

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



هوش مصنوعی

فصل ۱۱

طرح ریزی خودکار شده

Automated Planning

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

<http://courses.fouladi.ir/ai>

هوش مصنوعی

طرح ریزی خودکار شده

۱

تعریف
طرح ریزی
کلاسیک

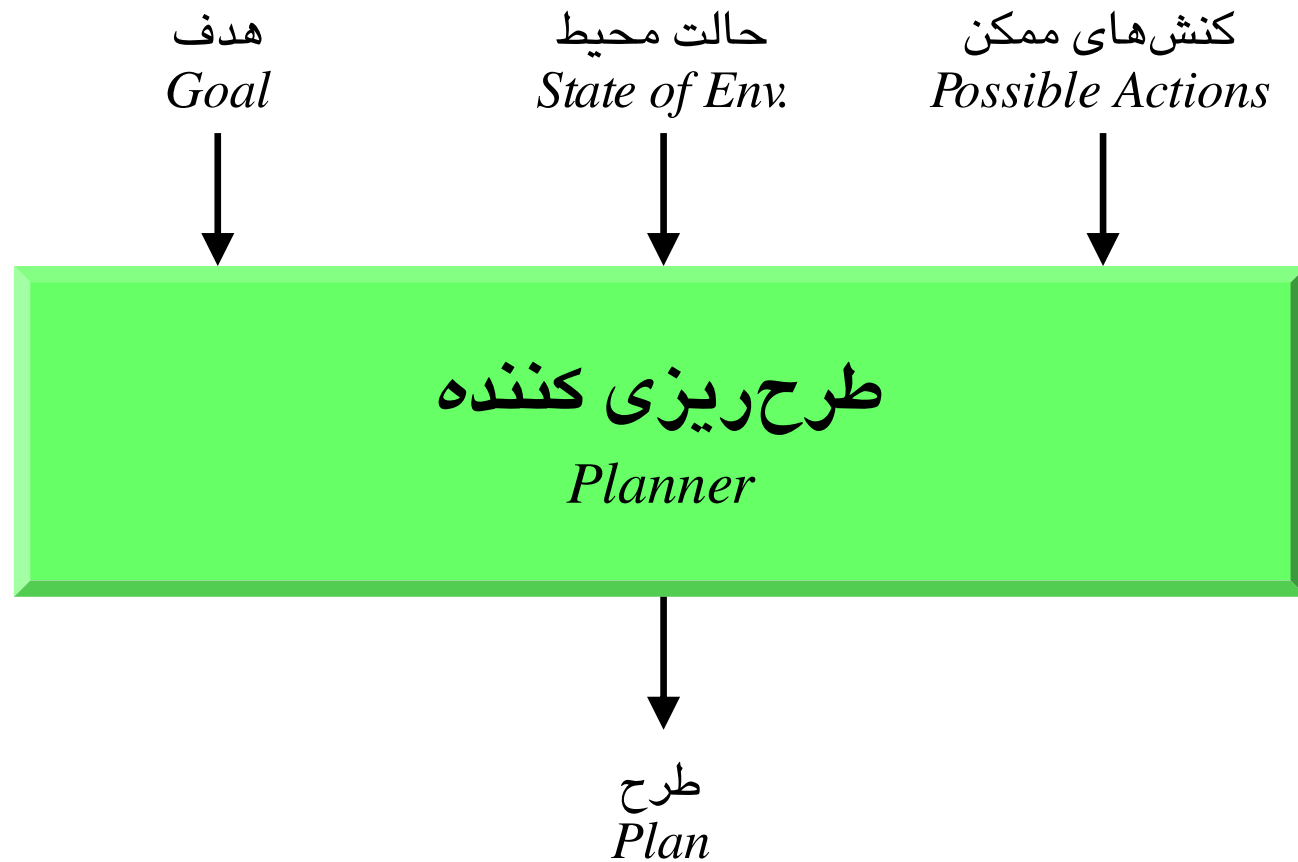
طرح ریزی

PLANNING

مسئله‌ی یافتن دنباله‌ای از کنش‌ها برای دستیابی به یک هدف

طرح ریزی
Planning

معماری طرح‌ریزی کننده

ARCHITECTURE OF A PLANNER

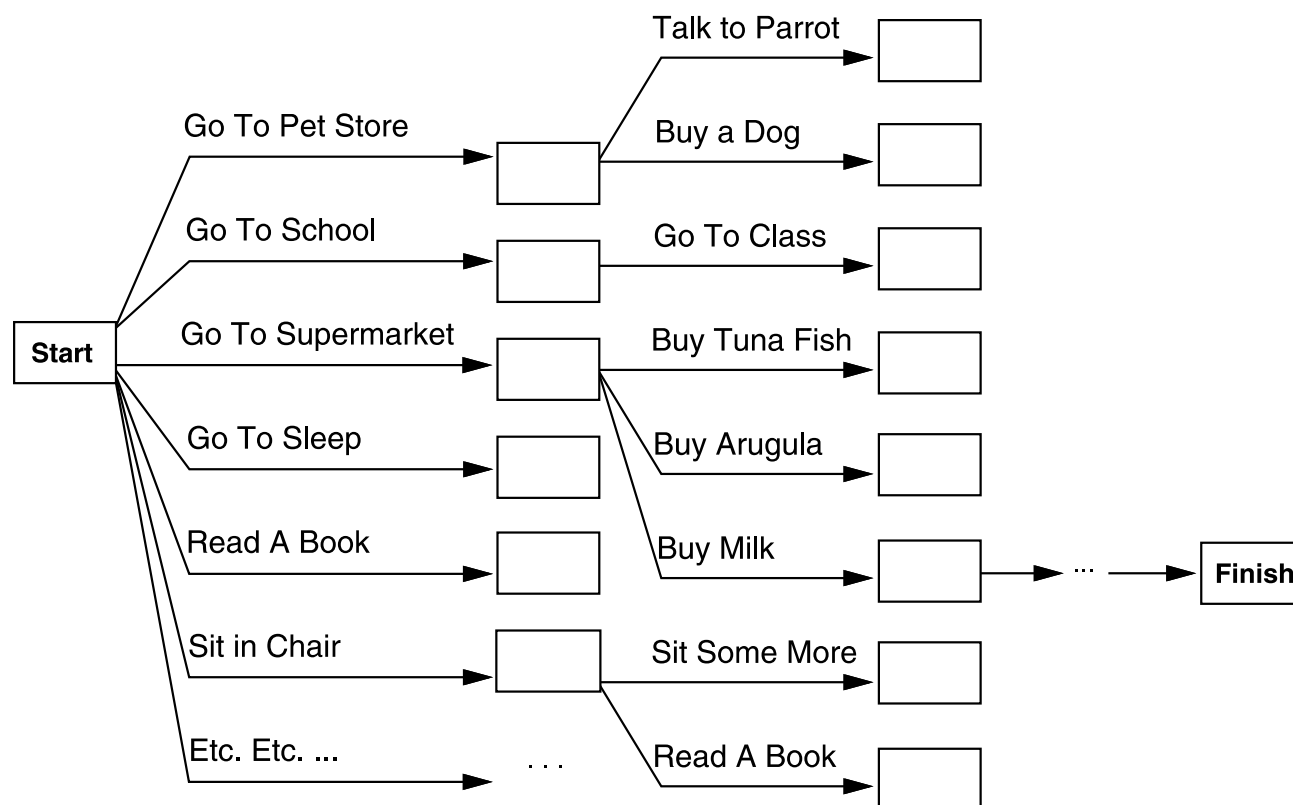
خصوصیات محیط‌های طرح‌ریزی کلاسیک

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

طرح ریزی

مثال

مسئله: تهیه شیر، موز، و یک مته بی سیم



به نظر می رسد الگوریتم های جستجوی استاندارد به طرز نامناسبی شکست می خورند.

ناکافی بودن هیوریستیک ها 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.

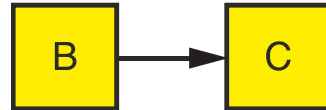
مقایسه‌ی «جستجو» با «طرح‌ریزی»

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

مسئله‌ی طرح‌ریزی

حالت

جستجوی استاندارد

