OPPA European Social Fund

Prague & EU: We invest in your future.

Lecture slides for

Automated Planning: Theory and Practice

Chapter 3
Complexity of Classical Planning

Dana S. Nau

CMSC 722, AI Planning

University of Maryland, Spring 2008

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: $\mathcal{P} = (\Sigma, s_0, S_g)$

Review: Set-Theoretic Representation

- Like classical representation, but restricted to propositional logic
- State: a set of propositions - these correspond to ground atoms
  - {on-c1-pallet, on-c1-r1, on-c1-c2, …, at-r1-l1, at-r1-l2, …}
- 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, in-c3-p1, top-c3-p1, on-c3-p1
add: holding-crane1-c3, top-c1-p1

- 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:

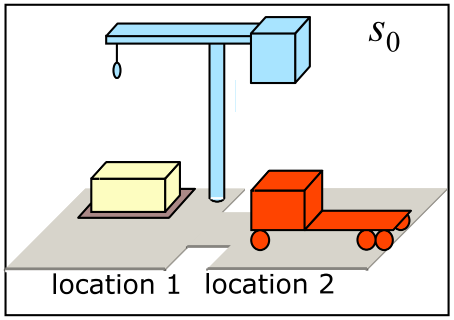
$\mathsf{load}(c, r, l)$
;; robot r loads container c at location l
precond: $\mathsf{rloc}(r) = l, \mathsf{cpos}(c) = l, \mathsf{rload}(r) = \mathsf{nil}$
effects: $\mathsf{rload}(r) \leftarrow c, \mathsf{cpos}(c) \leftarrow r$

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

- 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 many cases, state-variable representation is more convenient

Motivation

- Recall that in classical planning, even simple problems can have huge search spaces
  - Example:
    - » DWR with five locations, three piles, three robots, 100 containers
    - » 10^{277} states
    - » About 10^{190} times as many states as there are particles in universe



- How difficult is it to solve classical planning problems?
- The answer depends on which representation scheme we use
  - Classical, set-theoretic, state-variable

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 *decision problems* or *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

- Consider the following two languages:

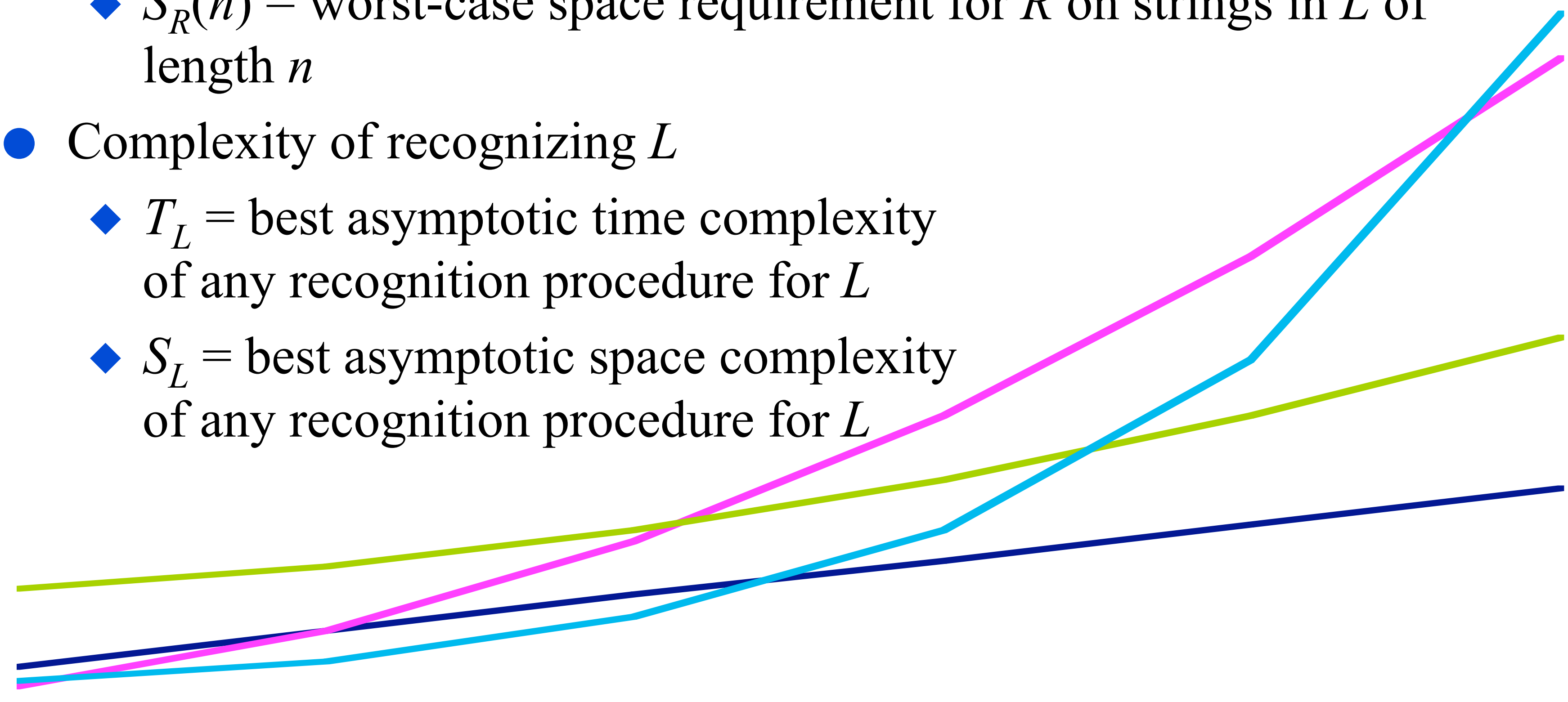
PLAN-EXISTENCE = $\{P : P$ is the statement of a planning problem that has a solution$\}$

