FACULTY OF NATURAL, MATHEMATICAL & ENGINEERING SCIENCES

DEPARTMENT OF INFORMATICS

KING'S College LONDON

Formalising Mathematics to Obtain Verified AI and Optimisation Software

Thomas Ammer   David Wang   (supervised by Mohammad Abdulaziz)

A Motivation: Software and Hardware Failures

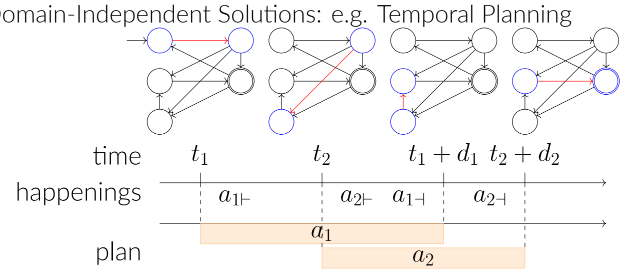
- Post Office Scandal(SF, > 900 postmasters wrongly convicted)
- Pentium FDIV (HF, flawed floating-point unit in processor)
- Therac-25 (SF, excessively high doses in radiation therapy)
- Boeing 737 (SF, screen blackouts when approaching specific runways, flight control)

Interactive Theorem Proving (ITP) and Isabelle

- Interactive Theorem Proving = Writing (partial) proofs s.t. they can be completed and checked by software
- Interactive Theorem Prover/Proof Assistant = Software to complete and check proofs
- We use Isabelle/HOL (others Coq/Rocq, Lean, etc.)
- Trustworthy Software
- In-depth understanding/explanation of behaviour

AI Planning

- See: https://formplan.github.io/
- Abstract Problem Representation: (1) Logic: e.g.
  s ⊨ ∀r::robot. ∀o::object.
  weight(o) ≤ capacity(r) ∧ ¬broken(r) ∧ ... ⟶ can_carry(r,o)
  (2) Probabilities: e.g. Markov Decision Processes
- Domain-Independent Solutions: e.g. Temporal Planning

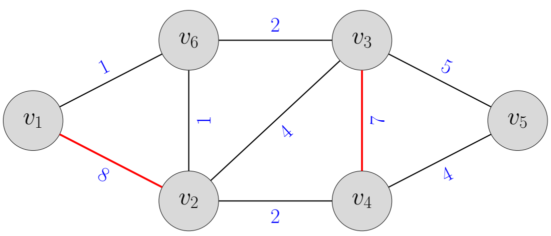


Find a schedule of concurrent actions that transform the initial state until a goal is satisfied. E.g. s ⊨ ∀x::task. completed(x)

- Isabelle formalisation of a part of the Planning Domain Definition Language (PDDL) and its semantics based on abstract timed transition systems

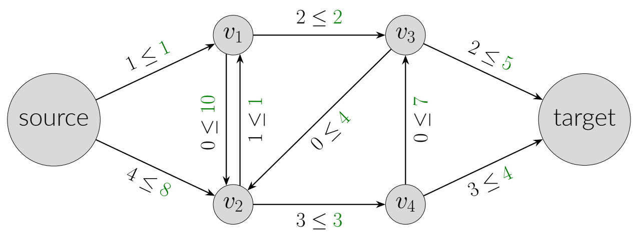
Examples for Combinatorial Optimisation (CO)

- Matching (certain one-to-one pairings allowed, selection from these, e.g. by Edmonds' Blossom Algorithm):



Combinatorial Optimisation (continued)

- Flows (send liquid from source to target while respecting connection capacities, e.g. by Ford-Fulkerson method):



Verifying Algorithms

- Planning and CO problems can be solved by algorithms
- Isabelle/HOL formalisation of mathematical theory to verify these algorithms
- Library for graph algorithms (together with collaborators): flows, matchings, matroids, spanning trees etc.
- Running Time: Time waiting for solution in relation to size of problem description
- Correctness guarantee for result if using verified code

Applications of Planning and CO Problems

Matching:

- Kidney Exchange in UK and Scandinavia [2, 9]
- Student-College Matching with preferences and fairness [1]
- Auctions and Market Design, e.g. Google's Adwords [18]

Flows:

- Electricity Grid Congestion, Pipe Network Analysis, and other transportation subject to capacities and costs
- Circuit and other Hardware Design. Esp. Very Large Scale Integration (VLSI), e.g. decrease of chip area by 23% [10], reduction of wire length by 10% [12], minimise cell movement [5], and others [6, 8, 14])

Planning:

- Logistics [21] (minimising costs of moving vessels), Enterprise Risk Management [20, 13] (Scenario Planning: i.e. modelling risk scenarios), Transportation [3] (Multi-modal (public) transportation)
- Robotics [7], Space Exploration [15, 11, 16, 19] (optimising data up- and downlink times, providing reasoning capabilities to autonomous agents), Terrestrial Exploration [17, 4] (improving returns of earth-observing satellites, guidance for autonomous underwater vehicles)

Further Remarks

Given the advantages of verification, also consider:

- Verification Effort vs. Increased Trustworthiness
- Unverified Assumptions (e.g. Compiler, Hardware)
- Good Running Time (thus high complexity) vs. Verification Effort
- for more info, go to https://fm-vs.github.io/