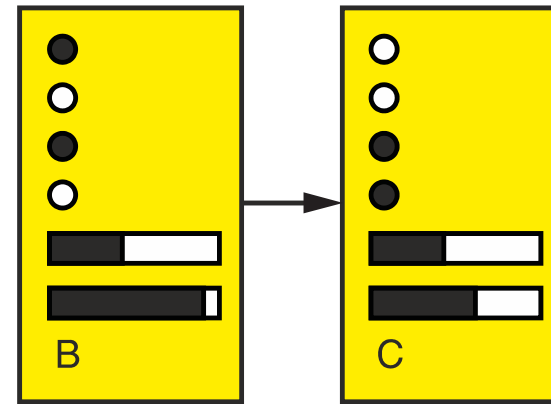


(a) Atomic

حالت: یک جعبه‌ی سیاه

آزمون هدف: تعیین هدف بودن یا هدف
نبودن یک حالت (گره): ضمنی یا صریح

مسئله‌ی طرح‌ریزی



(b) Factored

حالت: برداری از مقادیر خصیصه‌ها

آزمون هدف: به صورت جمله‌های منطقی

زبان طرح ریزی

PLANNING LANGUAGE**PDDL**

Planning Domain Definition Language

زبان تعریف دامنه‌ی طرح ریزی

آنچه در زبان طرح ریزی بیان می‌شود

آزمون هدف

Goal Test

نتیجه‌ی کنش‌ها

Results of Actions

کنش‌های موجود

Available Actions

حالت آغازین

Initial State

ویژگی‌های لازم برای یک زبان طرح ریزی

محدودیت کافی

Restrictive Enough

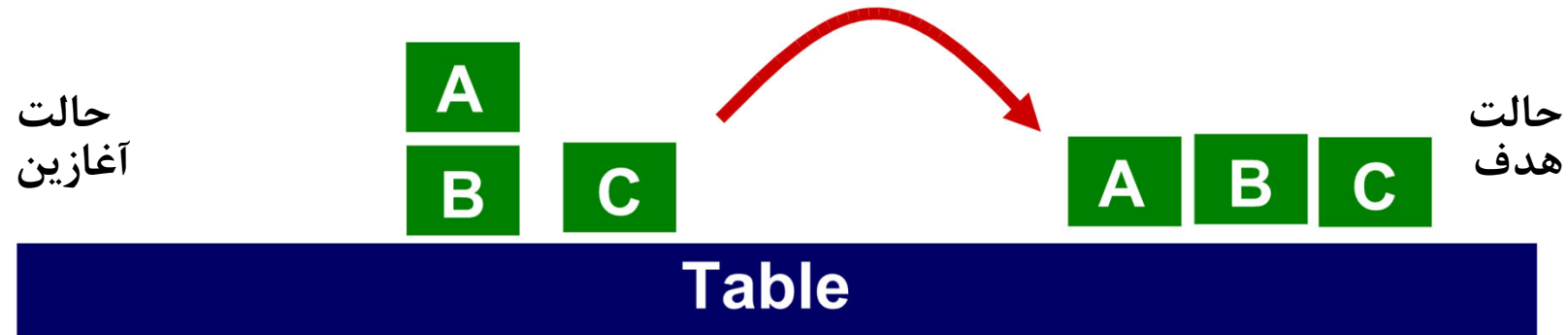
رسایی کافی

Expressive Enough

محدودیت کافی برای کارآمدی الگوریتم‌های مربوط

رسایی کافی توصیف انواع گسترده‌ای از مسائل

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	
عطف اتم‌های زمینی و آزاد از تابع	حالت

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	
یک حالت خاص	هدف

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

هر کنش دارای یک نام، پیش‌شرط و اثر می‌باشد.

