geier, Paul Nachtigall

Motivation

- Wunsch nach Planner für **beliebiges Problem**
  - Deterministische Automatische Planner
  - Heuristische Planner
  - Große Varianz
- **Tradeoff**: Allgemeinheit vs. Inhärentes Verständnis
- Wie versteht man ein beliebiges Problem?

Struktur erkennen!

Theorie: *Womit arbeiten wir?*

Definition Planning-Problem (MPT): $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{A}, \mathcal{O} \rangle$

- V: **Variablen** mit Domäne D_V
- s_0: **Startzustand** (vollst. über O)
- $s_\star$: Partielle **Zielzuweisung**
- A: **Axiome** (hier nicht relevant)
- O: **Operationen** (name, pre: 2^P, eff: 2^P)

Theorie: *Wichtige Strukturen*

Causal Graph

Domain Transitions Graph

Landmark Graph

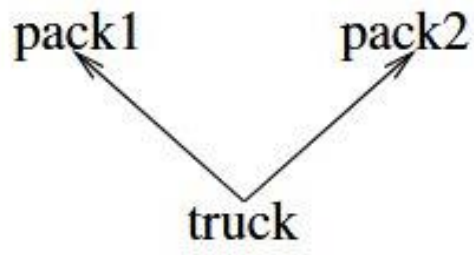
Ziel: *Darstellung von CG, DTG, LMG*

Theorie: *Causal Graph*

Graph CG:

$$\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{A}, \mathcal{O} \rangle$$

- ***V*** = Variablen
- (v, w) in ***E*** wenn:
  - Existiert Aktion a:
  - $pre_a(v)$ und $eff_a(w)$ definiert
  - Oder $eff_a(v)$ und $eff_a(w)$ definiert
- Also Kante zwischen Variablen, wenn
  - Eine **Änderung** von ***w*** setzt **Zustand** von ***v*** voraus
  - **Änderung** von ***w*** ändert auch ***v***



= Abhängigkeiten von Variablen

Theorie: *Domain Transition Graph einer Variable*

Graph DTG(v):

$$\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{A}, \mathcal{O} \rangle$$

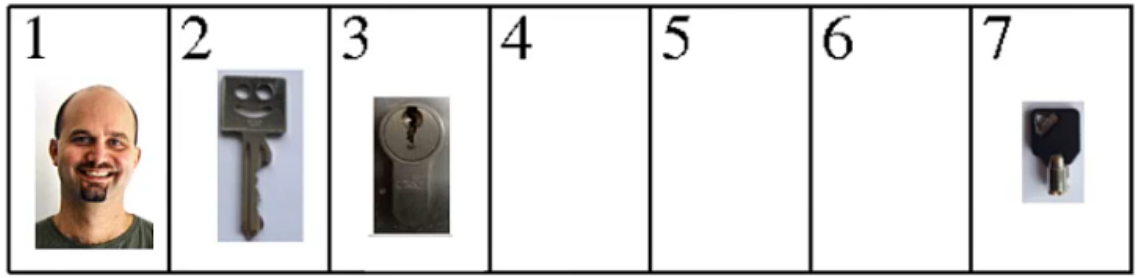
- ***V*** = Zustände einer Variablen D_V
- (d, d') in ***E*** wenn:
  - Existiert Aktion a:
  - $pre_a(v) = d$ *oder undefiniert* und $eff_a(v) = d'$
- Beschriftung: Aktion, pre, eff
- Also Kante zwischen Variablen, wenn
  - Man von Zustand ***d*** zu Zustand ***d'*** übergehen kann

= Übergänge zwischen Zuständen

Theorie: *Landmarks*

- Fact Landmark:
  - Zustand, der in jedem Plan vorkommen muss
- Action Landmark:
  - Aktion, die in jedem Plan vorkommen muss
- Reihenfolge?

Problem: Bring small key to position 1.



- `agent-at-2, agent-at-3, …`
- `big-key-held`
- `big-key-at-2`
- `lock-open`
- `small-key-held`

Theorie: *Landmark Graph*

Graph LG:

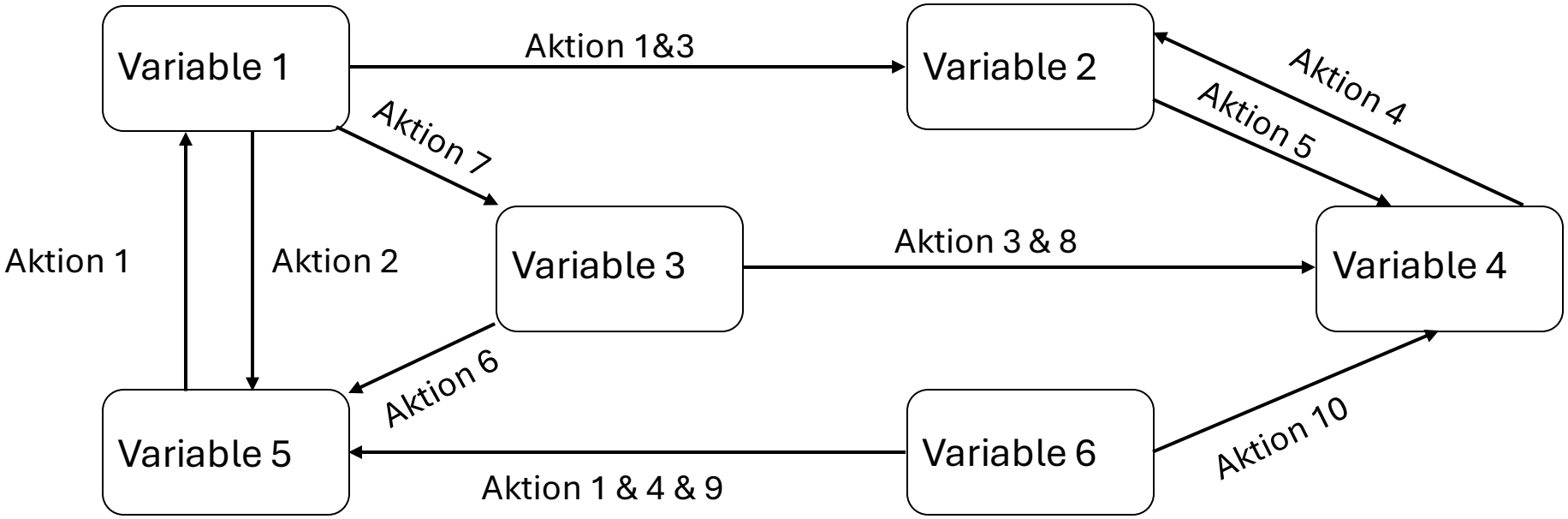
- ***V*** = Landmarks D_V
- (l, l') in ***E*** wenn:
  - ***l*** muss vor ***l'*** erreicht werden

$$\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{A}, \mathcal{O} \rangle$$

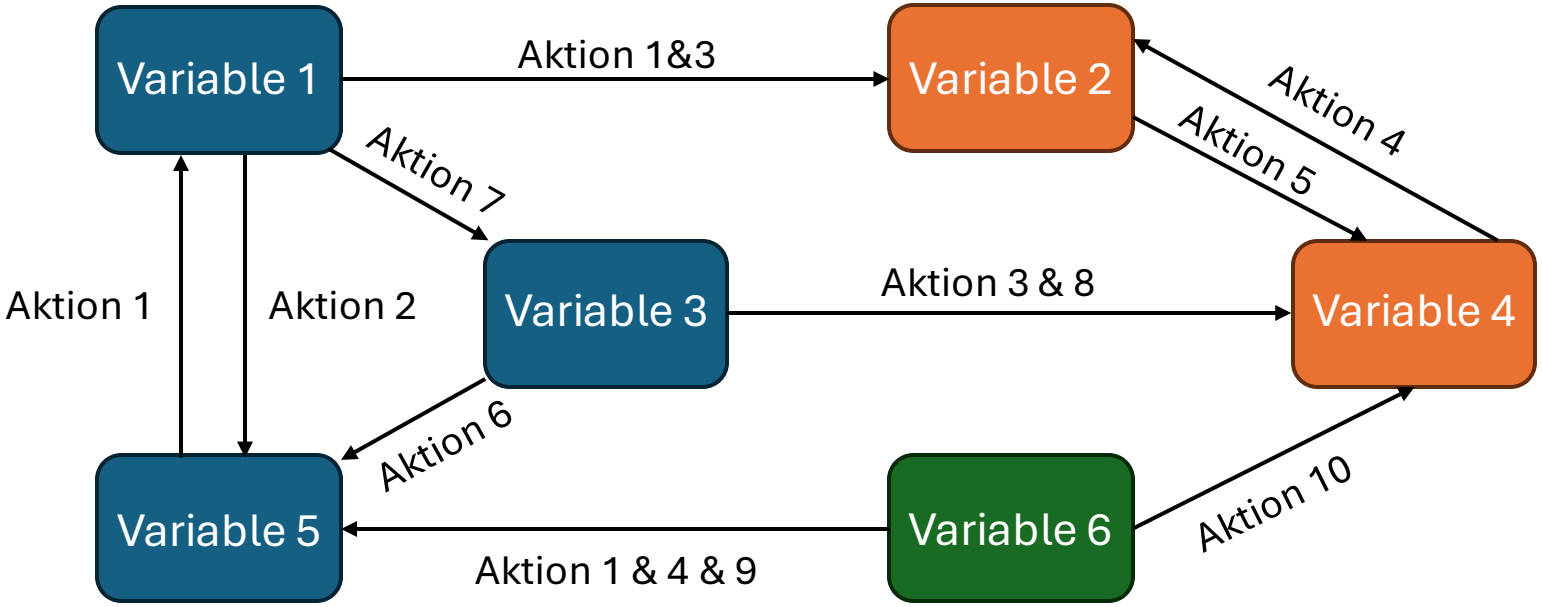
= notwendige Teilfolge jedes Plans

=> Pruning (Aussortieren)

