Planejamento em
Inteligência Artificial

Capítulo 3
Complexidade do Planejamento Clásico

José de Jesús Pérez-Alcázar

MAC 5788 - IME/USP
segundo semestre de 2005

Revisão: Representação Clássica

- Linguagem L de primeira ordem livre de Funções
- Declaração de um problema de planejamento clássico: $P = (s_0, g, O)$
- s_0: estado inicial - um conjunto de átomos ground de L
- g: fórmula meta - um conjunto de literais
- *Operador*: (nome, pre-condições, efeitos)

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)

- Problema de Planejamento Clássico: $\mathsf{P} = (\Sigma, s_0, S_g)$

Revisão: Representação de Teoria de Conjuntos

- Como a representação clássica, mas restringida à lógica proposicional
- Estado: um conjunto de proposições - estes correspondem a átomos ground
  - {on-c1-pallet, on-c1-r1, on-c1-c2, …, at-r1-l1, at-r1-l2, …}
- Não há operadores, só ações

  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

- Poder representacional mais fraca que a clássica
  - Declaração do problema pode ser exponencialmente maior.

Revisão: Representação de variáveis de estado

- Uma variável de estado é como uma estrutura de registro num programa de computador
  - No lugar de on(c1,c2) nós deveríamos escrever cpos(c1)=c2
- Operadores Carregue e Descarregue:

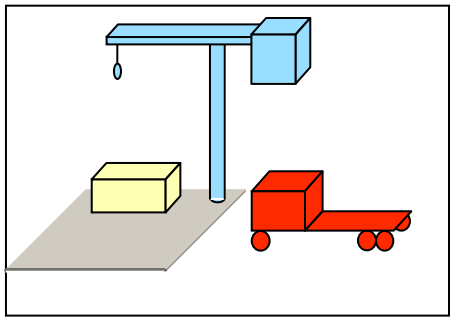
$\mathsf{load}(c, r, l)$
;; robot r loads container c at location l
precond: $\mathsf{rloc}(r) = l, \mathsf{cpos}(c) = l, \mathsf{rload}(r) = \mathsf{nil}$
effects: $\mathsf{rload}(r) \leftarrow c, \mathsf{cpos}(c) \leftarrow r$

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

- Potência equivalente à representação clássica
  - Cada representação precisa de uma quantidade de espaço similar
  - Cada uma pode ser traduzido na outra em tempo polinomial
- Representação clássica é mais popular, principalmente por razões históricas
  - Na prática, representação de variáveis de estado é provavelmente mais conveniente

Motivação

- Lembremos que no planejamento clássico, mesmo problemas simples podem ter imensos espaços de busca
  - Exemplo:
    - » DWR com cinco locações, três pilhas, três robots, 100 contenedores
      - 10^{277} estados
    - » Os estimativos mais completos de partículas no universo: somente cerca de 10^{87}



- Que tão difícil é resolver problemas de planejamento clássico?

Tópicos da aula

- Fundamentos sobre análise de complexidade
- Restrições (e umas poucas generalizações) do planejamento clássico
- Decidibilidade e indecibilidade
- Tabelas de resultados de complexidade
  - Representação clássica
  - Representação de teoria de conjuntos
  - Representação de variáveis de estado

Análise de Complexidade

- Análises de complexidade tradicionalmente são focados nos problemas de reconhecimento de *linguagens*
  - Uma linguagem é um conjunto L de strings de algum alfabeto A
  - Procedimento de reconhecimento para L
    - » Um procedimento $R(x)$ que retorna "yes" sse o string x está em L
    - » Se x não está em L, então $R(x)$ pode retornar "no" ou pode falhar para terminar
- Traduzir o planejamento clássico em um problema de reconhecimento de linguagens.
- Verificar a complexidade do problema de reconhecimento de linguagens

Planejamento como um problema de Reconhecimento de Linguagens

- Dado um conjunto D de declarações de problemas de planejamento
- Nós consideraremos dois problemas de reconhecimento de problemas:

PLAN-EXISTENCE(D)