کنش‌ها

Preconditions

Name(args)

Effects

زبان تعریف دامنه‌ی طرح‌ریزی

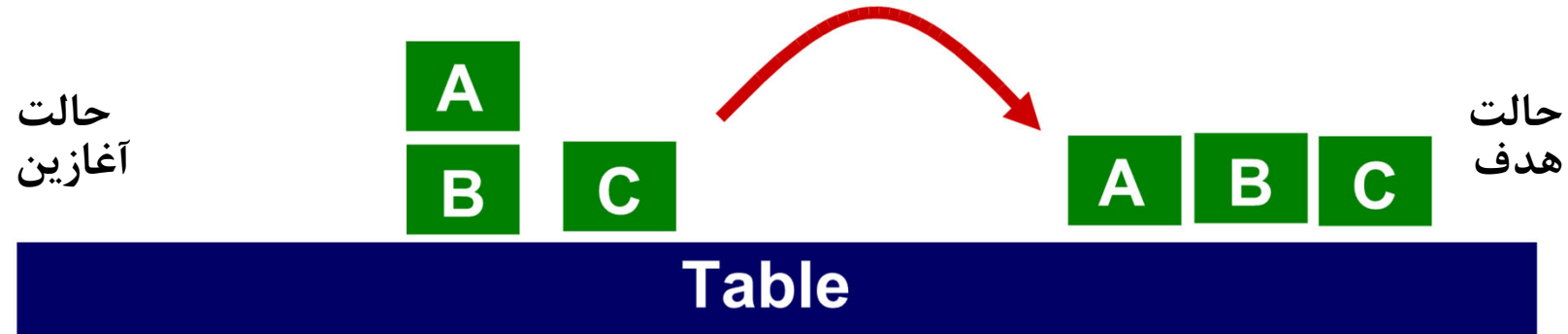
بازنمایی کنش‌ها: نمایش کد

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 شیء x را که در دست دارد، روی شیء y قرار می‌دهد.

$Action(Stack(x, y),$

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

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

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

$Stack(x, y)$

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

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: مثال

کنش $UnStack(x, y)$

بازوی روبات شیء x را از روی شیء 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 شیء x را از روی میز برمی‌دارد. $Action(Pickup(x),$ PRECOND: $Clear(x) \wedge OnTable(x) \wedge ArmEmpty$ EFFECT: $\neg Clear(x) \wedge \neg OnTable(x) \wedge \neg ArmEmpty \wedge Holding(x))$

زبان تعریف دامنه‌ی طرح‌ریزی

بازنمایی کنش‌ها: مثال

کنش $PutDown(x)$ بازوی روبات شیء x را روی میز می‌گذارد. $Action(PutDown(x),$ PRECOND: $Holding(x)$ EFFECT: $\neg Holding(x) \wedge OnTable(x) \wedge Clear(x) \wedge ArmEmpty)$

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: $Clear(C) \wedge OnTable(C) \wedge ArmEmpty$

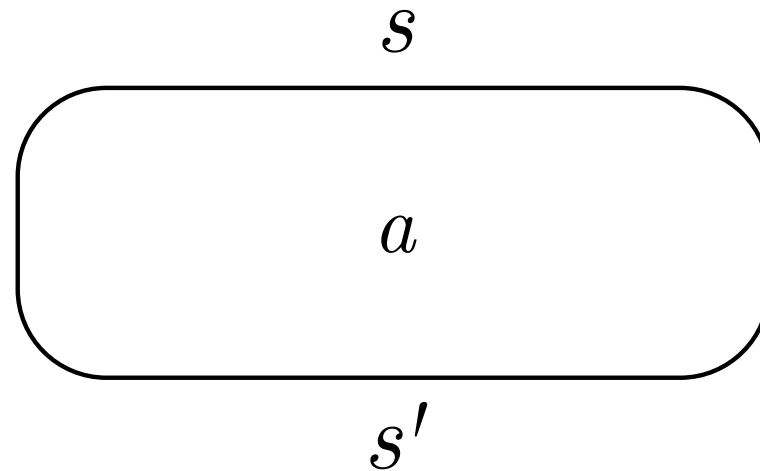
EFFECT: $\neg Clear(C) \wedge \neg OnTable(C) \wedge \neg ArmEmpty \wedge Holding(C)$)

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، به جز:
- 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 به کارگیری 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

دنباله‌ای از کنش‌ها برای دستیابی به یک هدف
(با متغیرهای جایگزین شده با ثابت‌ها)

طرح خطی
Linear Plan

درخت / گرافی از کنش‌ها برای دستیابی به یک هدف
(طرح‌های غیرخطی: حاوی شرط و حلقه)

طرح غیرخطی
Non-linear Plan

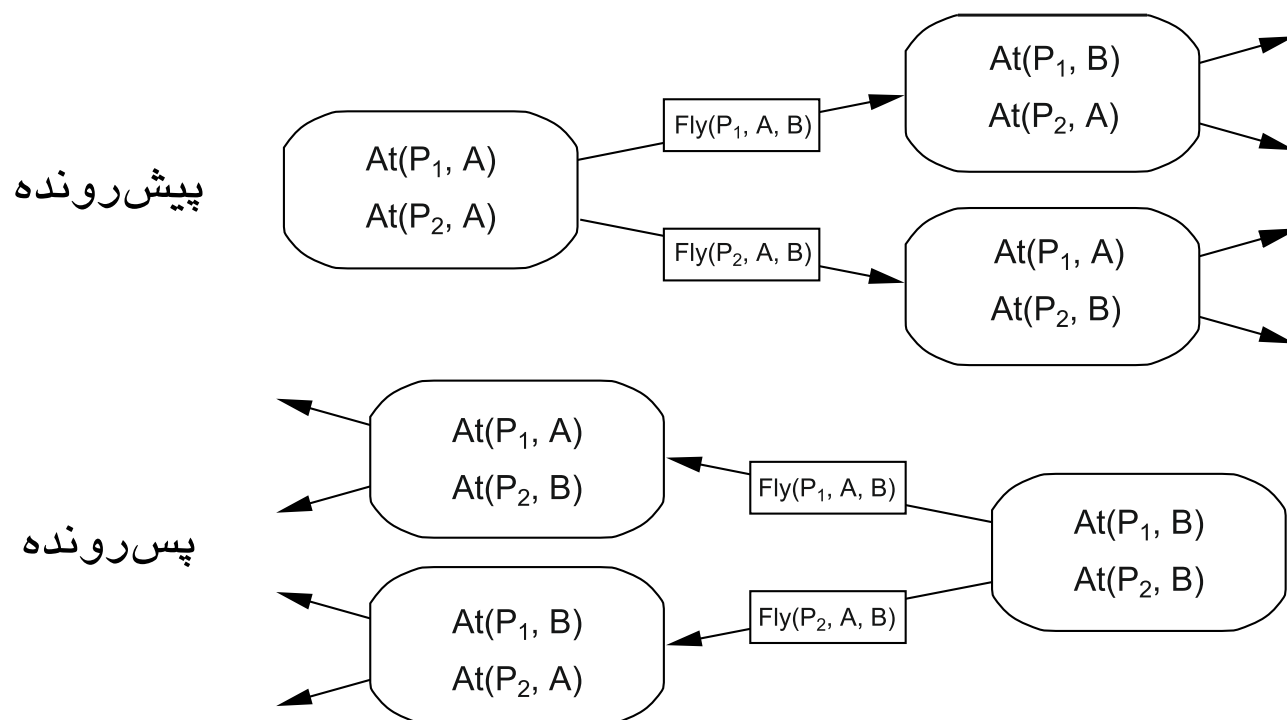
* بیشتر طرح‌های زندگی واقعی، غیرخطی هستند؛
اما اکثر طرح‌ریزی‌کننده‌های ساده نمی‌توانند این ساختارها را اداره کنند!

