Planning and Scheduling: Complexity of Classical Planning

Hochschule Bonn-Rhein-Sieg

b-it Bonn-Aachen International Center for Information Technology

Prof. Dr.-Ing. Gerhard K. Kraetzschmar

Acknowledgements

- These slides refer to Chapter 3 of the textbook:
  Malik Ghallab, Dana Nau, and Paolo Traverso:
  Automated Planning: Theory and Practice
  Morgan Kaufmann, 2004
- These slides are an adaptation of slides by Dana Nau
- The contributions of these authors are gratefully acknowledged

Review: Classical Representation

- Function-free first-order language L
- Statement of a classical planning problem: $P = (s_0, g, O)$
- s_0 initial state - a set of ground atoms of L
- g goal formula - a set of literals
- operator$(name, preconditions, effects)$

```
take(crane1,loc1,c3,c1,p1)
   ;; crane crane1 at location loc1 takes c3 off c1 in pile p1
   precond: belong(crane1,loc1), attached(p1,loc1),
            empty(crane1), top(c3,p1), on(c3,c1)
   effects: holding(crane1,c3), ¬empty(crane1), ¬in(c3,p1),
            ¬top(c3,p1), ¬on(c3,c1), top(c1,p1)
```

- Classical planning problem: $P = (\Sigma, s_0, S_g)$

Review: Set-Theoretic Representation

- Like classical representation, but restricted to propositional logic
- State: a set of propositions - these correspond to ground atoms
  - { on-c-1-pallet, on-c1-r1, on-c1-c2, . . . , at-r1-l1, . . . }
- No operators, just actions

take-crane1-loc1-c3-c1-p1

precond :	belong-crane1-loc1, attached-p1-loc1
	empty-crane1, top-c3-p1, on-c3-c1
delete :	empty-crane1, on-c3-p1, top-c3-p1, on-c3-p1
add :	holding-crane1-c3, top-c1-p

- Weaker representational power than classical representation
  - Problem statement can be exponentially larger

Review: State-Variable Representation

- A state variable is like a record structure in a computer program
  - Instead of $on(c1, c2)$ we might write $cpos(c1) = c2$
- Load and unload operators:

$$\begin{array}{l}
\text{load}(c, r, l) \\
\quad \text{;; robot } r \text{ loads container } c \text{ at location } l \\
\quad \text{precond: } \text{rloc}(r) = l, \text{cpos}(c) = l, \text{rload}(r) = \text{nil} \\
\quad \text{effects: } \text{rload}(r) \leftarrow c, \text{cpos}(c) \leftarrow r
\end{array}$$

$$\begin{array}{l}
\text{unload}(c, r, l) \\
\quad \text{;; robot } r \text{ unloads container } c \text{ at location } l \\
\quad \text{precond: } \text{rloc}(r) = l, \text{rload}(r) = c \\
\quad \text{effects: } \text{rload}(r) \leftarrow \text{nil}, \text{cpos}(c) \leftarrow l
\end{array}$$

- Equivalent power to classical representation
  - Each representation requires a similar amount of space
  - Each can be translated into the other in low-order polynomial time
- Classical representation is more popular, mainly for historical reasons
  - In practice, state-variable representation is probably more convenient

Motivation and Outline

- Recall that in classical planning even simple problems can have huge search spaces, e.g.
  - States of DWR with 5 locations, 3 piles, 3 robots and 100 containers is

    $$10^{277}$$

  - Largest estimates of particles in universe is only about

    $$10^{87}$$

- How difficult is it to solve classical planning problems?

Outline

- Background on complexity analysis
- Restrictions (and a few generalizations) of classical planning
- Decidability and undecidability
- Tables of complexity results
  - Classical representation
  - Set-theoretic representation
  - State-variable representation

Complexity Analysis

- Complexity analyses are done on language-recognition problems
  - A language is a set L of strings over some alphabet A
  - Recognition procedure for L
    - A procedure R(x) that returns "yes" iff the string x is in L
    - If x is not in L, then R(x) may return "no" or may fail to terminate
- Translate classical planning into a language-recognition problem
- Examine the language-recognition problem's complexity