= $\{P : P \in$ D é a declaração de um problema de planejamento que tem uma solução$\}$

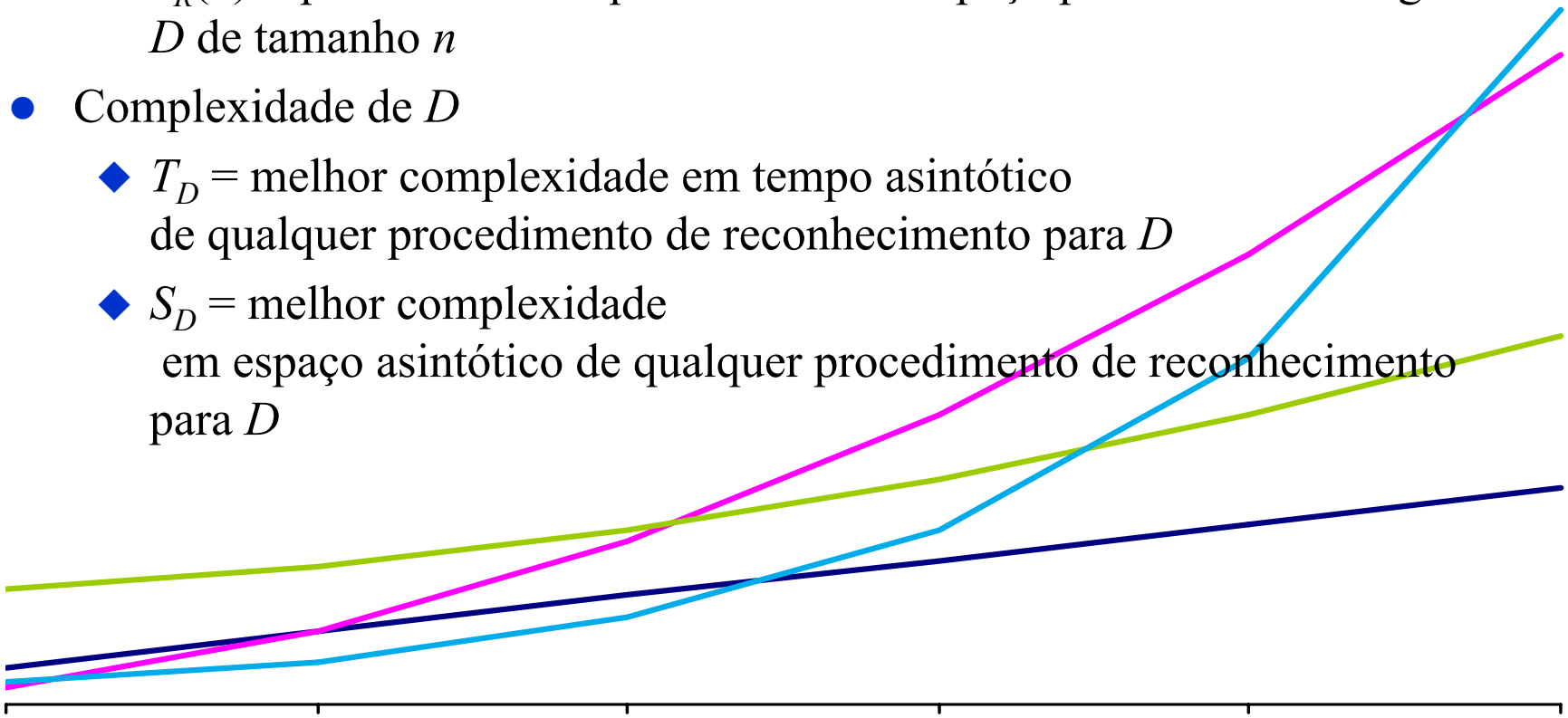
PLAN-LENGTH(D)

= $\{(P,n) : P \in$ D é a declaração de um problema de planejamento que tem uma solução de tamanho $\leq n\}$

- Verificar a complexidade de PLAN-EXISTENCE e PLAN-LENGTH sob diferentes condições (D pode ser o conjunto de todos os problemas de planejamento clássico)