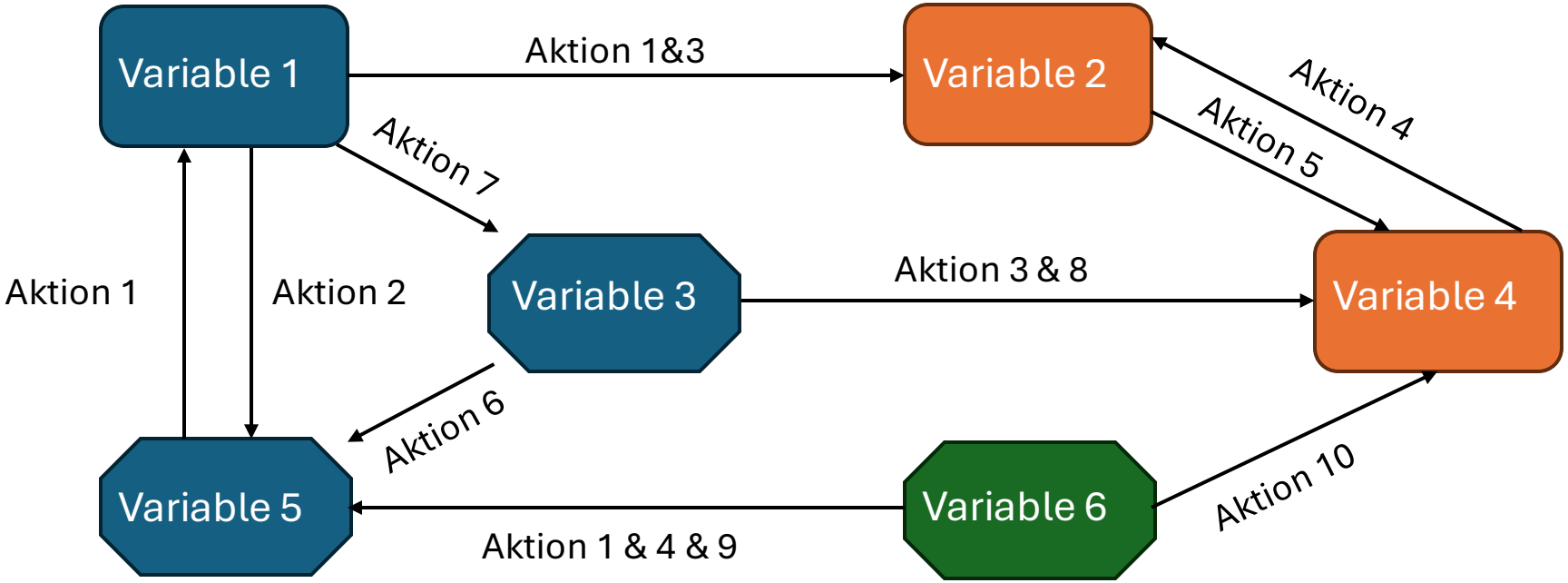
Beispiel: Causal Graph



Beispiel: Causal Graph

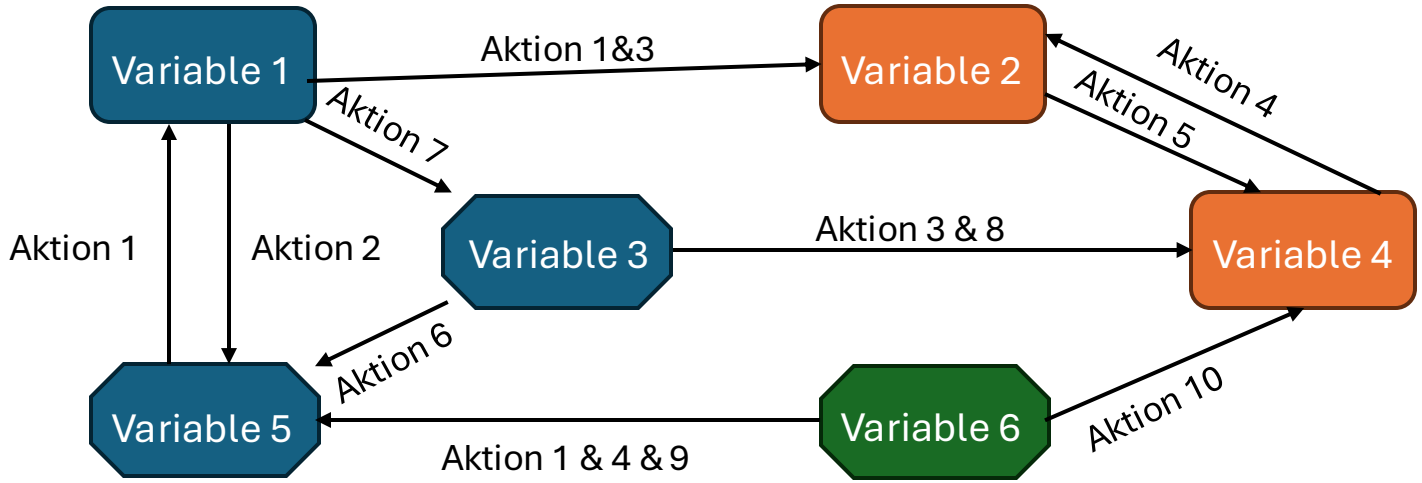


Beispiel: Causal Graph



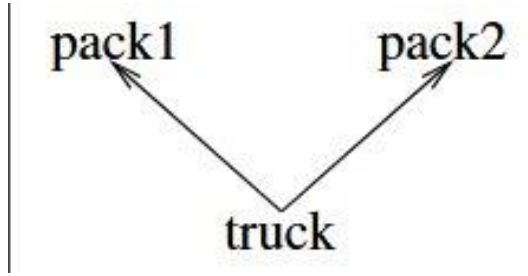
Beispiel: Causal Graph

- Aktionen: Übergänge
- Variablen: Knoten
- Zielvariablen: Achtecke
- Zusammenhangskomponenten: farbig

