بسم الله الرحمن الرحیم

هوش مصنوعی

فصل ۱۰

طرح‌ریزی کلاسیک

Classical Planning

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

http://courses.fouladi.ir/ai

هوش مصنوعی

طرح‌ریزی کلاسیک

۱

تعریف طرح‌ریزی کلاسیک

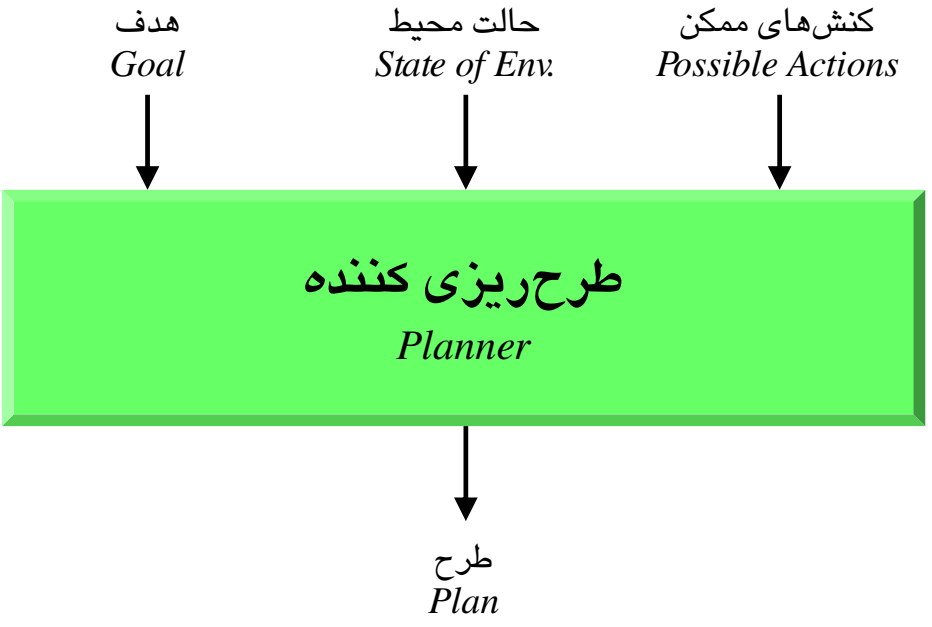
طرح‌ریزی

PLANNING

طرح‌ریزی *Planning*	مسئله‌ی یافتن **دنباله‌ای از کنش‌ها** برای دست‌یابی به یک **هدف**

معماری طرح‌ریزی کننده

ARCHITECTURE OF A PLANNER



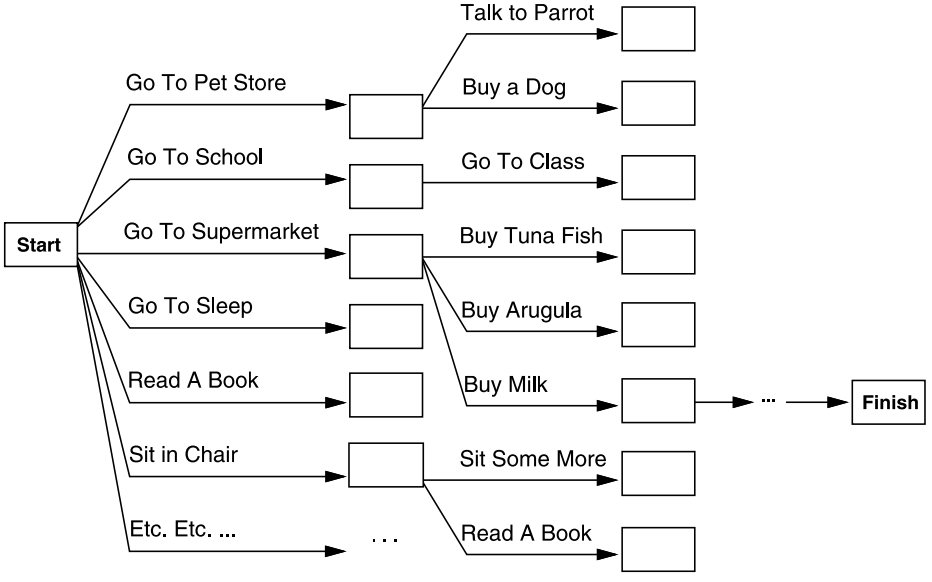
خصوصیات محیط‌های طرح‌ریزی کلاسیک

مشاهده‌پذیر کامل *Fully Observable*	مشاهده‌پذیر جزئی *Partially Observable*
تک عاملی *Single-agent*	چند عاملی *Multiagent*
قطعی *Deterministic*	اتفاقی *Stochastic*
مقطعی *Episodic*	**دنباله‌ای** *Sequential*
ایستا *Static*	پویا *Dynamic*
گسسته *Discrete*	پیوسته *Continuous*
شناخته‌شده *Known*	ناشناخته *Unknown*

طرح‌ریزی

مثال

مسئله : تهیه‌ی شیر، موز، و یک مته‌ی بی‌سیم



به نظر می‌رسد الگوریتم‌های جستجوی استاندارد به طرز نامناسبی شکست می‌خورند.

ناکافی بودن هیوریستیک‌ها و آزمون هدف After-the-fact

مقایسه‌ی «جستجو» با «طرح‌ریزی»

عامل‌های حل مسئله با استفاده از جستجو

کنش‌های مربوط *Relevant Actions*	کدام کنش‌ها **مربوط** هستند؟ تعریف فضای حالت قبلی این را روشن نمی‌کند!
تابع هیوریستیک خوب *Good Heuristic Function*	یک تابع هیوریستیک **خوب** چیست؟ باید توسط یک انسان برای هر مورد تهیه شود!
تجزیه‌ی مسئله *Problem Decomposition*	چگونه باید مسئله را **تجزیه** کرد؟ بیشتر مسائل دنیای واقعی تقریباً تجزیه‌پذیر هستند، اما تعریف فضای حالت هیچ‌گونه ساختاری از مسئله را آشکار نمی‌کند!