Planning as a Language-Recognition Problem

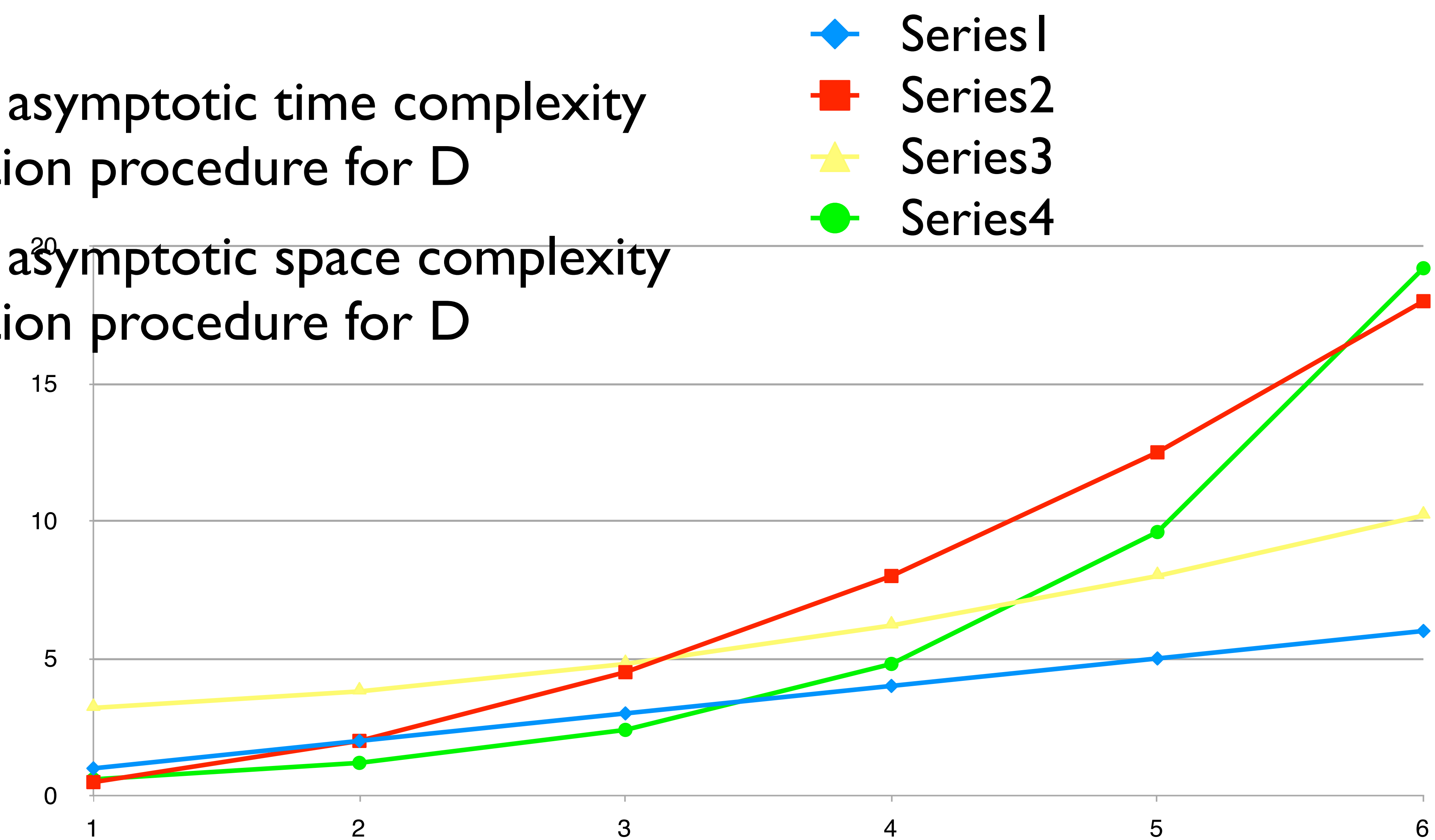
- We will consider two language-recognition problems:

$$\text{Plan-Existence} = \{P \mid P \text{ is the statement of a planning problem that has a solution}\}$$

$$\text{Plan-Length} = \{(P, n) \mid P \text{ is the statement of a planning problem that has a solution of } length \leq n\}$$