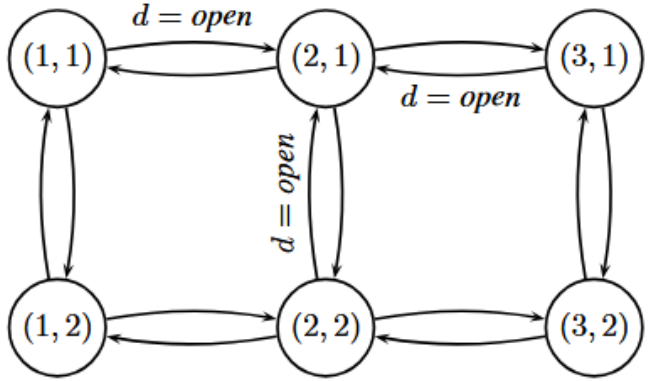


Zusammenfassung: *Wichtige Strukturen*

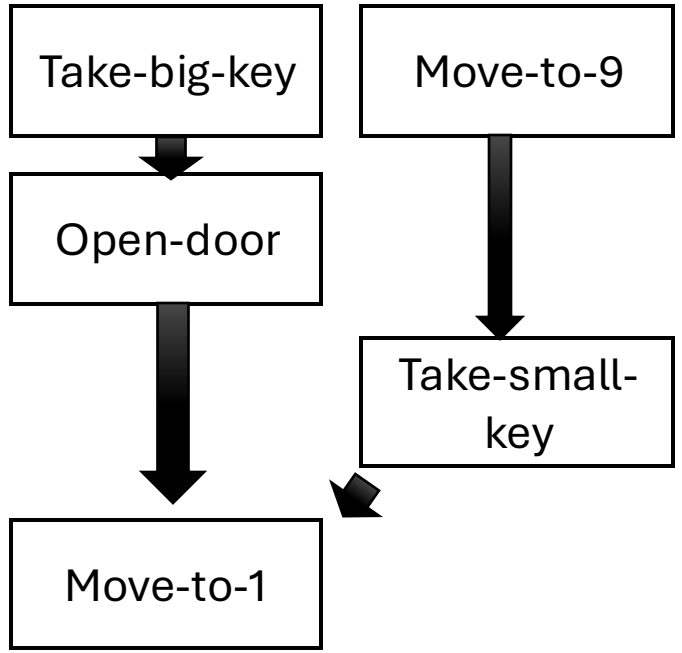
Causal Graph



Domain Transition Graph