مقایسه‌ی «جستجو» با «طرح‌ریزی»

آنچه برای عامل طرح‌ریزی کننده لازم است

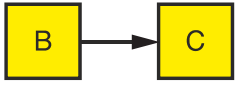
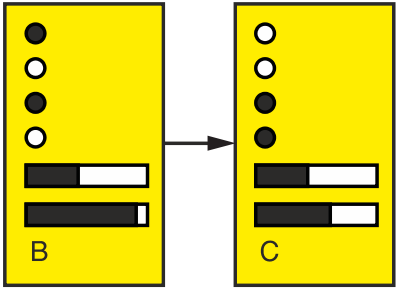
کنش‌های مربوط *Relevant Actions*	نیاز به تعامل بین **کنش‌ها** و **حالت‌ها** کنش‌ها، نیازمندی‌ها و پی‌آمدهایی دارند که به‌کارگیری آنها در یک حالت را محدود می‌کند.
تابع هیوریستیک خوب *Good Heuristic Function*	ضرورت وجود هیوریستیک‌های **معطوف به هدف** برای ایجاد طرح‌های معطوف به هدف
تجزیه‌ی مسئله *Problem Decomposition*	ضرورت حل زیرهدف‌ها به صورت **مستقل** بیشتر اجزای دنیا از بیشتر سایر اجزای آن مستقل است.

مقایسه‌ی «جستجو» با «طرح‌ریزی»

	جستجوی استاندارد *Standard Search*	طرح‌ریزی *Planning*
حالت‌ها *States*	ساختمان داده‌ها	جمله‌های منطقی
کنش‌ها *Actions*	کد	دارای پیش‌شرط‌ها / برآمدها
هدف *Goal*	کد	جمله‌های منطقی
طرح *Plan*	دنباله‌ای از حالت آغازین	دنباله‌ای با قیدهایی روی کنش‌ها

مسئله‌ی طرح‌ریزی

حالت

جستجوی استاندارد	مسئله‌ی طرح‌ریزی
	
(a) Atomic	(b) Factored
حالت: یک جعبه‌ی سیاه	**حالت:** برداری از مقادیر خصیصه‌ها
آزمون هدف: تعیین هدف بودن یا هدف نبودن یک حالت (گره): ضمنی یا صریح	**آزمون هدف:** به صورت جمله‌های منطقی

زبان طرح‌ریزی

PLANNING LANGUAGE

PDDL

Planning Domain Definition Language

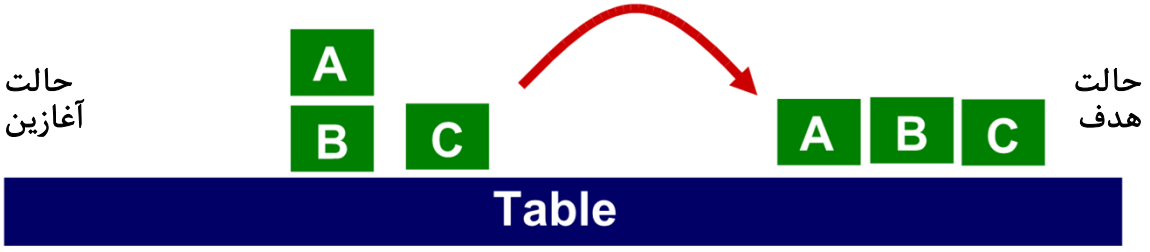
زبان تعریف دامنه‌ی طرح‌ریزی

آنچه در زبان طرح‌ریزی بیان می‌شود			
حالت آغازین *Initial State*	کنش‌های موجود *Available Actions*	نتیجه‌ی کنش‌ها *Results of Actions*	آزمون هدف *Goal Test*

ویژگی‌های لازم برای یک زبان طرح‌ریزی	
رسایی کافی *Expressive Enough*	محدودیت کافی *Restrictive Enough*
رسایی کافی توصیف انواع گسترده‌ای از مسائل	محدودیت کافی برای کارآمدی الگوریتم‌های مربوط

دنیای بلوک‌ها

BLOCKS WORLD



برای بازنمایی محیط به نام‌های محمول زیر نیاز داریم:

$On(x, y)$	obj x on top of obj y
$OnTable(x)$	obj x is on the table
$Clear(x)$	nothing is on top of obj x
$Holding(x)$	arm is holding x
$ArmEmpty$	robot arm is empty

بازنمایی FOL برای دنیای بلوک‌ها مطابق شکل بالا:

$$Clear(A) \wedge Clear(C) \wedge On(A, B) \wedge OnTable(B) \wedge OnTable(C) \wedge ArmEmpty$$

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی حالت‌ها

PDDL: Representation of States

PDDL	
حالت	عطف اتم‌های زمینی و آزاد از تابع

State = conjunction of ground and function free atoms
– e.g. $At(A, B)$, $Clear(C)$, ...
But not $At(A, x)$ or $At(neighbour(A), B)$!

فرض دنیای بسته
Closed-World Assumption

هر اتمی که در حالت نیامده باشد، **false** فرض می‌شود.

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی هدف

PDDL: Representation of Goal

PDDL	
هدف	یک حالت خاص

A goal is a particular state
– e.g. $OnTable(A) \wedge OnTable(B) \wedge OnTable(C)$

یک حالت s یک هدف g را ارضا می‌کند،
اگر
s شامل همه‌ی اتم‌های g (و شاید اتم‌های دیگر) باشد.

$Rich \wedge Famous \wedge Miserable$ satisfies $Rich \wedge Famous$

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها

PDDL: Representation of Actions

PDDL	
کنش‌ها	هر کنش دارای یک نام، پیش‌شرط و اثر می‌باشد.