Complexidade dos problemas de reconhecimento de linguagens

- Suponha que R é um procedimento de reconhecimento para D
- Complexidade de R
  - $T_R(n)$ = pior caso em tempo para R sobre strings em D de tamanho n
  - $S_R(n)$ = pior caso em requerimentos de espaço para R sobre strings em D de tamanho n
- Complexidade de D
  - T_D = melhor complexidade em tempo asintótico de qualquer procedimento de reconhecimento para D
  - S_D = melhor complexidade em espaço asintótico de qualquer procedimento de reconhecimento para D



Classes de Complexidade

- Classes de complexidade :
  - NLOGSPACE (procedimento não-determinístico, espaço logarítmico)
    - $\subseteq$ P (procedimento determinístico, tempo polinomial)
    - $\subseteq$ NP (procedimento não-determinístico, tempo polinomial)
    - $\subseteq$ PSPACE (procedimento determinístico, espaço polinomial)
    - $\subseteq$ EXPTIME (procedimento determinístico, tempo exponencial)
    - $\subseteq$ NEXPTIME (procedimento não-determinístico, tempo exponencial)
    - $\subseteq$ EXPSPACE (procedimento determinístico, espaço exponencial)
- Seja C uma classe de complexidade e p um problema de reconhecimento de linguagens
  - p é C-difícil se para todo problema q em C, q pode ser reduzido a p em tempo polinomial
    - » NP-difícil, PSPACE-difícil, etc.
  - p é C-completo se p is C-difícil e p é também em C
    - » NP-completo, PSPACE-completo, etc.

Possíveis restrições

- Nós damos os operadores como entrada ao algoritmo de planejamento, ou os estabelecemos de antemão?
- Permitimos estados iniciais infinitos?*
- Permitimos símbolos funcionais?*
- Permitimos efeitos negativos?
- Permitimos pré-condições negativas?
- Permitimos mais de uma pre-condição?
- Permitimos que os operadores tenham efeitos condicionais?*
  - i.e., efeitos que ocorrem unicamente quando pré-condições adicionais são verdadeiras

