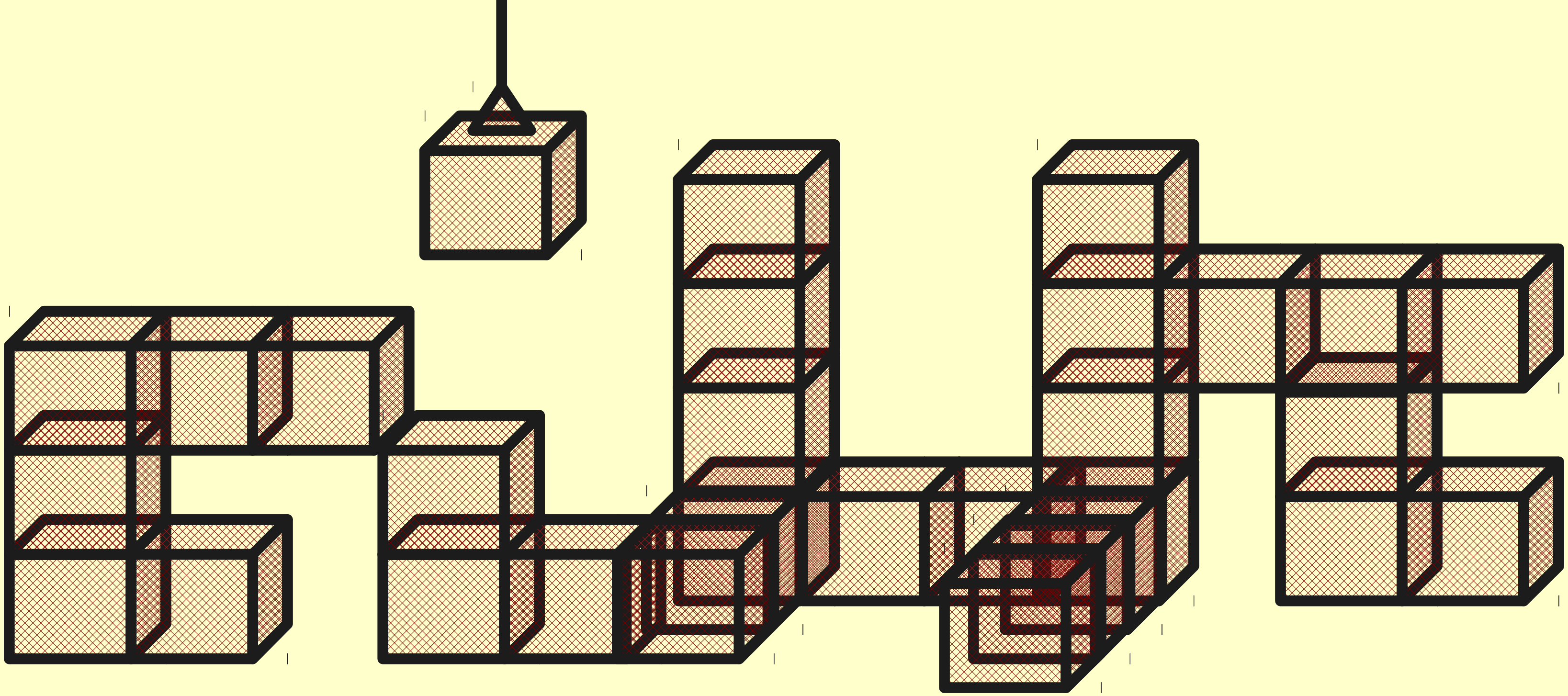
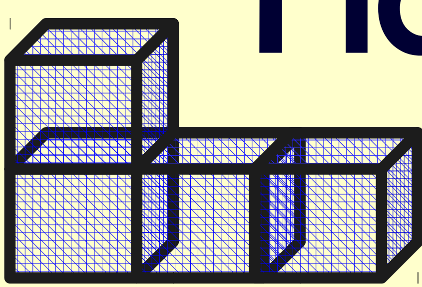