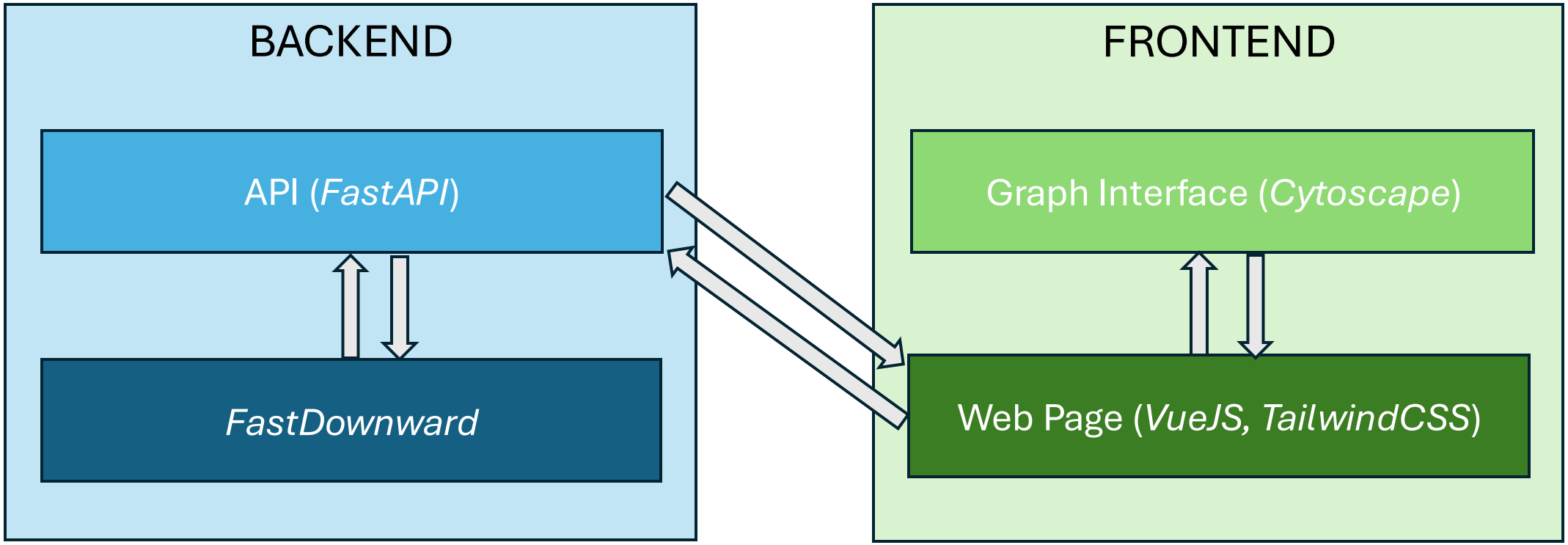


Landmark Graph



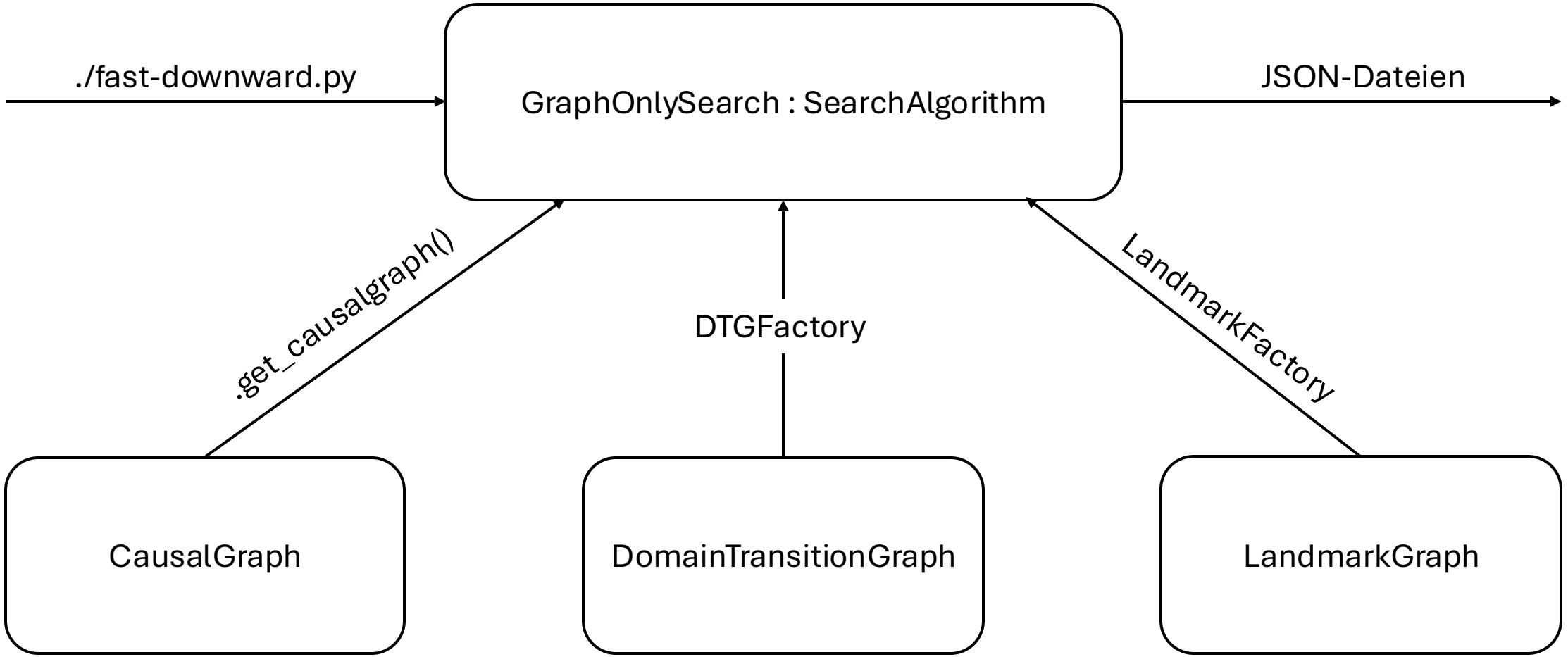
=> Struktur des Problems

Project Plan

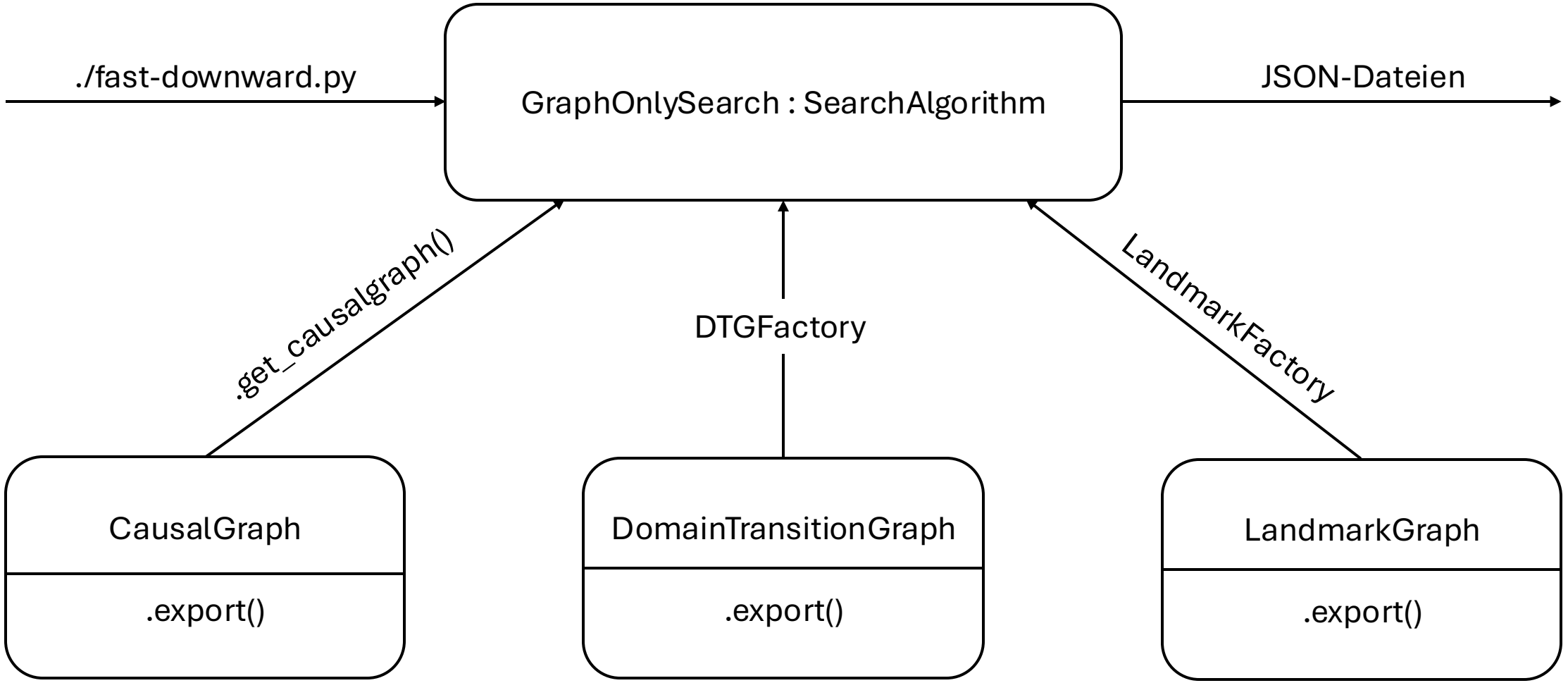