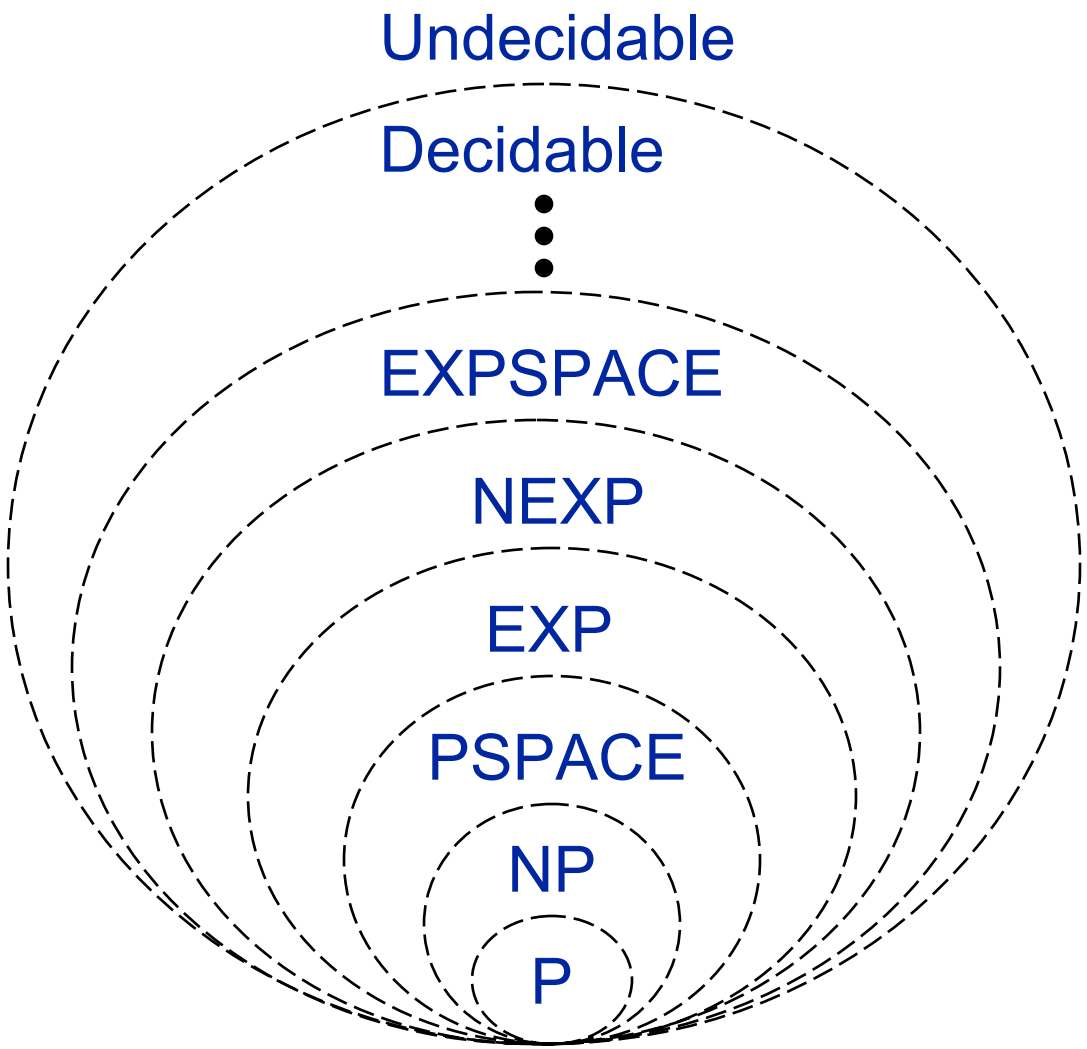
*Estes estão fora do planejamento clássico.

Decidibilidade

Problema de Decisão: um problema com uma resposta sim/não
ex:. “N é primo?”

- *Decidível:* Se existe um programa (ex: uma Máquina de Turing) que toma qualquer instância e para corretamente com uma resposta “sim” ou “não”.
- *Semi-decidível:* Se o programa para com a resposta certa num dos casos (“sim” ou “não”) mas não no outro caso (não para)
- *Indecidível:* Não existe algoritmo para resolver o problema. Ex: Problema da Parada.

Hierarquia das classes de complexidade



$PSPACE \subset EXPSPACE$

$P \subset EXP$

$PSPACE = NPSPACE$

$P \subseteq NP \subseteq PSPACE$

$P =? NP$

Decidibilidade de Planejamento

Proposição 3.1

Problema da Parada

Pode sair da busca em tota trajetória de tamanho *n*

Allow function symbols?	Decidability of PLAN-EXISTENCE	Decidability of PLAN-LENGTH
no[α]	decidable	decidable
yes	semidecidable[β]	decidable

[α]This is ordinary classical planning.

[β]True even if we make several restrictions (see text).

A seguir: análise de complexidade para os casos decidíveis.

Decidibilidade de Planejamento

- Proposição 3.1. Para o caso clássico o problema PLAN-EXISTENCE é decidível → o número de possíveis estados é finito, um algoritmo de força bruta pode achar uma solução.
- Proposição 3.2. Se estendemos o planejamento clássico (ou variáveis de estado) para permitir símbolos funcionais, então PLAN-LENGTH é ainda decidível → podemos encurtar a busca quando se chegue a um plano de tamanho k.
- Proposição 3.3. Se estendemos o planejamento clássico (ou variáveis de estado) para permitir símbolos funcionais, PLAN-EXISTENCE é semi-decidível → para uma declaração $P = (s_0, g, O)$ os algoritmos de busca (ex: lifted-backward-search(P)) terminará se existir uma solução para P. Para demonstrar que PLAN-EXISTENCE é não decidível ...