Deterministic Sequential Planning



Milan Ježek

Outline

- **Deterministic sequential planning**
- **Best planners** of IPC 2011
- **Plan-space planning**
- **My Basic Planner**

Restricted State-Transition
System Σ = (S,A,γ)

- **S** - Set of **states**
- **A** - Set of **actions**
- (**E** - Set of **events**)
- **γ** - **Transition function** $\gamma : S \times A \rightarrow P(S)$

Planning Problem P = (Σ, s_0, g):

- **Σ** - System modeling **states** and **transitions**
- s_0 - The **initial state**
- **g** - The **goal states**

States and Goals

- Represented as **sets of facts**
- **Closed World Assumption (CWA)**
  - Fact not listed in a state are assumed to be false
- **Goal state** - any state with all the goal facts

Operators and Actions

- **Operator** o = (**name**(o), **precond**(o), **effects**(o))
- An **action** is any ground instance of an operator
- **Move**(r, l, m) // Example of an operator
  - **Precond**:
    adjacent(l, m), at(r, l), not occupied(m)
  - **Effects**:
    at(r, m), occupied(m), not occupied(l), not at(r, l)

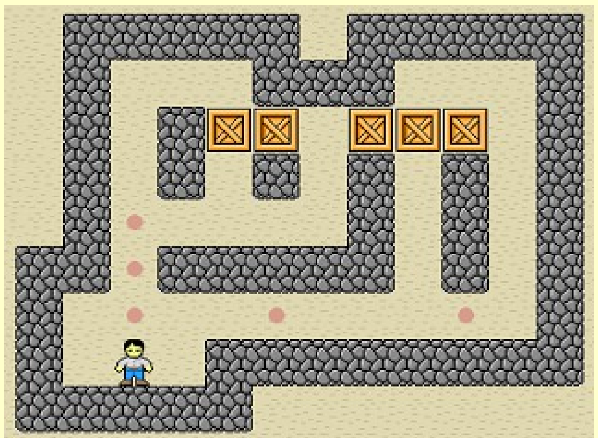
Solution of Planning Problem

- Sequence of **actions** $\langle a_1, a_2, \ldots, a_k \rangle$
- Sequence of **states** $\langle s_0, s_1, \ldots, s_k \rangle$
- Such that:
  - $\mathbf{s_i = \gamma(s_{i-1}, a_i)}$
  - $\mathbf{s_k}$ **satisfies g**

Extensions of the Classical Representation

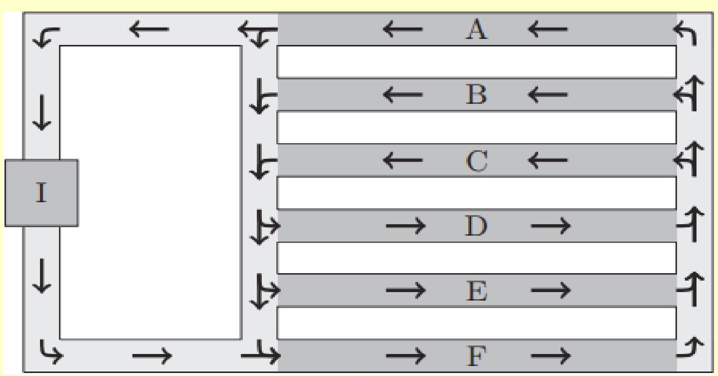
- Typed variables
- Non-negative costs
- Conditional effects
- Optional - **derived predicates**