۲

الگوریتم‌های
طرح ریزی
در قالب
جستجوی
فضای حالت

طرح‌ریزی در قالب جستجوی فضای حالت

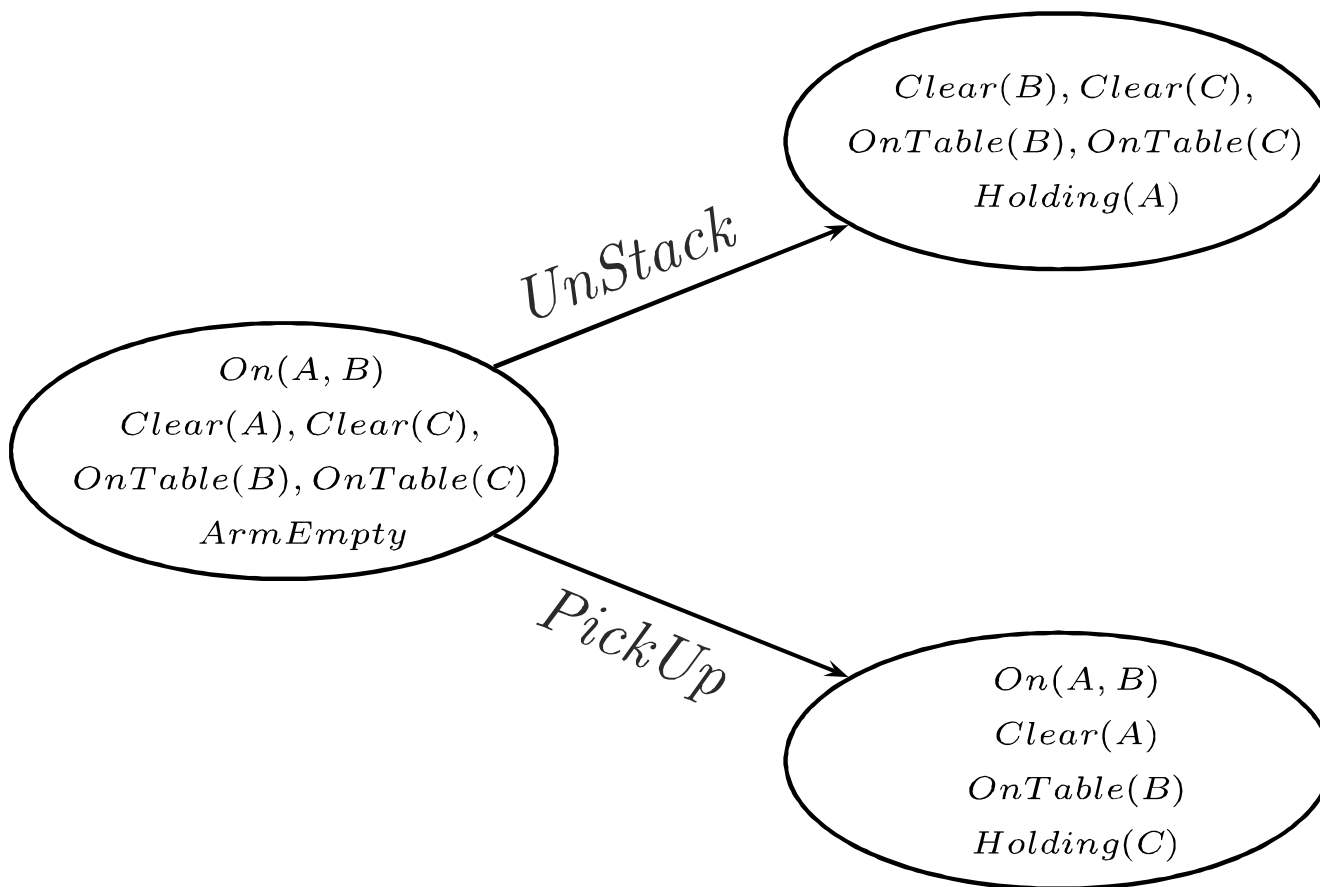
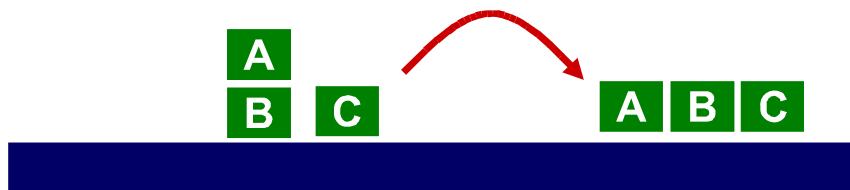
PLANNING AS STATE-SPACE SEARCH



چالش: ایجاد هیوریستیک‌های مستقل از دامنه

طرح ریزی پیش رونده

مثال: دنیای بلوکها



۳

طرح ریزی
ترتیب
جزئی

رابطه‌ی ترتیب در طرح‌ریزی

طرح‌ریزی	
طرح‌ریزی ترتیب جزئی <i>Partial-Order Planning</i>	طرح‌ریزی ترتیب کلی <i>Total-Order Planning</i>
طرح: ترتیب جزئی از کنش‌ها	طرح: ترتیب کلی از کنش‌ها
استفاده از استقلال زیرهدف‌ها	در نظر نگرفتن استقلال زیرهدف‌ها
	طرح‌ریزی‌کننده‌ی پیش‌رونده <i>Progressive (Forward) Planner</i>
	طرح‌ریزی‌کننده‌ی پس‌رونده <i>Regressive (Backward) Planner</i>