Resultados de decidibilidade de[Erol et al. 94]

- Explora o relacionamento entre planejamento e programação em lógica.
- Podemos transformar um problema de planejamento sem listas delete (efeitos negativos) ou pré-condições negativas a um programa em lógica (e vice-versa) em tempo polinomial:
  - R1: $a \leftarrow b1 \wedge b2 \wedge b3$
  - Op_R1: [pre: {b1, b2, b3} add: {a} del: {}]
- Símbolos funcionais → não decidível
  - A menos que existam condições de acyclicity e boundedness.
- Se não existem símbolos funcionais e o estado inicial é finito → decidível

Resultados da Complexidade

- Como é apresentado na tabela a complexidade computacional para os problemas de planejamento clássico (teoria de conjuntos e variáveis de estado) pode ser constante ou até EXPSPACE-completo dependendo das restrições.
- Nau et. al. (2004) demonstram que o caso comúm de planejamento clássico é EXPSPACE-completo (primeira linha da tabela). Eles reduzem um conhecido problema de reconhecimento de linguagens EXPSPACE-completo, o problema da máquina de Turing EXPSPACE-limitado ao problema de planejamento clássico.

Outros resultados: Estados, operadores, planos.
Quantos, Que tão grande?

- Assumindo estados finitos, n objetos, m predicados com aridade r, e o operadores (com s variáveis max cada) e que não existem símbolos funcionais:
  - Possíveis átomos: $p = m\ n^r$

    => Cada estado precisa espaço exponencial
  - Possíveis estados = Powerset{p} = 2^p

    => O espaço de estados é double exponential
  - Possíveis operadores ground = o n^s

Outros resultados

- Sem restrições: uma instância de operador precisaria aparecer várias vezes no mesmo plano → busca a través de todos os estados; cada estado consome um espaço exponencial → PLAN-EXISTENCE pode ser resolvido em EXPSPACE.
- Sem efeitos negativos: operadores unicamente precisam aparecer uma só vez; como o número de operadores é exponencial assim será sua escolha → PLAN-EXISTENCE pode ser resolvido em NEXPTIME.
- Se restringimos ainda para não ter pré-condições negativas → nenhum operador pode remover as pré-condições dos outros → a ordem dos operadores não importa → PLAN-EXISTENCE pode ser resolvido em EXPTIME.
- A pesar das restrições → PLAN-LENGTH permanece NEXPTIME.
- Se cada operador é restrito a uma pré-condição no máximo → fácil busca "backward" e o número de sub-metas não incrementa → PLAN-EXISTENCE e PLAN-LENGTH podem ser resolvidos em PSPACE

Outros resultados

- Se restringimos todos os átomos para serem ground

Propriedades interessantes achadas

- Estender os operadores de planejamento para permitir efeitos condicionais não afeta os nossos resultados.
- Comparando a complexidade de PLAN-EXISTENCE no caso de teoria de conjuntos com o caso clássico revela um padrão regular: em muitos casos, a complexidade do primeiro é um nível menos difícil que o segundo.

Complexidade de Planejamento

PSPACE-completo ou NP-completo para algúns conjuntos de operadores

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

no máximo tem1 pré-condição

- Advertência: estes são resultados no pior *caso*
  - Domínios de planejamento Individuais podem ser muito mais fáceis
- Exemplo: tanto DWR como o mundo dos blocos colocs. aqui, mas nenhum é difícil
  - para eles, PLAN-EXISTENCE está em P e PLAN-LENGTH é 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