کنش *Action*	**نام** *Name*	نام (می‌تواند دارای آرگومان باشد)	
	پیش‌شرط *pre-condition*	لیست واقعیت‌هایی که باید **true** باشند تا کنش بتواند اجرا شود.	
	اثر *effect*	تغییرات حاصل از اجرای کنش	
		لیست حذف *Delete List*	لیترال‌های منفی: واقعیت‌هایی که پس از انجام کنش دیگر **true** نیستند.
		لیست اضافه *Add List*	لیترال‌های مثبت: واقعیت‌هایی که پس از انجام کنش **true** می‌شوند.

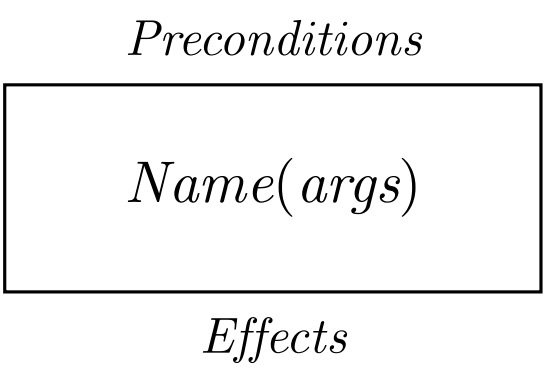
هر بخش کنش، می‌تواند حاوی متغیرها باشد.

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: نمایش در نمودار

PDDL: Representation of Actions

PDDL	
کنش‌ها	هر کنش دارای یک نام، پیش‌شرط و اثر می‌باشد.



زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: نمایش کد

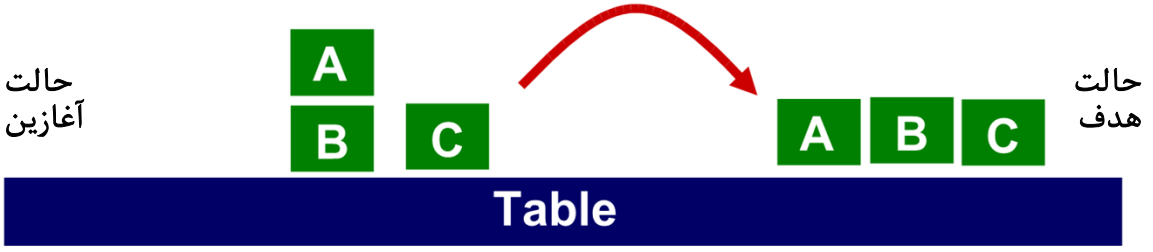
PDDL: REPRESENTATION OF ACTIONS

PDDL	
کنش‌ها	هر کنش دارای یک نام، پیش‌شرط و اثر می‌باشد.

Action($Name(args)$,
PRECOND: $Preconditions$
EFFECT: $Effects$)

دنیای بلوک‌ها

BLOCKS WORLD



برای بازنمایی محیط به نام‌های محمول زیر نیاز داریم:

$On(x, y)$	obj x on top of obj y
$OnTable(x)$	obj x is on the table
$Clear(x)$	nothing is on top of obj x
$Holding(x)$	arm is holding x
$ArmEmpty$	robot arm is empty

بازنمایی FOL برای دنیای بلوک‌ها مطابق شکل بالا:

$$Clear(A) \wedge Clear(C) \wedge On(A, B) \wedge OnTable(B) \wedge OnTable(C) \wedge ArmEmpty$$

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: مثال

کنش $Stack(x,y)$

بازوی روبات شیئ x را که در دست دارد، روی شیئ y قرار می‌دهد.

$$\begin{aligned}&\text{Action}(Stack(x,y),\\ &\quad \text{PRECOND: } Clear(y) \wedge Holding(x)\\ &\quad \text{EFFECT: } \neg Clear(y) \wedge \neg Holding(x) \wedge ArmEmpty \wedge On(x,y))\end{aligned}$$

$$Clear(y) \wedge Holding(x)$$

$$\boxed{Stack(x,y)}$$

$$\neg Clear(y) \wedge \neg Holding(x) \wedge ArmEmpty \wedge On(x,y)$$

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: مثال

کنش $UnStack(x,y)$
بازوی روبات شیئ x را از روی شیئ دیگر y برمی‌دارد.

Action($UnStack(x, y)$,
PRECOND: $On(x, y) \wedge Clear(x) \wedge ArmEmpty$
EFFECT: $\neg On(x, y) \wedge \neg ArmEmpty \wedge \neg Clear(x)$
$\wedge Holding(x) \wedge Clear(y)$)

$Stack$ and $UnStack$ are inverses of one-another.

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: مثال

کنش $Pickup(x)$
بازوی روبات شیئ x را از روی میز برمی‌دارد.

$$\begin{aligned}&\text{Action}(Pickup(x),\\ &\quad \text{PRECOND: } Clear(x) \wedge OnTable(x) \wedge ArmEmpty\\ &\quad \text{EFFECT: } \neg Clear(x) \wedge \neg OnTable(x) \wedge \neg ArmEmpty \wedge Holding(x))\end{aligned}$$

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: مثال

کنش $PutDown(x)$
بازوی روبات شیئ x را روی میز می‌گذارد.

$$\begin{aligned}&\text{Action}(PutDown(x),\\ &\quad \text{PRECOND: } Holding(x)\\ &\quad \text{EFFECT: } \neg Holding(x) \wedge OnTable(x) \wedge Clear(x) \wedge ArmEmpty)\end{aligned}$$

زبان تعریف دامنه‌ی طرح‌ریزی

قابلیت به‌کارگیری کنش‌ها

APPLICABILITY OF ACTIONS

یک کنش در حالتی قابل به‌کارگیری است که
در آن پیش‌شرط‌هایش ارضا شده باشند.

اگر کنش حاوی متغیرها بود، کنش ابتدا نمونه‌سازی می‌شود.

