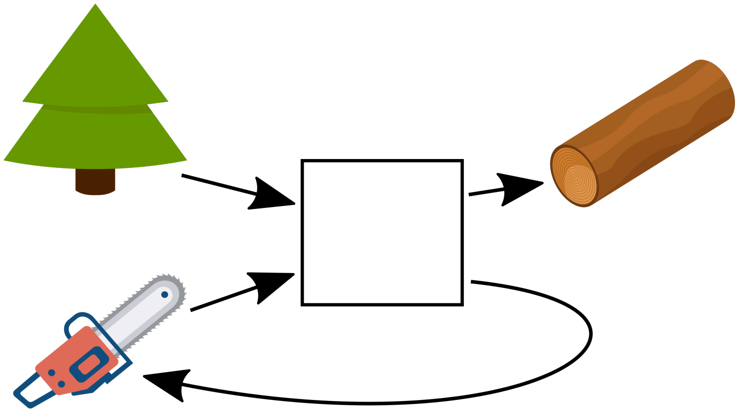
Categories for industrial planning

David Kruml, Jan Paseka

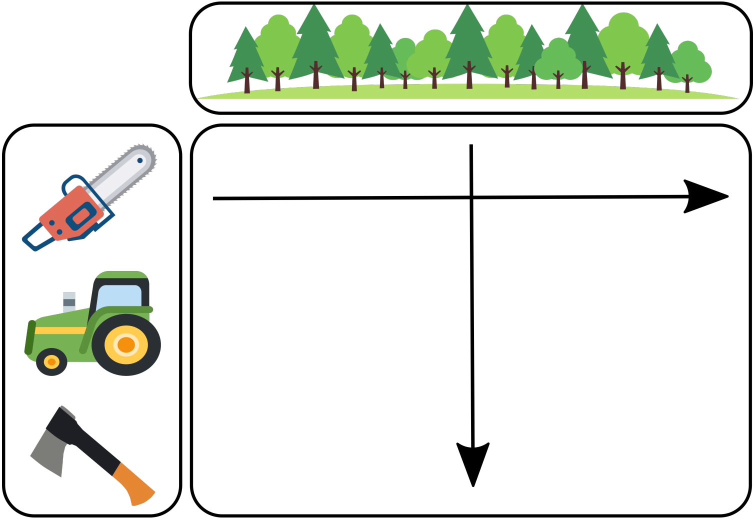
Masaryk University

CT 2025, Brno

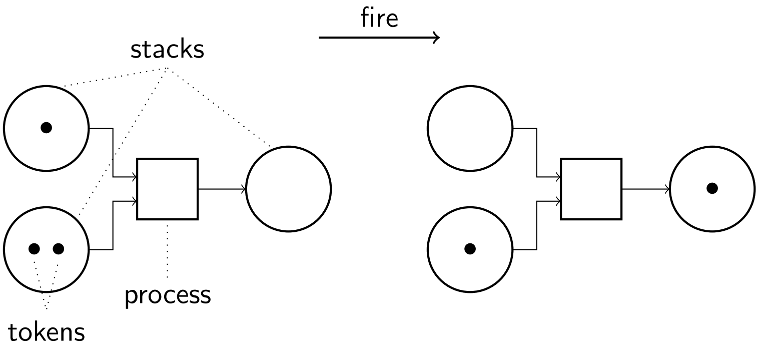
An example of process



Space of events and resource paths

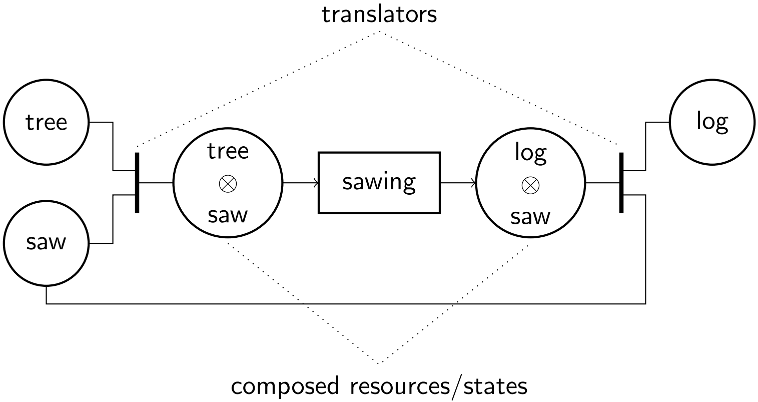


Petri nets



“Tensored” Petri nets

Strongly inspired by the Oxford school (Abramsky, Coecke, et al.) for quantum protocols.