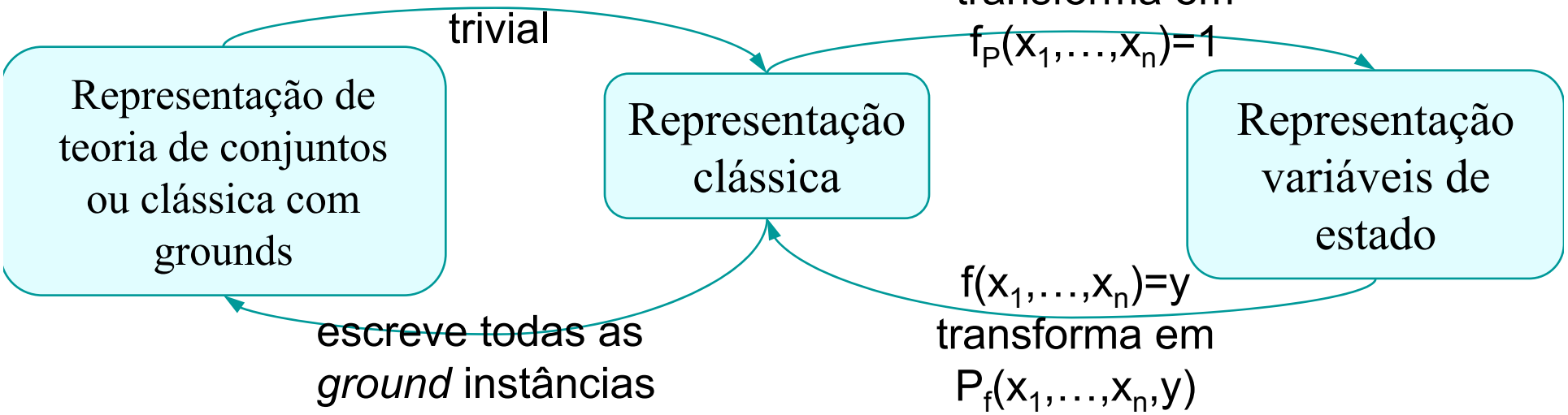
Aqui, PLAN-LENGTH é mais fácil que PLAN-EXISTENCE pela mesma razão que na tabela decidibilidade

- Podemos eliminar toda trajetória de busca na profundidade *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	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

Equivalências

- A representação de teoria de conjuntos e a representação clássica com termos ground são basicamente idênticas
  - Para os dois casos, há explosão combinatória no tamanho da entrada
  - Assim, a complexidade diminui em função do tamanho da entrada
- As representações clássica e de variáveis de estado são equivalentes, exceto por algumas restrições que não são possíveis na representação de variáveis de estado
  - Portanto, poucas linhas na tabela



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
			no^{α}/no^{β}	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

Como rep clássica, mas poucas linhas na tabela

no máximo tem1 pré-condição

todo operador com >1 pré-condições é a composição de outros operadores

Conclusões

- Planejamento independente de domínio é em geral muito difícil: PSPACE, NP, …
- Mesmo para casos restritos (Veja Bylander 94):
  - 2 preconds positivas, 2 efeitos (PSPACE)
  - 1 precond, 1 efeito posítivo (NP)

    … no pior caso …
- O que acontece no caso médio, domínios estruturados, etc.?

=> Heurísticas, reuso de soluções, aprendizagem

Bibliografia

- Kutluhan Erol, Dana Nau, and V.S. Subrahmanian. "Complexity, Decidability and Undecidability Results for Domain-Independent Planning", Artificial Intelligence Journal, Vol. 76, Nr. 1-2, pps. 75-88, July, 1995.
- Tom Bylander, "The Computational Complexity of Propositional STRIPS Planning," Artificial Intelligence, 69:165-204, 1994.
- Notas do curso dos professores: Jim Blythe, Jose-Luis Ambite, e Yolanda Gil do Information Science Institute –USC.

  http://www.isi.edu/~blythe/cs541/

Anexo

Lifted-backward-search(O, s_0, g)

$\pi \leftarrow$ the empty plan

loop

if s_0 satisfies g then return π

$A \leftarrow \{(o, \theta) | o$ is a standardization of an operator in O,

θ is an mgu for an atom of g and an atom of $\text{effects}^+(o)$,

and $\gamma^{-1}(\theta(g), \theta(o))$ is defined$\}$

if $A = \emptyset$ then return failure

nondeterministically choose a pair $(o, \theta) \in A$

$\pi \leftarrow$ the concatenation of $\theta(o)$ and $\theta(\pi)$

$g \leftarrow \gamma^{-1}(\theta(g), \theta(o))$