زبان تعریف دامنه‌ی طرح‌ریزی

قابلیت به‌کارگیری کنش‌ها: مثال

APPLICABILITY OF ACTIONS

State:

$$Clear(A) \wedge Clear(C) \wedge On(A, B) \wedge OnTable(B) \wedge OnTable(C) \wedge ArmEmpty$$

Action:

Action($Pickup(x)$,
PRECOND: $Clear(x) \wedge OnTable(x) \wedge ArmEmpty$
EFFECT: $\neg Clear(x) \wedge \neg OnTable(x) \wedge \neg ArmEmpty \wedge Holding(x)$)

Instantiation, $\{x/C\}$

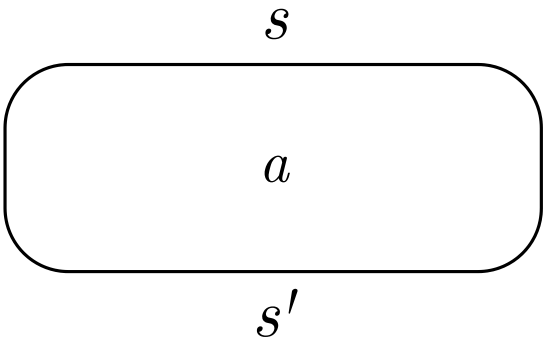
Action($Pickup(C)$,
PRECOND: $Clear(C) \wedge OnTable(C) \wedge ArmEmpty$
EFFECT: $\neg Clear(C) \wedge \neg OnTable(C) \wedge \neg ArmEmpty \wedge Holding(C)$)

زبان تعریف دامنه‌ی طرح‌ریزی

اثر به‌کارگیری کنش‌ها

EFFECT OF ACTIONS

اثر اجرای کنش a در حالت s یک حالت s' است:



s' همان s است، به‌جز:

- هر اتم از لیست اضافه‌ی کنش، به s' اضافه می‌شود.
- هر اتم از لیست حذف کنش، از s' حذف می‌شود.
- مقدار سایر اتم‌ها تغییر نمی‌کند.

زبان تعریف دامنه‌ی طرح‌ریزی

اثر به‌کارگیری کنش‌ها: مثال

EFFECT OF ACTIONS

State:

$$Clear(A) \wedge Clear(C) \wedge On(A, B) \wedge OnTable(B) \wedge OnTable(C) \wedge ArmEmpty$$

Action:

Action($Pickup(x)$,
PRECOND: $Clear(x) \wedge OnTable(x) \wedge ArmEmpty$
EFFECT: $\neg Clear(X) \wedge \neg OnTable(x) \wedge \neg ArmEmpty \wedge Holding(x)))$

Result $\{x/C\}$:
$Clear(A) \wedge On(A, B) \wedge OnTable(B) \wedge Holding(C)$

طرح

PLAN

طرح خطی *Linear Plan*	**دنباله‌ای از کنش‌ها** برای دست‌یابی به یک **هدف** (با متغیرهای جایگزین‌شده با ثابت‌ها)
طرح غیرخطی *Non-linear Plan*	**درخت/گرافی از کنش‌ها** برای دست‌یابی به یک **هدف** (طرح‌های غیرخطی: حاوی شرط و حلقه)

❋ بیشتر طرح‌های زندگی واقعی، غیرخطی هستند؛
اما اکثر طرح‌ریزی‌کننده‌های ساده نمی‌توانند این ساختارها را اداره کنند!

هوش مصنوعی

طرح‌ریزی کلاسیک

۲

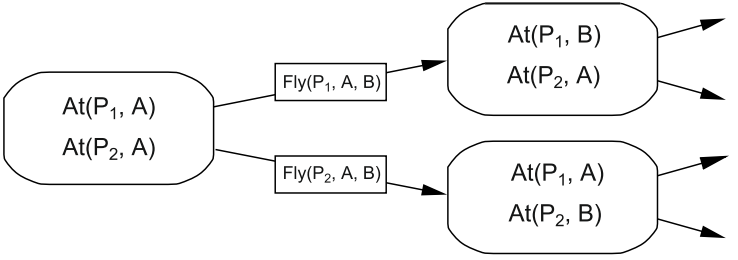
الگوریتم‌های طرح‌ریزی در قالب جستجوی فضای حالت

طرح‌ریزی در قالب جستجوی فضای حالت

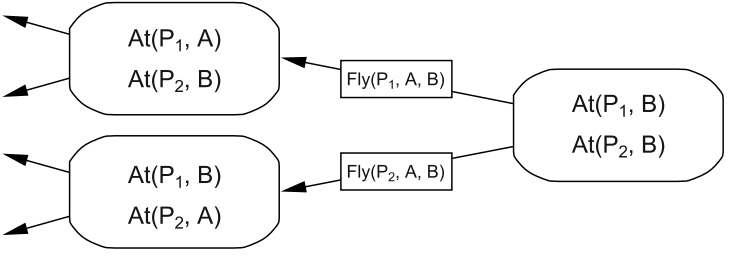
PLANNING AS STATE-SPACE SEARCH

طرح‌ریزی در قالب جستجوی فضای حالت	
طرح‌ریزی‌کننده‌ی پیش‌رونده *Progressive (Forward) Planner*	**طرح‌ریزی‌کننده‌ی پس‌رونده** *Regressive (Backward) Planner*
با جستجوی پیش‌روی فضای حالت	با جستجوی پس‌روی فضای حالت