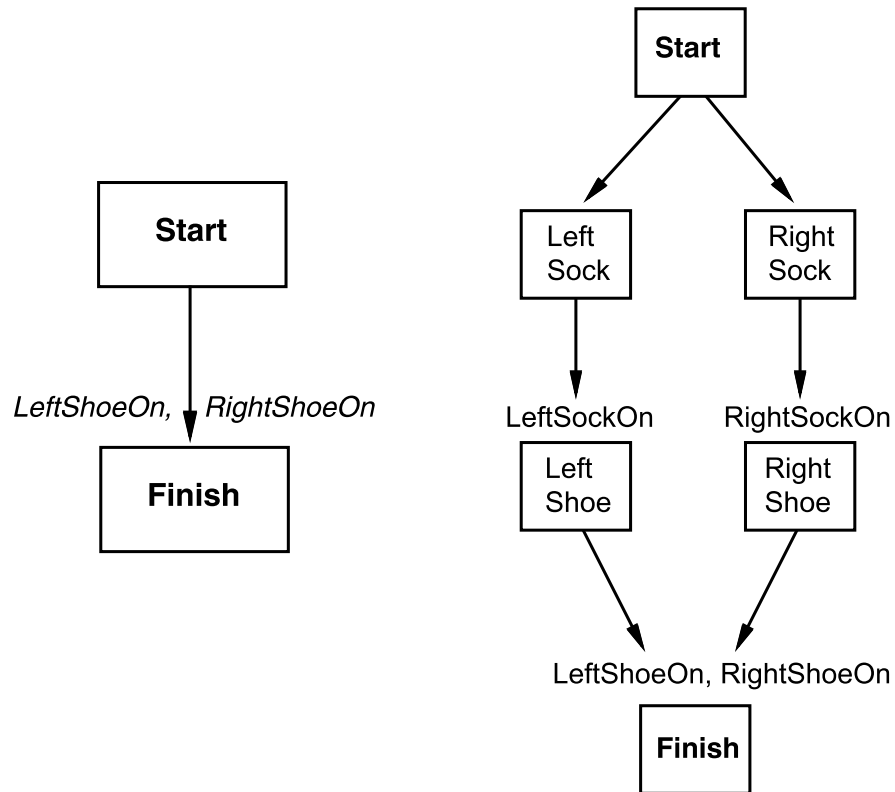
رابطه‌ی ترتیب در طرح‌ریزی

مثال: مسئله‌ی پوشیدن جوراب و کفش

طرح‌ریزی ترتیب جزئی

Partial-Order Planning

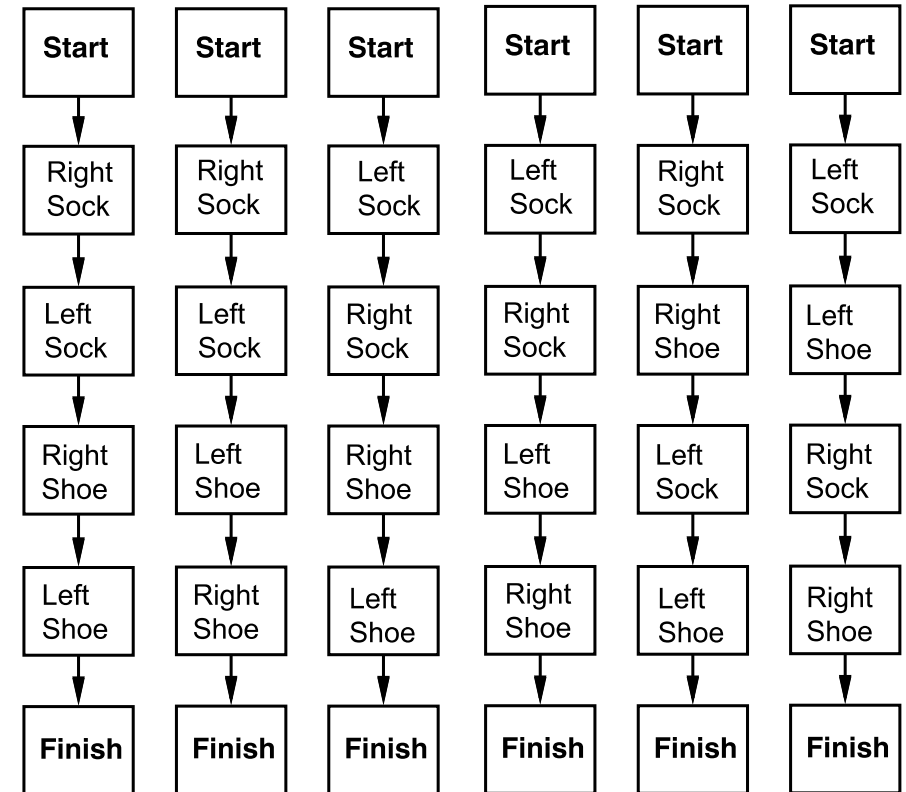
Partial-Order Plan:



طرح‌ریزی ترتیب کلی

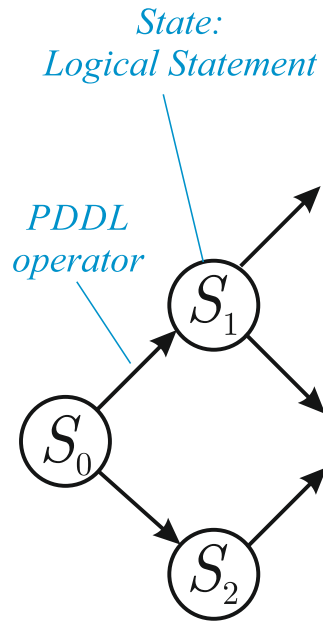
Total-Order Planning

Total-Order Plans:



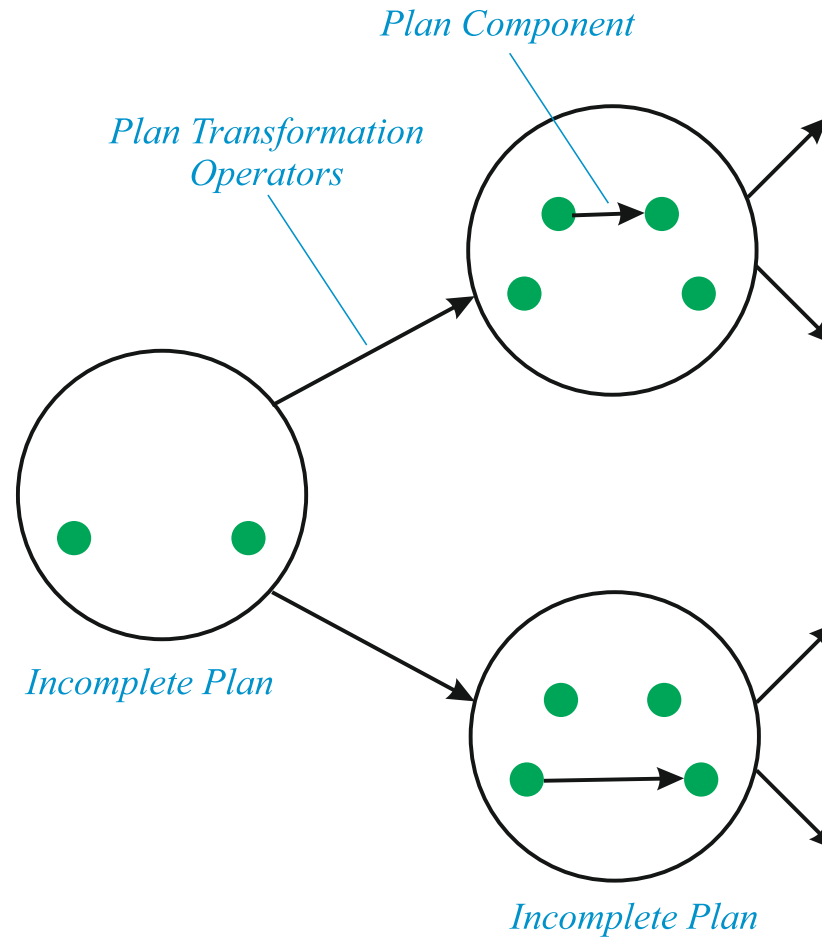
طرح ریزی با جستجو

جستجو در فضای حالت / جستجو در فضای طرح



State-Space Search

جستجوی در فضای حالت:
گره = حالت انضمامی دنیا



Plan-Space Search

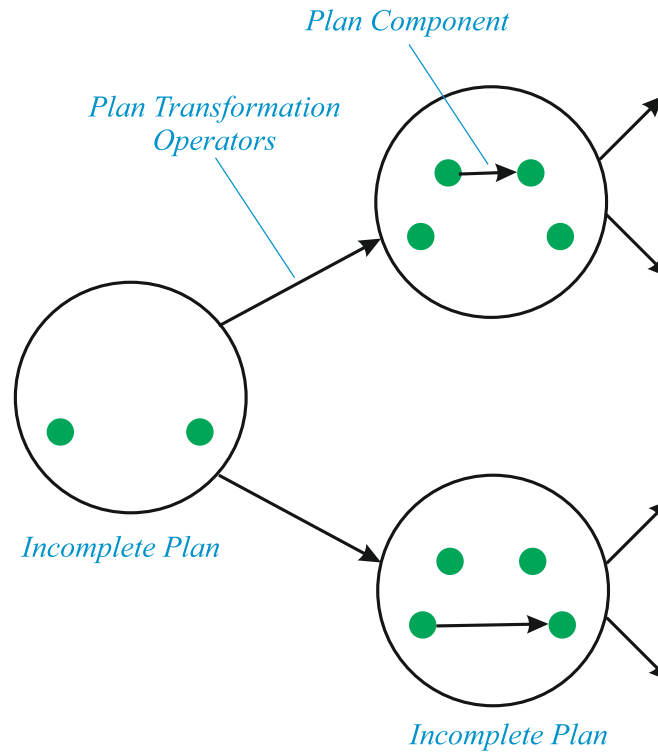
جستجوی در فضای طرح:
گره = طرح جزئی

طرح ریزی ترتیب جزئی

PARTIAL-ORDER PLANNING

الگوریتم: به هنگام سازی گام به گام طرح های جزئی

طرح ریزی ترتیب جزئی
Partial-Order Planning (POP)



Plan-Space Search

طرح ریزی ترتیب جزئی

طرح جزئی

PARTIAL-ORDER PLANNING

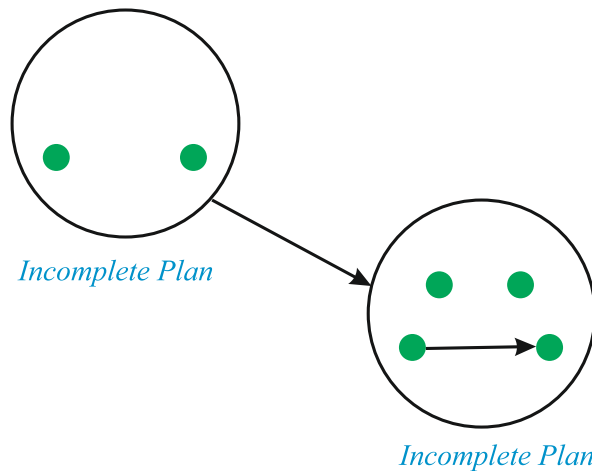
مجموعه‌ای از شماهای عملگر «گام‌ها» برای مسئله S_i	کنش‌ها Actions	اجزای یک طرح (جزئی)
قیدهای ترتیب جزئی (موقتی) $S_i < S_j$	قیدهای ترتیب‌دهی Ordering Constraints	
رابطه‌ی علی بین دو کنش $S_i \xrightarrow{c} S_j$: S_i به S_j دست می‌یابد که پیش‌شرط S_j است. (مقصود از گام‌ها را ثبت می‌کند.)	پیوندهای علی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

طرح‌ریزی ترتیب جزئی

عملگرهای تبدیل طرح جزئی

PARTIAL-ORDER PLANNING

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



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

عقب‌گرد: در صورتی که یک پیش شرط باز دست‌نیافتنی باشد یا یک تداخل رفع‌نشده باشد.

طرح‌ریزی ترتیب جزئی

گام‌ها

PARTIAL-ORDER PLANNING

گام شروع

Start Step

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

پیوندهای علیّ 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.