- Look at complexity of PLAN-EXISTENCE and PLAN-LENGTH under different conditions

- Suppose R is a recognition procedure for D
- Complexity of R
  - $T_R(n) =$ worst-case runtime for R on strings in D of length n
  - $S_R(n) =$ worst-case space requirement for R on strings in D of length n
- Complexity of D
  - $T_D =$ best asymptotic time complexity of any recognition procedure for D
  - $S_D =$ best asymptotic space complexity of any recognition procedure for D



Complexity Classes

NLOGSPACE $\subseteq$	(nondeterministic procedure, logarithmic space)
$\subseteq$ P	(deterministic procedure, polynomial time)
$\subseteq$ NP	(nondeterministic procedure, polynomial time)
$\subseteq$ PSPACE	(deterministic procedure, polynomial space)
$\subseteq$ EXPTIME	(deterministic procedure, exponential time)
$\subseteq$ NEXPTIME	(nondeterministic procedure, exponential time)
$\subseteq$ EXPSPACE	(deterministic procedure, exponential space)

- Let C be a complexity class and p be a language-recognition problem
  - p is C-hard if for every problem q in C,
    q can be reduced to p in a polynomial amount of time
    - NP-hard, PSPACE-hard, etc.
  - p is C-complete if p is C-hard and p is also in C
    - NP-complete, PSPACE-complete, etc.

Possible Conditions

- Do we give the operators as input to the planning algorithm, or fix them in advance?
- Do we allow infinite initial states?*
- Do we allow function symbols?*
- Do we allow negative effects?
- Do we allow negative preconditions?
- Do we allow more than one precondition?
- Do we allow operators to have conditional effects?*
  - i.e., effects that only occur when additional preconditions are true

- Question marked with * and answered “yes” take us outside of classical planning

Decidability of Planning

Halting problem

Can cut off the search at every path of length n

Allow function symbols?	Decidability of PLAN-EXISTENCE	Decidability of PLAN-LENGTH
no$^{\alpha}$	decidable	decidable
yes	semidecidable$^{\beta}$	decidable

$^{\alpha}$This is ordinary classical planning.

$^{\beta}$True even if we make several restrictions (see text).

Next: Analyze complexity for the decidable cases

Complexity of Planning: Classical Representation

PSPACE-complete or NP-complete for some sets of operators

Kind of representation	How the operators are given	Allow negative effects?	Allow negative preconditions?	Complexity of PLAN-EXISTENCE	Complexity of PLAN-LENGTH
classical rep.	in the input	yes	yes/no	EXPSPACE-complete	NEXPTIME-complete
		no	yes	NEXPTIME-complete	NEXPTIME-complete
			no	EXPTIME-complete	NEXPTIME-complete
			no^{α}	PSPACE-complete	PSPACE-complete
	in advance	yes	yes/no	PSPACE $^{\gamma}$	PSPACE $^{\gamma}$
		no	yes	NP $^{\gamma}$	NP $^{\gamma}$
			no	P	NP $^{\gamma}$
			no^{α}	NLOGSPACE	NP

no operator has >1 precondition

Caveat: Worst-Case Results

- Caveat: these are worst-case results
  - Individual planning domains can be much easier
- Example: both DWR and Blocks World fit here , but neither is that hard
  - For them, PLAN-EXISTENCE is in P and PLAN-LENGTH is NP-complete

Kind of represen-tation	are given	effects?	precon-ditions?	EXISTENCE	exity LENGTH
classical rep.	in the input	yes	yes/no	EXPSPACE-complete	NEXPTIME-complete
		no	yes	NEXPTIME-complete	NEXPTIME-complete
			no	EXPTIME-complete	NEXPTIME-complete
			no$^{\alpha}$	PSPACE-complete	PSPACE-complete
	in advance	yes	yes/no	PSPACE $^{\gamma}$	PSPACE $^{\gamma}$
		no	yes	NP $^{\gamma}$	NP $^{\gamma}$
			no	P	NP $^{\gamma}$
			no$^{\alpha}$	NLOGSPACE	NP