پیش‌رونده



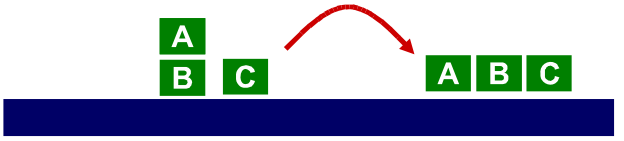
پس‌رونده

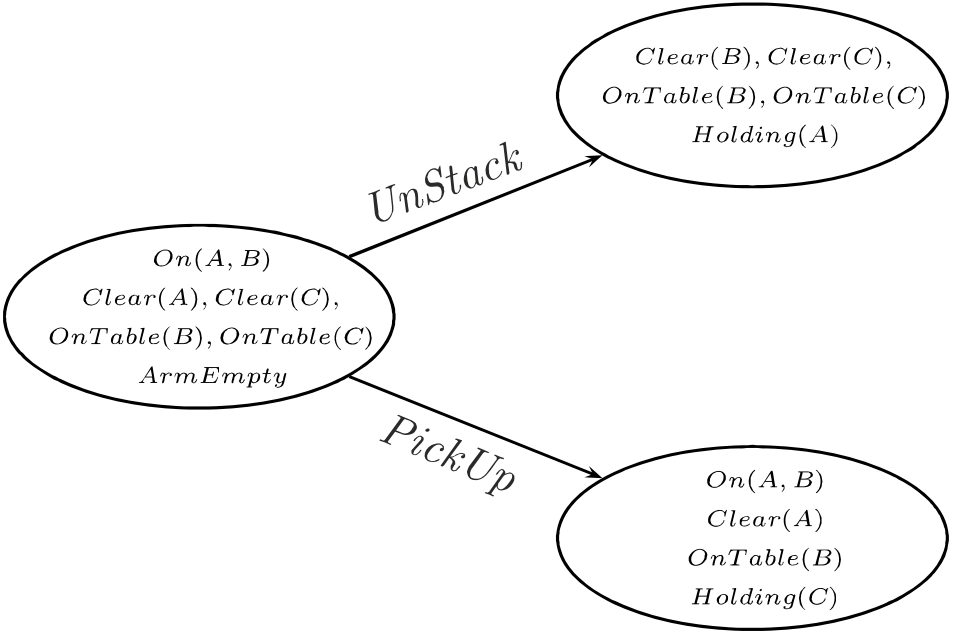


چالش: ایجاد هیوریستیک‌های مستقل از دامنه

طرح‌ریزی پیش‌رونده

مثال: دنیای بلوک‌ها





هوش مصنوعی

طرح‌ریزی کلاسیک

۳

طرح‌ریزی ترتیب جزئی

رابطه‌ی ترتیب در طرح‌ریزی

طرح‌ریزی	
طرح‌ریزی ترتیب کلی *Total-Order Planning*	**طرح‌ریزی ترتیب جزئی** *Partial-Order Planning*
طرح: ترتیب کلی از کنش‌ها	طرح: ترتیب جزئی از کنش‌ها
در نظر نگرفتن استقلال زیرهدف‌ها	استفاده از استقلال زیرهدف‌ها

طرح‌ریزی‌کننده‌ی پیش‌رونده
Progressive (Forward) Planner

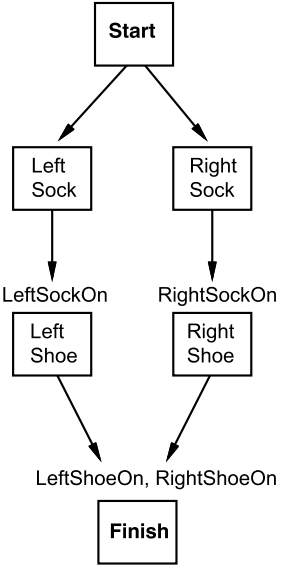
طرح‌ریزی‌کننده‌ی پس‌رونده
Regressive (Backward) Planner

رابطه‌ی ترتیب در طرح‌ریزی

مثال: مسئله‌ی پوشیدن جوراب و کفش

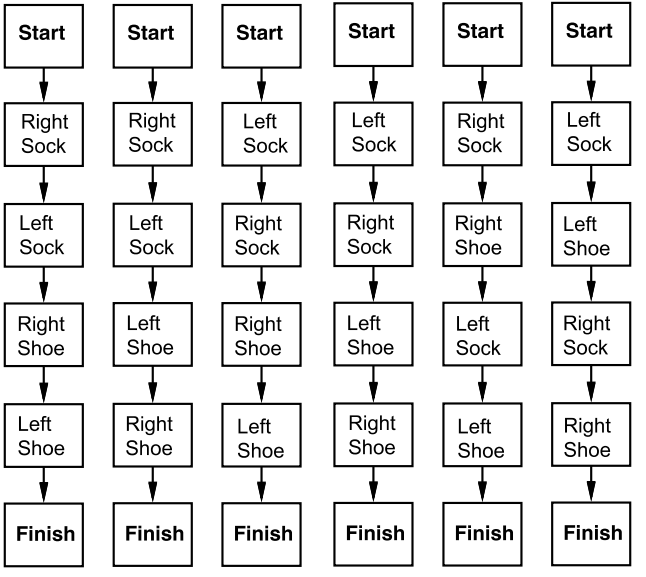
طرح‌ریزی ترتیب جزئی
Partial-Order Planning

Partial-Order Plan:



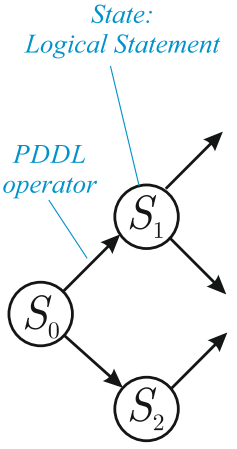
طرح‌ریزی ترتیب کلی
Total-Order Planning

Total-Order Plans:



طرح‌ریزی با جستجو

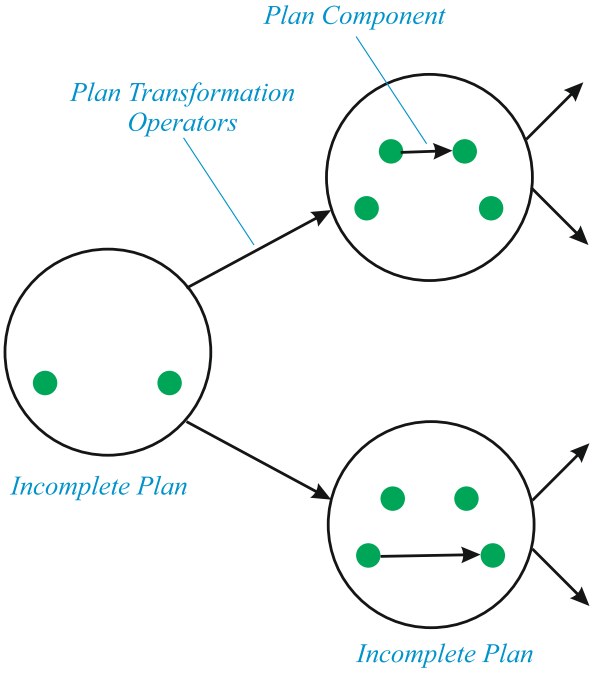
جستجو در فضای حالت / جستجو در فضای طرح



State-Space Search

جستجوی در فضای حالت:

گره = حالت انضمامی دنیا



Plan-Space Search

جستجوی در فضای طرح:

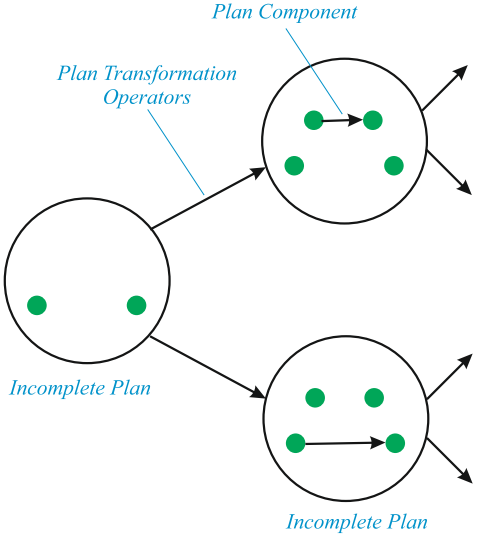
گره = طرح جزئی

طرح‌ریزی ترتیب جزئی

PARTIAL-ORDER PLANNING

طرح‌ریزی ترتیب جزئی
Partial-Order Planning (POP)

الگوریتم: به‌هنگام‌سازی گام به گام **طرح‌های جزئی**



Plan-Space Search

طرح‌ریزی ترتیب جزئی

طرح جزئی

PARTIAL-ORDER PLANNING

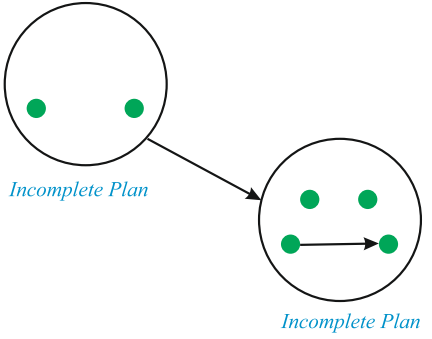
اجزای یک طرح (جزئی)		
	کنش‌ها Actions	مجموعه‌ای از شماهای عملگر «گام‌ها» برای مسئله S_i
	قیدهای ترتیب‌دهی Ordering Constraints	قیدهای ترتیب جزئی (موقتی) $S_i \prec S_j$
	پیوندهای علّی Causal Links	رابطه‌ی علی بین دو کنش $S_i \xrightarrow{c} S_j$: S_i به c دست می‌یابد که پیش‌شرط S_j است. (مقصود از گام‌ها را ثبت می‌کند.)
	پیش‌شرط‌های باز Open Preconditions	پیش‌شرطی از یک گام که هنوز پیوند علّی پیدا نکرده است.

طرح‌ریزی ترتیب جزئی

عملگرهای تبدیل طرح جزئی

PARTIAL-ORDER PLANNING

عملیات بر روی طرح‌های جزئی	
اضافه کردن یک پیوند علّی	از یک کنش موجود به یک پیش‌شرط باز
اضافه کردن یک گام	برای دست‌یابی به یک پیش‌شرط باز
ترتیب‌دهی یک گام	ترتیب‌دهی یک گام نسبت به دیگران (برای حذف تداخل‌های احتمالی)



حرکت تدریجی از طرح‌های ناکامل/مبهم به طرح‌های کامل و درست

عقب‌گرد: در صورتی که یک پیش‌شرط باز **دست‌نیافتنی** باشد یا یک تداخل **رفع‌نشدنی** باشد.

طرح‌ریزی ترتیب جزئی

گام‌ها

PARTIAL-ORDER PLANNING

گام شروع
Start Step

گام $Start$ دارای توصیف حالت آغازین در قالب اثر آن

پیوندهای علّی از برآمد یک گام به پیش‌شرط گام دیگر ایجاد می‌شوند.

بین جفت گام‌ها ترتیب‌دهی موقتی صورت می‌گیرد.

گام پایان
Finish Step

گام $Finish$ دارای توصیف هدف در قالب پیش‌شرط آن

یک طرح زمانی **کامل** است که همه‌ی پیش‌شرط‌های آن **دست‌یابی** شده باشد.

یک پیش‌شرط **دست‌یابی** شده است
هرگاه
اثر یک گام قبلی‌تر باشد و هیچ گامی با امکان تداخل، آن را برگردان نکند.

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۱ از ۸)

SPARE TIER PROBLEM

We want to change a flat tire.