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایر یکنی (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):

- $Remove(Spare, Trunk)$
- $Remove(Flat, Axle)$
- $PutOn(Spare, Axle)$
- $LeaveOvernight$

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (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: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
 $\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle) \wedge ClearAxle$)

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (۳ از ۸)

SPARE TIER PROBLEM

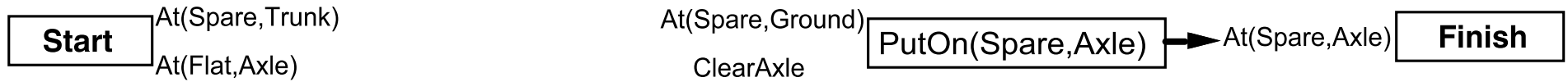


Initial plan: Start with effects and Finish with preconditions

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (۴ از ۸)

SPARE TIER PROBLEM



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

Only $PutOn(Spare, Axle)$ achieves that

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

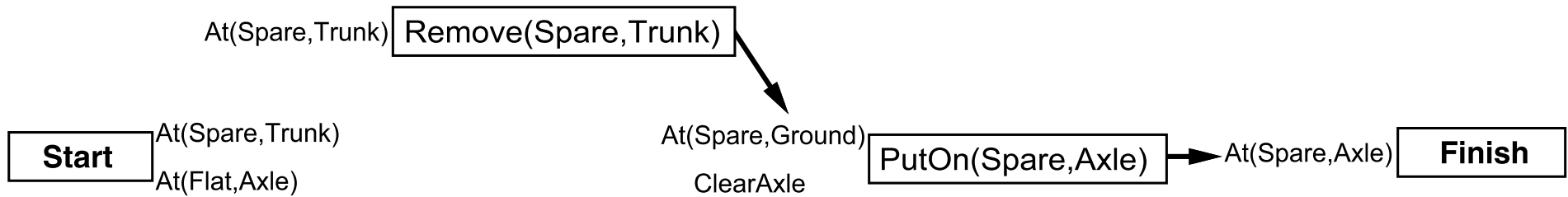
Add constraint $PutOn(Spare, Axle) < Finish$

– $PutOn(Spare, Axle)$ is on the left

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (۵ از ۸)

SPARE TIER PROBLEM



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

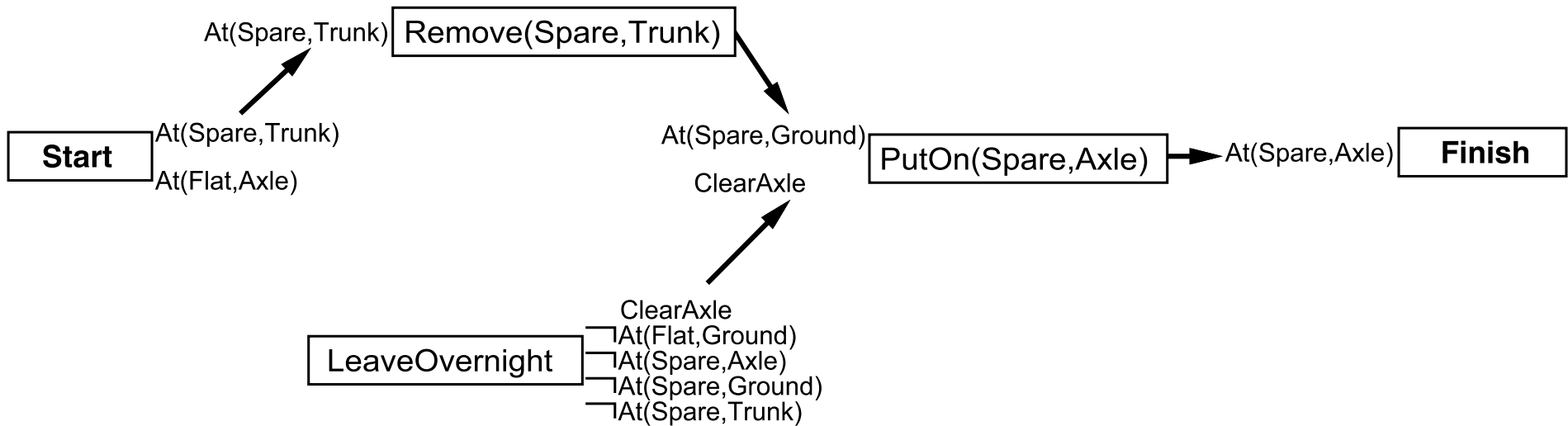
Only $Remove(Spare, Trunk)$ achieves that

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

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (۶ از ۸)

SPARE TIER PROBLEM



Pick an open precondition *ClearAxle*

E.g. *LeaveOvernight* can achieve that

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

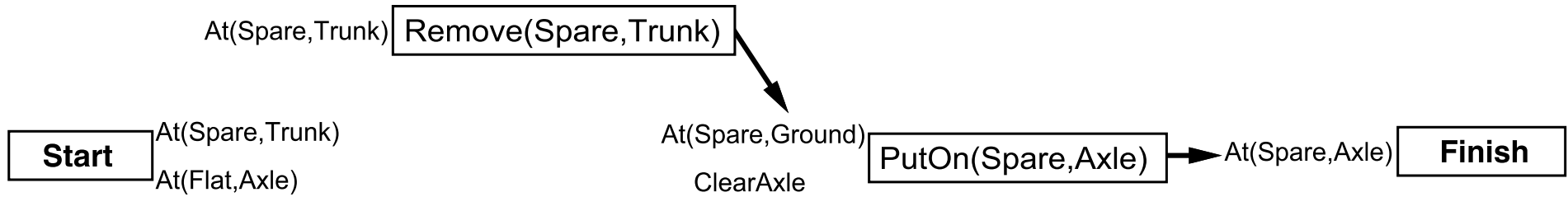
Add constraint $LeaveOvernight < Remove(Spare, Trunk)$

shift *LeaveOvernight* further left

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (۷ از ۸)