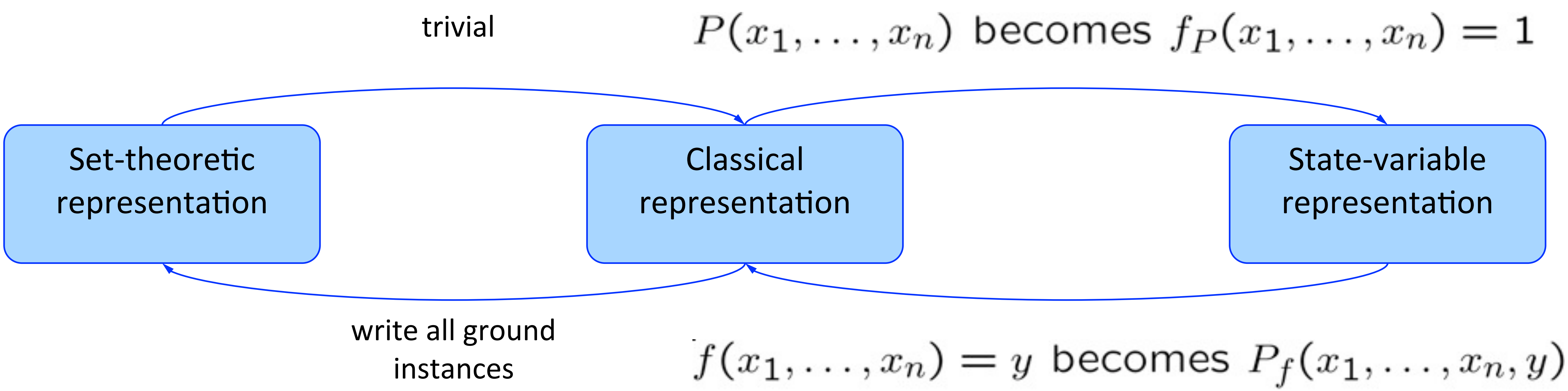
Plan-Length vs Plan-Existence

Kind of representation	How the operators are given	Allow negative effects?	Allow negative preconditions?	Complexity of PLAN-EXISTENCE	Complexity of PLAN-LENGTH
classical rep.	in the input	yes	yes/no	EXPSPACE-complete	NEXPTIME-complete
		no	yes	NEXPTIME-complete	NEXPTIME-complete
			no	EXPTIME-complete	NEXPTIME-complete
			no$^{\alpha}$	PSPACE-complete	PSPACE-complete
				PSPACE $^{\gamma}$	PSPACE $^{\gamma}$
				NP $^{\gamma}$	NP $^{\gamma}$
				P	NP $^{\gamma}$
			no$^{\alpha}$	NLOGSPACE	NP

- Here, PLAN-LENGTH is easier than PLAN-EXISTENCE
  for the same reason as in the decidability table
  - Can cut off every search path at depth n

Equivalences

- Set-theoretic representation and ground classical representation are basically identical
  - For both, exponential blowup in the size of the input
  - Thus complexity looks smaller as a function of the input size
- Classical and state-variable representations are equivalent, except that some of the restrictions aren't possible in state-variable representations
  - Hence, fewer lines in the table



Complexity of Planning: Set-Theoretic and State-Variable Representations

no operator has >1 precondition

every operator with >1 precondition is the composition of other operators

Like classical rep, but fewer lines in the table

Kind of representation	How the operators are given	Allow negative effects?	Allow negative preconditions?	Complexity of PLAN-EXISTENCE	Complexity of PLAN-LENGTH
set-theoretic or ground classical rep.	in the input	yes	yes/no	PSPACE-complete	PSPACE-complete
		no	yes	NP-complete	NP-complete
			no	P	NP-complete
			$\text{no}^{\alpha}/\text{no}^{\beta}$	NLOGSPACE-complete	NP-complete
	in advance	yes/no	yes/no	constant time	constant time
state-variable rep.	in the input	yes^{δ}	yes/no	EXPSPACE-complete	NEXPTIME-complete
	in advance	yes^{δ}	yes/no	PSPACE $^{\gamma}$	PSPACE $^{\gamma}$
ground state-variable rep.	in the input	yes^{δ}	yes/no	PSPACE-complete	PSPACE-complete
	in advance	yes^{δ}	yes/no	constant time	constant time