PLAN-LENGTH = $\{(P,n) : P$ is the statement of a planning problem that has a solution of length $\leq n\}$

- Look at complexity of recognizing PLAN-EXISTENCE and PLAN-LENGTH under different conditions
    - Classical, set-theoretic, and state-variable representations
    - Various restrictions and extensions on the kinds of operators we allow

Complexity of Language-Recognition Problems

- Suppose R is a recognition procedure for a language L
- Complexity of R
  - $T_R(n)$ = worst-case runtime for R on strings in L of length n
  - $S_R(n)$ = worst-case space requirement for R on strings in L of length n
- Complexity of recognizing L
  - T_L = best asymptotic time complexity of any recognition procedure for L
  - S_L = best asymptotic space complexity of any recognition procedure for L



Complexity Classes

- Complexity classes:
  - NLOGSPACE (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 L be a language
  - Recognizing L is C-hard if for every language L' in C, L' can be reduced to L in a polynomial amount of time
    - » NP-hard, PSPACE-hard, etc.
  - Recognizing L is C-complete if L is C-hard and L 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

These take us outside 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

$^{\gamma}$ 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$^{\alpha}$	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

$^{\alpha}$ no operator has >1 precondition

- **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

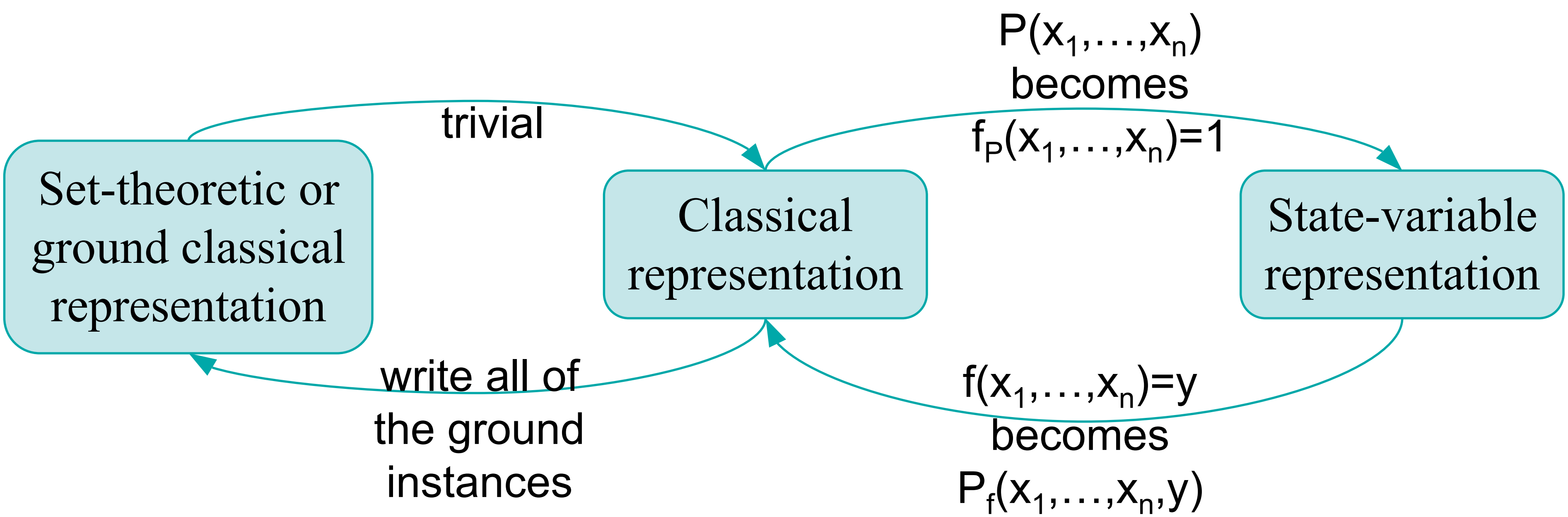
Kind of representation	How the operators are given	Allow negative effects?	Allow negative preconditions?	Complexity of PLAN-EXISTENCE	Complexity of PLAN-LENGTH
classical rep.	in the input	yes	yes/no	EXPSPACE-complete	NEXPTIME-complete
		no	yes	NEXPTIME-complete	NEXPTIME-complete
			no	EXPTIME-complete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

- Often PLAN-LENGTH is harder than PLAN-EXISTENCE
- But it's easier here:
  - We can cut off every search path at depth *n*

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	$\boxed{\text{NEXPTIME-complete}}$
		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^{γ}	NP^{γ}
			no	P	NP^{γ}
			no^{α}	NLOGSPACE	NP

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



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

Like classical rep, but fewer lines in the table

$^{\alpha}$ no operator has >1 precondition

$^{\beta}$ every operator with >1 precondition is the composition of other operators

OPPA European Social Fund
Prague & EU: We invest in your future.