SPARE TIER PROBLEM

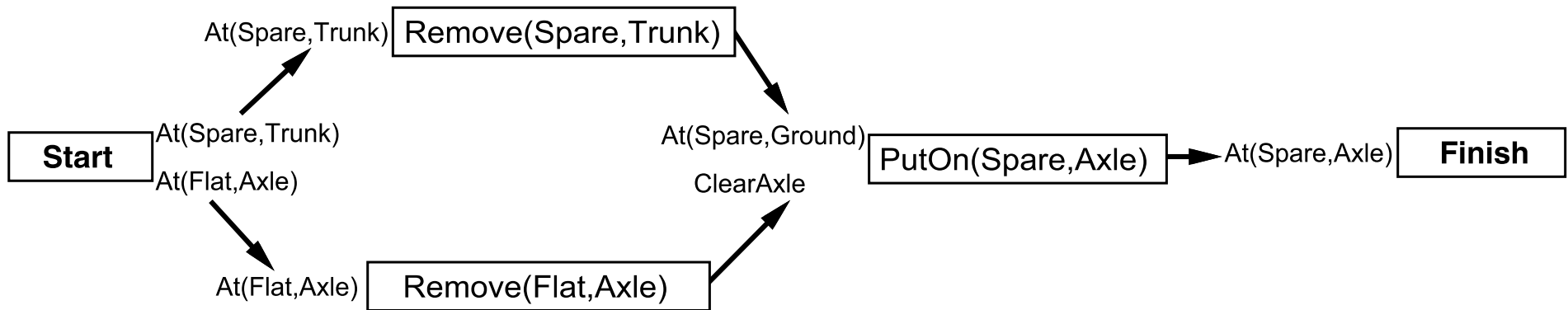


Undo ...

طرح ریزی ترتیب جزئی

مثال: مسئله‌ی تایریدکی (۸ از ۸)

SPARE TIER PROBLEM



Pick an open precondition *ClearAxle*

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

Finish the search

طرح ریزی ترتیب جزئی

شبه کد (۱ از ۲)

PARTIAL-ORDER PLANNING

function POP(*initial, goal, operators*) **returns** *plan*

plan $\leftarrow$ MAKE-MINIMAL-PLAN(*initial, goal*)

loop do

if SOLUTION?(*plan*) **then return** *plan*

$S_{need}, c \leftarrow$ 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

طرح ریزی 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)



---

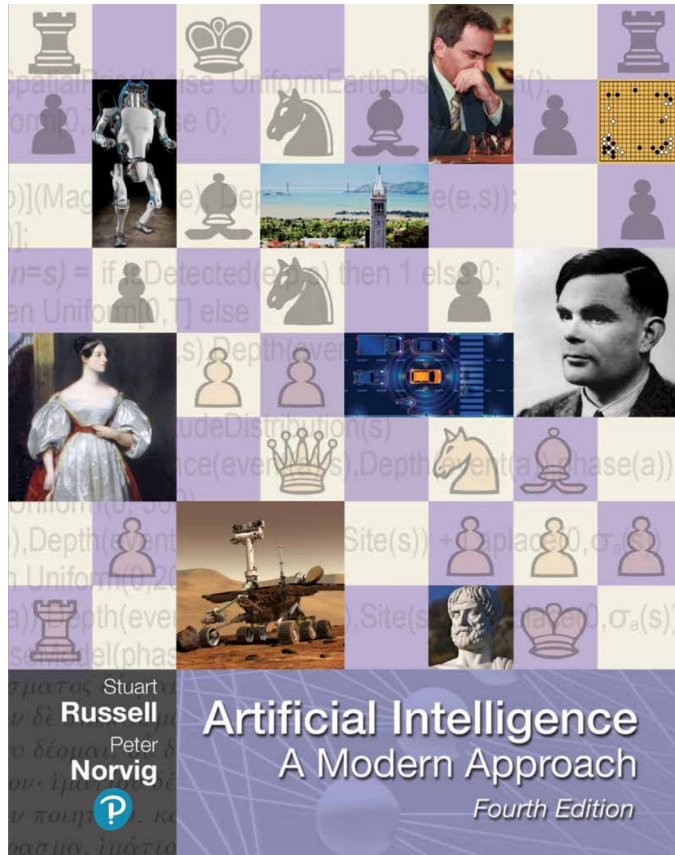

procedure RESOLVE-THREATS(plan)

  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
  
```

۴

منابع

منبع اصلی



Stuart Russell and Peter Norvig,
Artificial Intelligence: A Modern Approach,
 4th Edition, Prentice Hall, 2020.

Chapter 11

CHAPTER 11

AUTOMATED PLANNING

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

Planning a course of action is a key requirement for an intelligent agent. The right representation for actions and states and the right algorithms can make this easier. In Section 11.1 we introduce a general **factored** representation language for planning problems that can naturally and succinctly represent a wide variety of domains, can efficiently scale up to large problems, and does not require ad hoc heuristics for a new domain. Section 11.4 extends the representation language to allow for hierarchical actions, allowing us to tackle more complex problems. We cover efficient algorithms for planning in Section 11.2, and heuristics for them in Section 11.3. In Section 11.5 we account for partially observable and nondeterministic domains, and in Section 11.6 we extend the language once again to cover scheduling problems with resource constraints. This gets us closer to planners that are used in the real world for planning and scheduling the operations of spacecraft, factories, and military campaigns. Section 11.7 analyzes the effectiveness of these techniques.

11.1 Definition of Classical Planning

Classical planning

Classical planning is defined as the task of finding a sequence of actions to accomplish a goal in a discrete, deterministic, static, fully observable environment. We have seen two approaches to this task: the problem-solving agent of Chapter 3 and the hybrid propositional logical agent of Chapter 7. Both share two limitations. First, they both require ad hoc heuristics for each new domain: a heuristic evaluation function for search, and hand-written code for the hybrid wumpus agent. Second, they both need to explicitly represent an exponentially large state space. For example, in the propositional logic model of the wumpus world, the axiom for moving a step forward had to be repeated for all four agent orientations, T time steps, and n^2 current locations.

PDDL

In response to these limitations, planning researchers have invested in a **factored representation** using a family of languages called **PDDL**, the Planning Domain Definition Language (Ghallab *et al.*, 1998), which allows us to express all $4Tn^2$ actions with a single action schema, and does not need domain-specific knowledge. Basic PDDL can handle classical planning domains, and extensions can handle non-classical domains that are continuous, partially observable, concurrent, and multi-agent. The syntax of PDDL is based on Lisp, but we will translate it into a form that matches the notation used in this book.

State

In PDDL, a **state** is represented as a conjunction of ground atomic fluents. Recall that “ground” means no variables, “fluent” means an aspect of the world that changes over time,