INITIAL STATE: $At(Flat, Axle) \wedge At(Spare, Boot)$
GOAL STATE: $At(Spare, Axle)$
ACTIONS (definition follows):
- $Remove(Spare, Trunk)$
- $Remove(Flat, Axle)$
- $PutOn(Spare, Axle)$
- $LeaveOvernight$

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۲ از ۸)

SPARE TIER PROBLEM

Action($Remove(Spare, Trunk)$,
PRECOND: $At(Spare, Trunk)$
EFFECT: $\neg At(Spare, Trunk) \wedge At(Spare, Ground)$)

Action($Remove(Flat, Axle)$,
PRECOND: $At(Flat, Axle)$
EFFECT: $\neg At(Flat, Axle) \wedge At(Flat, Ground) \wedge ClearAxle$)

Action($PutOn(Spare, Axle)$,
PRECOND: $At(Spare, Ground) \wedge ClearAxle$
EFFECT: $\neg At(Spare, Ground) \wedge \neg ClearAxle \wedge At(Spare, Axle)$)

Action($LeaveOvernight$,
PRECOND:
EFFECT: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
$\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle) \wedge ClearAxle$)

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۳ از ۸)

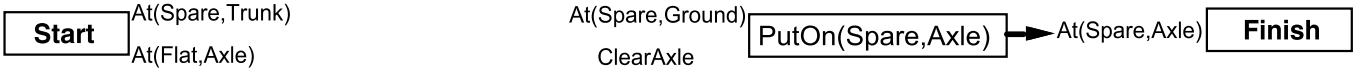
SPARE TIER PROBLEM

Initial plan: Start with effects and Finish with preconditions

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۴ از ۸)

SPARE TIER PROBLEM



Pick an open precondition $At(Spare, Axle)$

Only $PutOn(Spare, Axle)$ achieves that

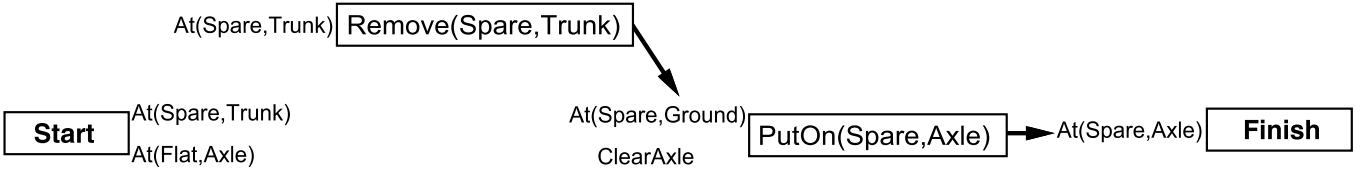
Add link $PutOn(Spare, Axle) \xrightarrow{At(Spare,Axle)} Finish$

Add constraint $PutOn(Spare, Axle) < Finish$
- $PutOn(Spare, Axle)$ is on the left

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۵ از ۸)

SPARE TIER PROBLEM



Pick an open precondition $At(Spare, Ground)$

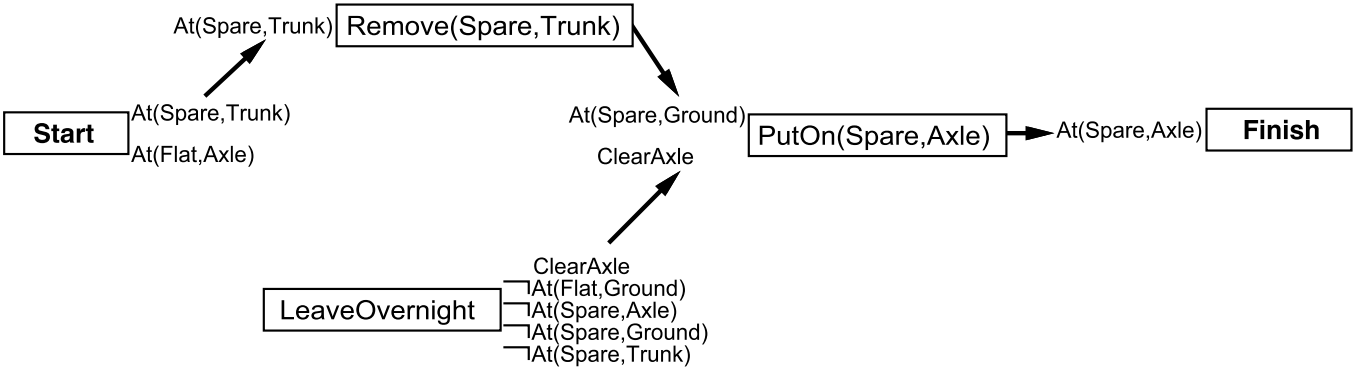
Only $Remove(Spare, Trunk)$ achieves that

Add link $Remove(Spare, Trunk) \xrightarrow{At(Spare,Ground)} PutOn(Spare, Axle)$

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۶ از ۸)

SPARE TIER PROBLEM



Pick an open precondition $ClearAxle$

E.g. $LeaveOvernight$ can achieve that

CONFLICT: $Remove(Spare, Trunk) \xrightarrow{At(Spare,Ground)} PutOn(Spare, Axle)$

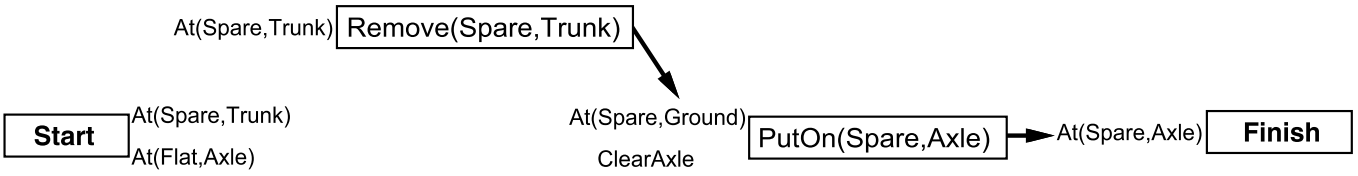
Add constraint $LeaveOvernight < Remove(Spare, Trunk)$

shift $LeaveOvernight$ further left

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۷ از ۸)