Sequential Satisfying Domains



- Sokoban

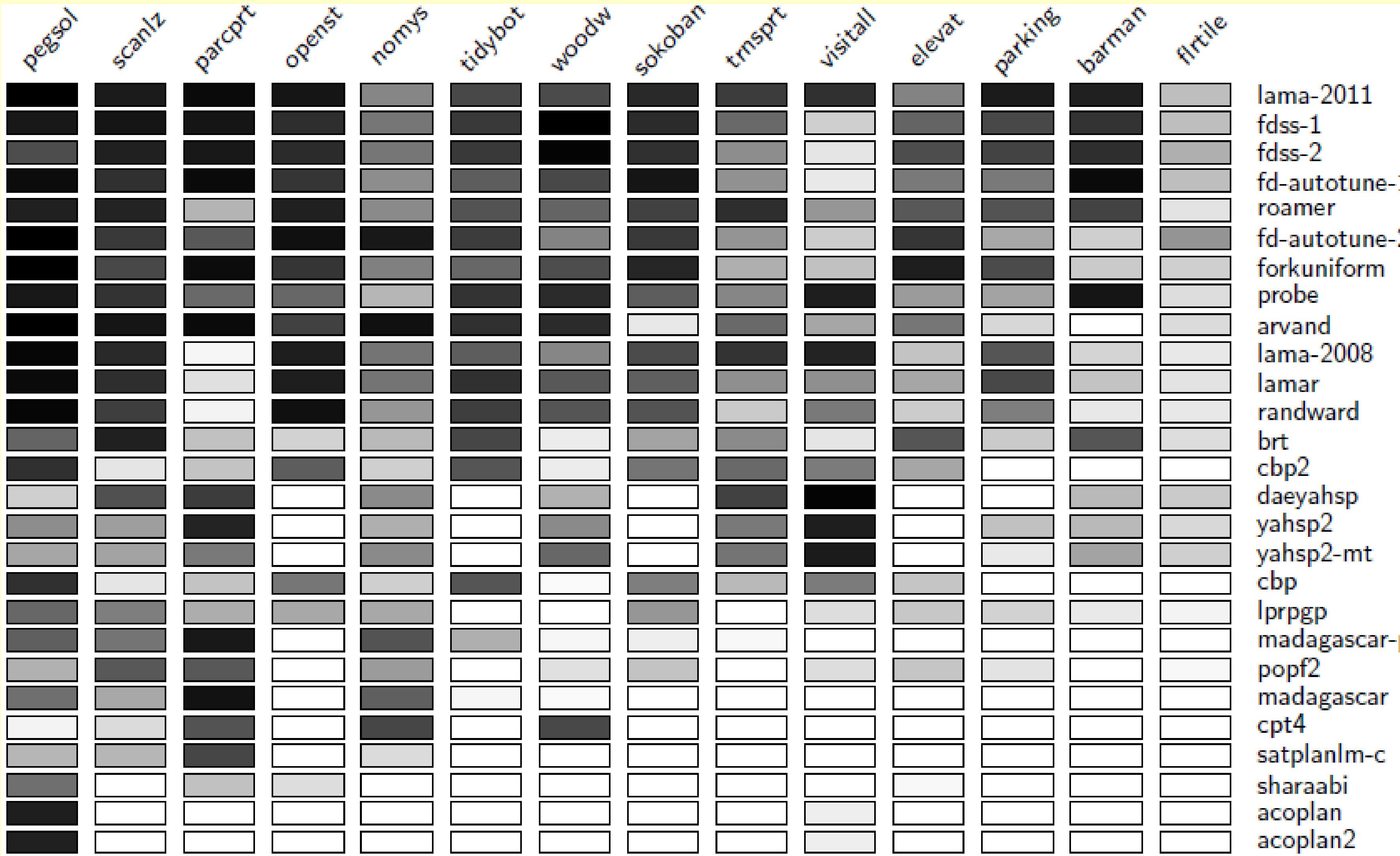
- Peg solitaire



- ScanAnalyzer

- TSP
- Elevators
- Transport

Best planners of IPC 2011



Some used **techniques**

- **Forward search, planning graph**
- **Landmarks** - variable assignments that must occur at some point in every solution plan
- ACOPlan - **Ant colony optimalization**
- Arvand - **Monte Carlo random walks** (MRW)
- BRT - (**Biased Rapidly exploring Tree**)
- Divide-and-Evolve – **Evolutionary computation**
- Fast downward (autotune), lama – **various algorithms and heuristics**
- Madagascar – **SAT**
- POPF2 - **Forward-Chaining Partial Order Planner**