Fast Downward Modifikation

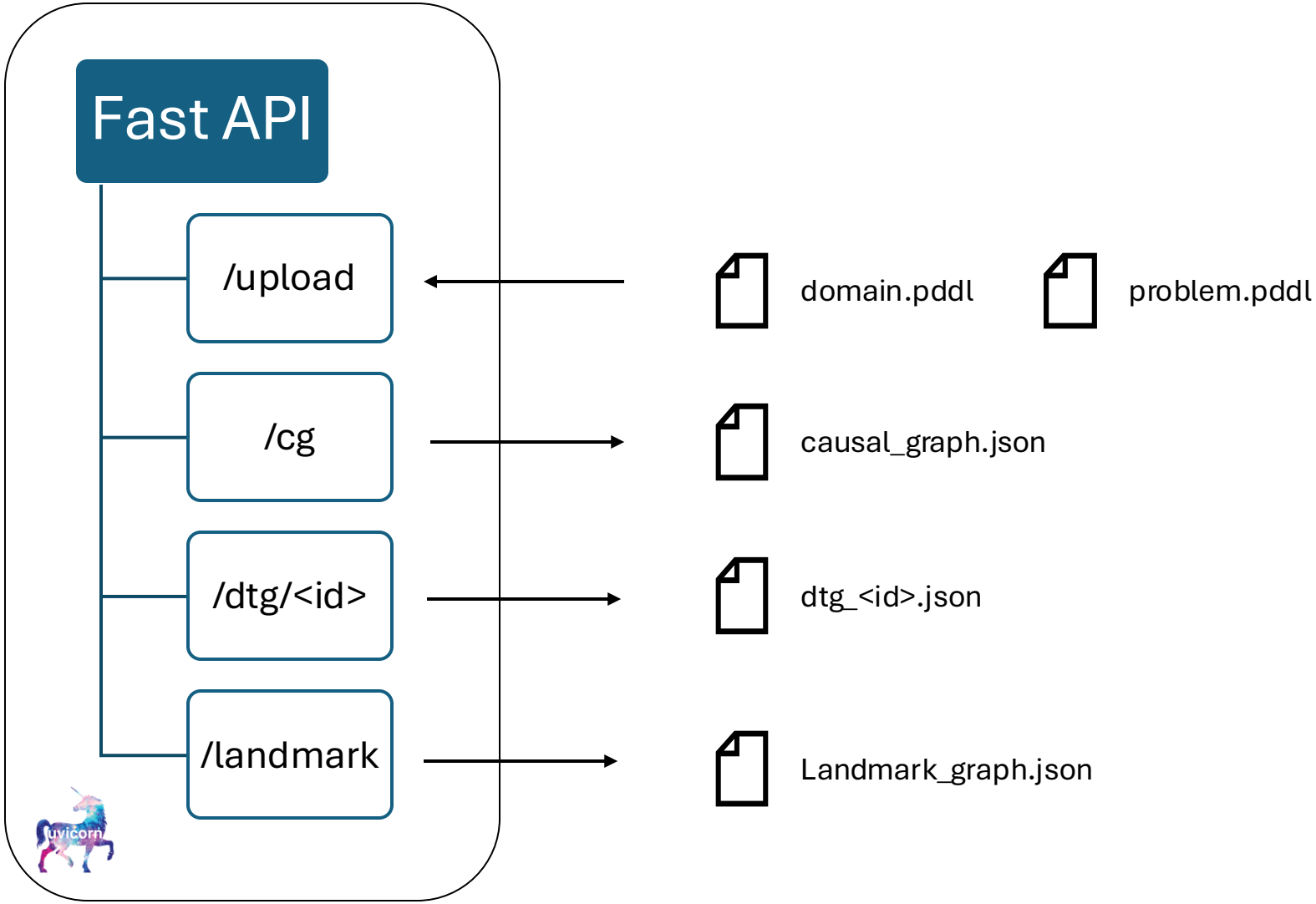
Fast Downward Modifikation



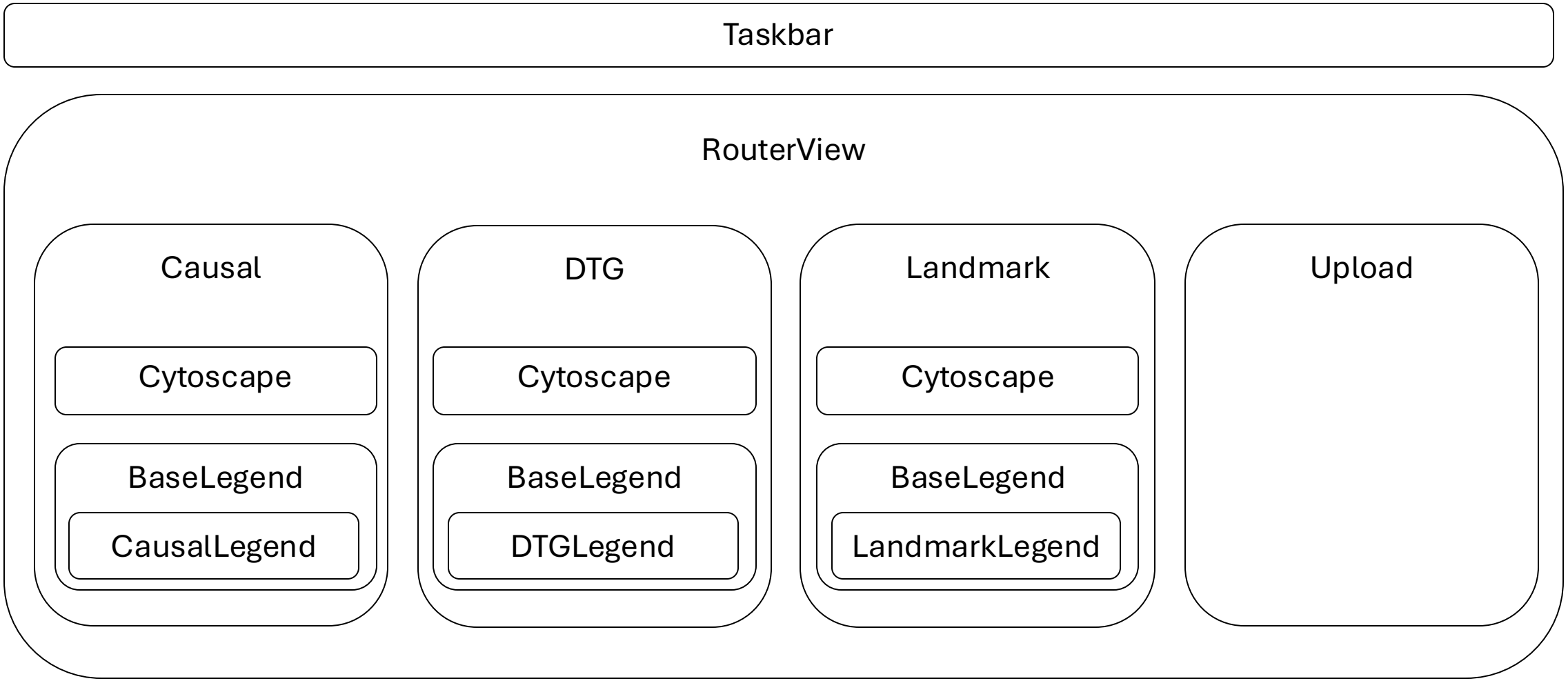
Fast Downward Modifikation