SPARE TIER PROBLEM

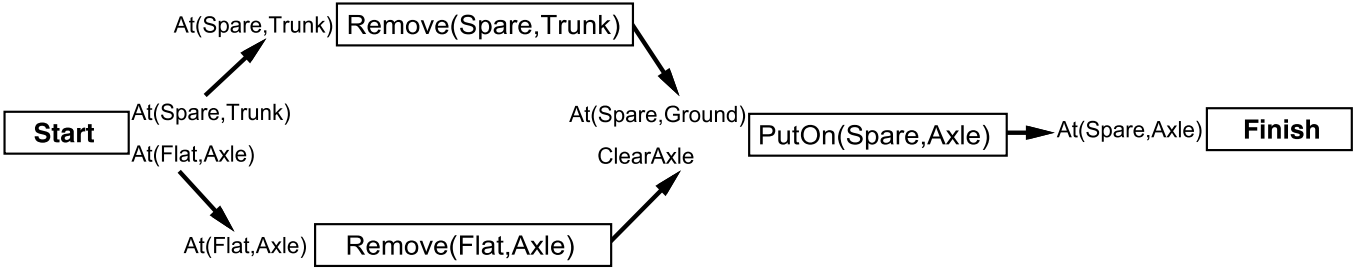


Undo …

طرح‌ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یدکی (۸ از ۸)

SPARE TIER PROBLEM



Pick an open precondition $ClearAxle$

Another possibility: $Remove(Flat, Axle)$ can achieve that

Finish the search

طرح‌ریزی ترتیب جزئی

شبه کد (۱ از ۲)

PARTIAL-ORDER PLANNING

```
function POP(initial, goal, operators) returns plan
    plan ← MAKE-MINIMAL-PLAN(initial, goal)
    loop do
        if SOLUTION?(plan) then return plan
        S_need, c ← SELECT-SUBGOAL(plan)
        CHOOSE-OPERATOR(plan, operators, S_need, c)
        RESOLVE-THREATS(plan)
    end
```

```
function SELECT-SUBGOAL(plan) returns S_need, c
    pick a plan step S_need from STEPS(plan)
        with a precondition c that has not been achieved
    return S_need, c
```

طرح‌ریزی ترتیب جزئی

شبه کد (۲ از ۲)

PARTIAL-ORDER PLANNING

procedure CHOOSE-OPERATOR($plan$, $operators$, S_{need}, c)

  choose a step S_{add} from $operators$ or STEPS($plan$) that has c as an effect
  if there is no such step **then fail**
  add the causal link $S_{add} \xrightarrow{c} S_{need}$ to LINKS($plan$)
  add the ordering constraint $S_{add} \prec S_{need}$ to ORDERINGS($plan$)
  if S_{add} is a newly added step from $operators$ **then**
    add S_{add} to STEPS($plan$)
    add $Start \prec S_{add} \prec Finish$ to ORDERINGS($plan$)

---

procedure RESOLVE-THREATS($plan$)

  for each S_{threat} that threatens a link $S_i \xrightarrow{c} S_j$ in LINKS($plan$) **do**
    choose either
      Demotion: Add $S_{threat} \prec S_i$ to ORDERINGS($plan$)
      Promotion: Add $S_j \prec S_{threat}$ to ORDERINGS($plan$)
    if not CONSISTENT($plan$) **then fail**
  end

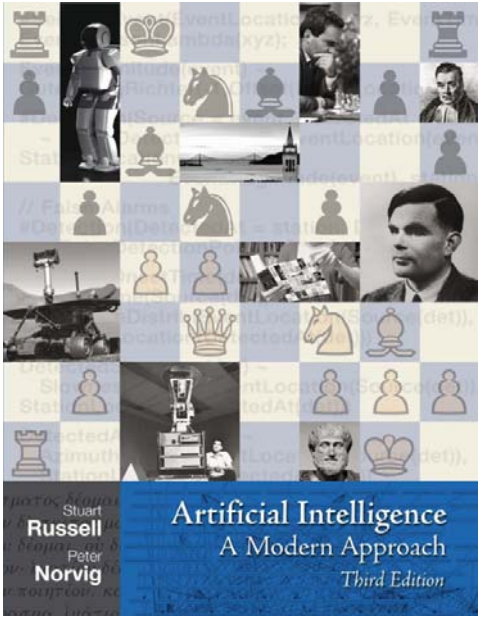
هوش مصنوعی

طرح‌ریزی کلاسیک

۳

منابع، مطالعه، تکلیف

منبع اصلی



Stuart Russell and Peter Norvig,
Artificial Intelligence: A Modern Approach,
3rd Edition, Prentice Hall, 2010.

Chapter 10

10 CLASSICAL PLANNING

In which we see how an agent can take advantage of the structure of a problem to construct complex plans of action.

We have defined AI as the study of rational action, which means that **planning**—devising a plan of action to achieve one's goals—is a critical part of AI. We have seen two examples of planning agents so far: the search-based problem-solving agent of Chapter 3 and the hybrid logical agent of Chapter 7. In this chapter we introduce a representation for planning problems that scales up to problems that could not be handled by those earlier approaches.

Section 10.1 develops an expressive yet carefully constrained language for representing planning problems. Section 10.2 shows how forward and backward search algorithms can take advantage of this representation, primarily through accurate heuristics that can be derived automatically from the structure of the representation. (This is analogous to the way in which effective domain-independent heuristics were constructed for constraint satisfaction problems in Chapter 6.) Section 10.3 shows how a data structure called the planning graph can make the search for a plan more efficient. We then describe a few of the other approaches to planning, and conclude by comparing the various approaches.

This chapter covers fully observable, deterministic, static environments with single agents. Chapters 11 and 17 cover partially observable, stochastic, dynamic environments with multiple agents.

10.1 DEFINITION OF CLASSICAL PLANNING

The problem-solving agent of Chapter 3 can find sequences of actions that result in a goal state. But it deals with atomic representations of states and thus needs good domain-specific heuristics to perform well. The hybrid propositional logical agent of Chapter 7 can find plans without domain-specific heuristics because it uses domain-independent heuristics based on the logical structure of the problem. But it relies on ground (variable-free) propositional inference, which means that it may be swamped when there are many actions and states. For example, in the wumpus world, the simple action of moving a step forward had to be repeated for all four agent orientations, T time steps, and n^2 current locations.

366