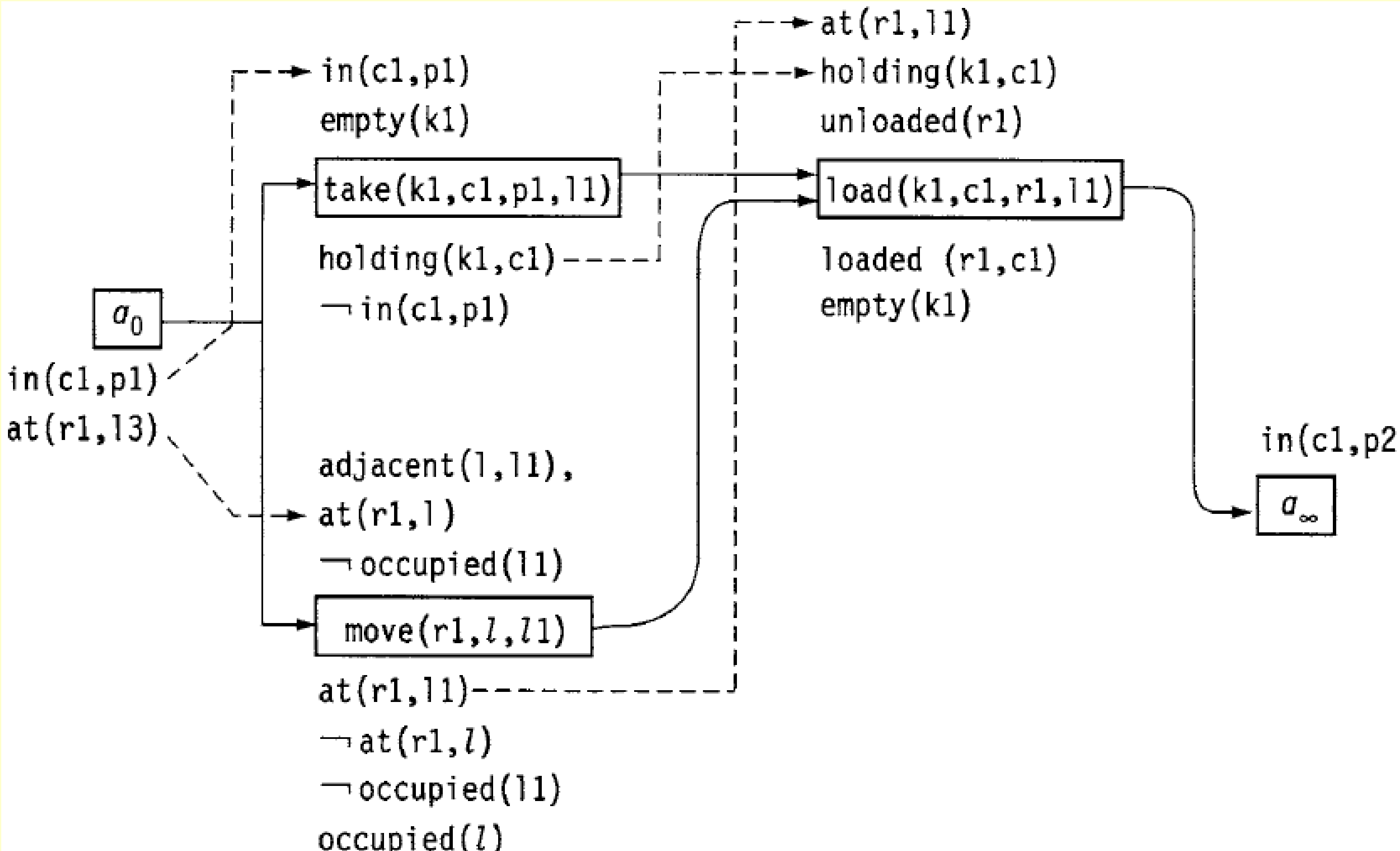
Plan-Space Planning

- Partially specified plans
- Refinement operations
- Least commitment principle

Partial Plan Π = (A, <, B, L)