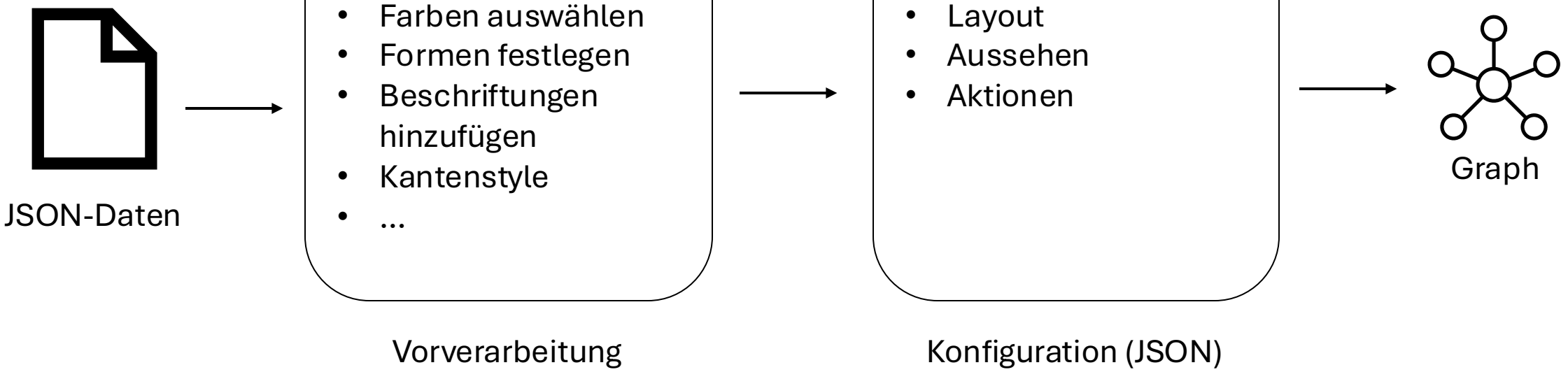
Fast API


Fast API
/upload
/cg
/dtg/<id>
/landmark
uvicorn
domain.pddl
problem.pddl
causal_graph.json
dtg_<id>.json
Landmark_graph.json