Categorical formalization

- category theory — language, the way of thinking,
- **Proc** — dagger compact "corpus" category of all resources/states (objects) and processes (morphisms)
- I — bounded finite poset,
- *receipt* $R : I \to \mathbf{Proc}$,
- *schedule* (*Gantt diagram*) $S : I \to \mathbb{R}$ (time)
- *plan* = receipt + schedule

“Good” plan

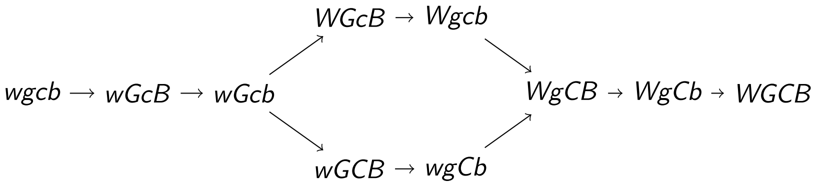
- We focus on resource inventories (MRP, MRP II, ERP): stacks must not underflow neither overflow.
- *Defects* (errors, collisions) are penalized $\Rightarrow$ multi criteria decision, objective function, we can optimize the plan.
- In practice, we prefer “soft constrains” than “hard constrains” — risky strategies could be more profitable (money save most of defects).

Benefits of categorical modeling

- All resources (material, machines, people, energy, externalities, ...) are "emancipated" and modeled the same way. (However, the economists should calculate all the weights for defects.)
- Two types of aggregation:
  - "categorical" — $\circ, \otimes$ (breakdown structures),
  - "instances $\to$ class" — functors (sharing of processes and subreceipts).
- The "logic" of **Proc** seems to be classical (cf. with linear logic of quantum protocols) and probably will be expressed by means of relations ($\Rightarrow$ allegories).
- Indices and orderings on summands, evaluation and comparison of plans $\Rightarrow$ 2-categories?

The wolf, goat, and cabbage problem

- Elementary resources: wolf, goat, cabbage, boat (with the farmer).
- Each item is in one of two states:
  - w, g, c, b — start bank,
  - W, G, C, B — final bank.
- Composed states:
  - $wgcb, wGcB, WgCb, \dots, WGCB$ — acceptable,
  - $wgCB, WgcB, WGcb, \dots$ — forbidden.
- Elementary process (operation): $wgcb \to wGcB$,
- Dagger: $(wgcb \to wGcB)^\dagger = wGcB \to wgcb$.
- Two optimal solutions:



Personalized views $\Rightarrow$ partitions on states/morphisms

- **Farmer's view:** 4 actions: "take a goat" $= \{wgcb \to wGcB, Wgcb \to WGcB, wGcB \to wgcb, \dots\}$, "take a wolf", "take cabbage", "manipulation cruise".
- **Wolf's view:** W/w, "alone with a goat".
- **Optimization view:**
  - forbidden states: penalty -100,
  - acceptable non-terminal states: penalty -1,
  - terminal state $WGCB$: penalty 0.
- Views could be expressed as functors.
- Now the problem is ready to be encoded to constraint programming language and solved with a computer (e. g. MiniZinc solver).

Other questions

- In manufacturing, many resources are "indistinguishable" and processes are repetitive $\Rightarrow$ high level of aggregation, powers of morphisms, etc. Multilevel planning $\Rightarrow$ "higher order regularity" of the flow code.
- Randomness = lack of knowledge, result of aggregation.
- The real planning problems are hard. We optimize by *simulated annealing*.
- In practice, departments of the company can "compete" (e. g. farmer vs. wolf) $\Rightarrow$ different weights, different evaluation of the flow, game theory.
- AI sometimes succeeds in encoding the task but it is still bad in optimization.

Thank you for your attention!