- **A** - Set of **partially instantiated operators** $\{a_1, \ldots, a_k\}$
- **<** - **Partial order** on A ($a_i < a_j$)
- **B** - Set of **constraints** $x=y$, $x \neq y$ or $x \in D_x$
- **L** - Set of **causal relations** ($p: a_i \rightarrow a_j$)

Partial Plan Π = (A, <, B, L)



Plan-Space Planning

- Start with an **empty plan**
- **Repair** all **flaws** in partial plan step by step
  - **Add actions** to satisfy **open goals**
  - **Remove threats**
    - **Bind variables**
    - **Add ordering between actions**
    - **Add causal relations**

Solution for problem $P = (E, s_0, g)$

- Partial plan $\Pi = (A, <, B, L)$
  - Partial ordering $<$ and constraints B are **globally consistent**
  - **Any linearly ordered sequence of** fully instantiated **actions** from A **satisfying** $<$ and **B** goes **from s_0 to** a state satisfying **g**

Solution for problem P = (E, s_0, g)

- Partial plan Π = (A, <, B, L)
  - Partial ordering < and constraints B are **globally consistent**
  - There are **no flaws**
    - No open goals
    - No threats

PSP procedure

PSP(π)

flaws ← OpenGoals(π) U Threats(π)

if flaws = Ø **then return** π

select any flaw φ ∈ flaws

resolvers ← Resolve(φ, π)

if resolvers = Ø **then return** failure

non-deterministically choose a resolver p ∈ resolvers

π' ← Refine(p, π)

return PSP(π')

Algorithm PoP

PoP(π, agenda) // where π = (A, <, B, L)

if flaws = Ø then return π

setect any pair (aj, p) in and remove it from agenda

relevant $\leftarrow$ Providers(p, π)

if relevant = Ø then return failure

nondeterministica[[y choose an action $a_i \in$ relevant

L $\leftarrow$ L U { (p: $a_i \rightarrow a_j$) }

update B with the binding constraints of this causal link

if a_i is a new action in A then

update A with a_i

update < with ($a_i < a_j$), ($a_0 < a_i < a_\infty$)

update agenda with all preconditions of a_i

for each threat on (p: $a_i \rightarrow a_j$) or due to a_i do

resolvers $\leftarrow$ set of resolvers for this threat

if resolvers - 0 then return failure

non-deterministically choose a resolver in resolvers

add that resolver to < or to B

return PoP(π, agenda)

Algorithm PoP - **extensions**

- Conditional operators
- Flaw-Selection Heuristics
- Resolver-Selection Heuristics

My Basic Planner

- PDDL parser
- Preprocessor
    - Analyze operators/actions
        - map possible predecessors/successors for each action
        - Replace some operators with meta-operators
    - Analyze domain/instance of the problem
        - Derive some restrictions
- Plan-Space planning
- (CSP for some sub-problems)

Possible future extensions

- **Durative actions**
- **Qualitative** (temporal) **relations**
- **Preferences**
- **Learning** (domain specific properties)

Q&A

References

- Automated Planning: Theory and Practice; M. Ghallab, D. Nau, P. Traverso; Morgan Kaufmann, 2004
- http://ktiml.mff.cuni.cz/~bartak/planovani/
- http://sokoban-jd.blogspot.cz/
- http://en.wikipedia.org/wiki/Peg_solitaire
- The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem; M. Helmert, H. Lasinger; ICAPS 2010
- http://www.fast-downward.org/
- http://ipc.icaps-conference.org/
- http://www.plg.inf.uc3m.es/ipc2011-deterministic/