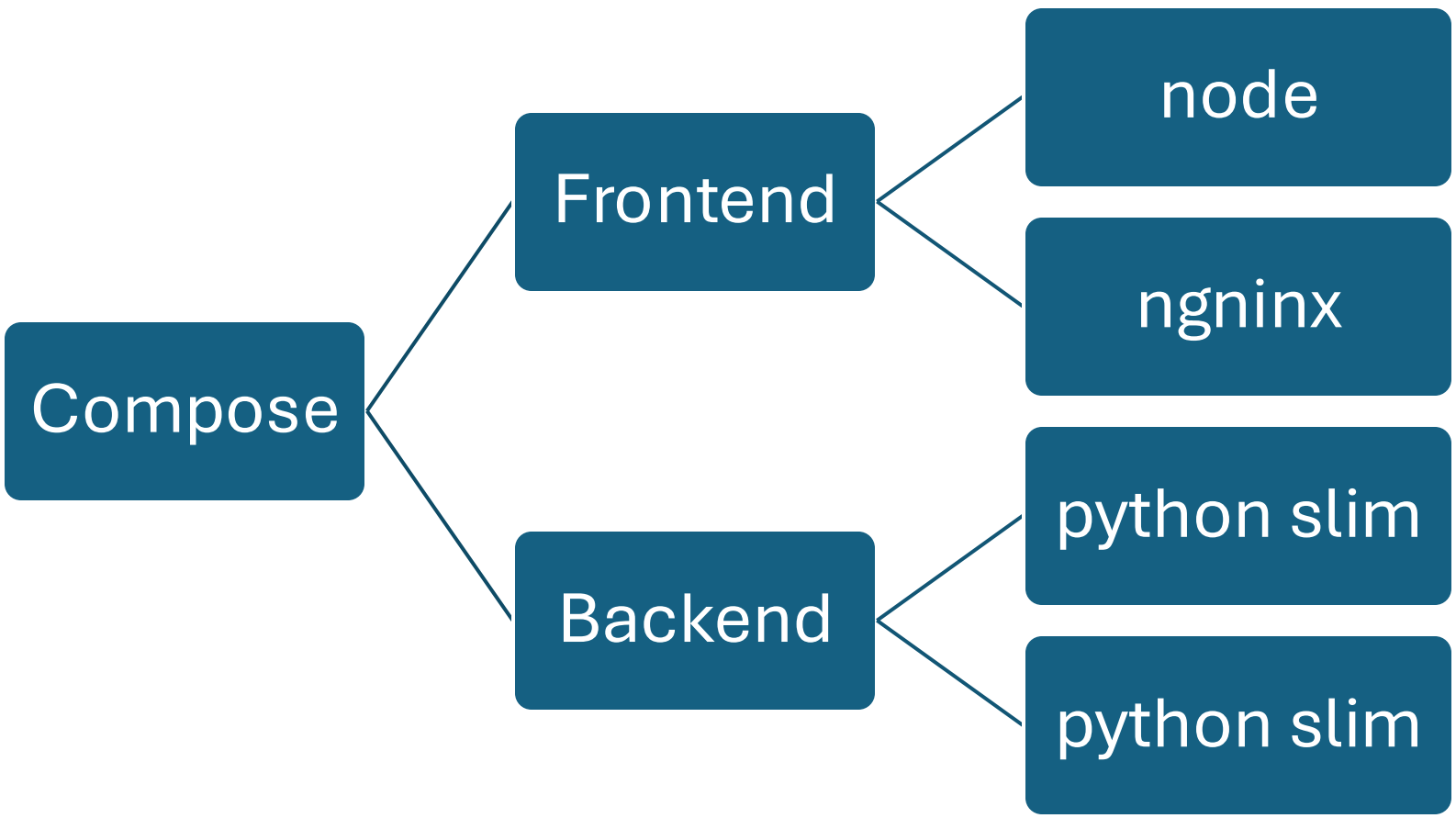
Frontend (Vue.js + Tailwindcss)



Cytoscape


JSON-Daten
JS-Code
• Farben auswählen
• Formen festlegen
• Beschriftungen hinzufügen
• Kantenstyle
• ...
Cytoscape JS
• Layout
• Aussehen
• Aktionen
Graph
Vorverarbeitung
Konfiguration (JSON)

Docker


Compose
Frontend
Backend
node
ngninx
python slim
python slim

Ergebnisse/Demo

Lessons Learned

- Git (Development branch)
- Kommunikation
- Vue.js
- Docker
- Arbeiten mit größerem C++ Projekt

Fazit + Ausblick

- Fazit
  - Ziel erreicht
  - Projekt Plan umgesetzt
- Ausblick
  - Backend um Sessions erweitern
  - Verschiedene Graphen-Layouts
  - Docker Images
  - Darkmode

Fragen