

# بسمه تعالی

هوش مصنوعی

## جستجو در محیطهای پیچیده - ۳

نیمسال اول ۱۴۰۴-۱۴۰۳

دکتر مازیار پالهنک

آزمایشگاه هوش مصنوعی

دانشکده مهندسی برق و کامپیوتر

دانشگاه صنعتی اصفهان

# یادآوری

- الگوریتمهای جستجوی محلی
- حالت فعلی را نگهدار - سعی کن آن را بهبود دهی
- جستجوی تپه نوردی
- تنوعها:
- تپه نوردی تصادفی، تپه نوردی اولین انتخاب، تپه نوردی با باز شروع تصادفی
- سرد شدن شبیه سازی شده
- جستجوی پرتو محلی، و تصادفی
- الگوریتم ژنتیک
- جستجوی محلی در فضای پیوسته
- گرادیان، نیوتن - رافسن
- جستجو با اعمال غیر قطعی

# جستجو با اعمال غیر قطعی

■ مثال ربات جاروی سرگردان:

■ محیط مشاهده پذیر-غیر قطعی

■ عمل مکش:

■ در خانه کثیف آن خانه را تمیز ولی گاهی خانه

همسایه را نیز تمیز می کند،

■ در خانه تمیز گاهی آشغال می ریزد.

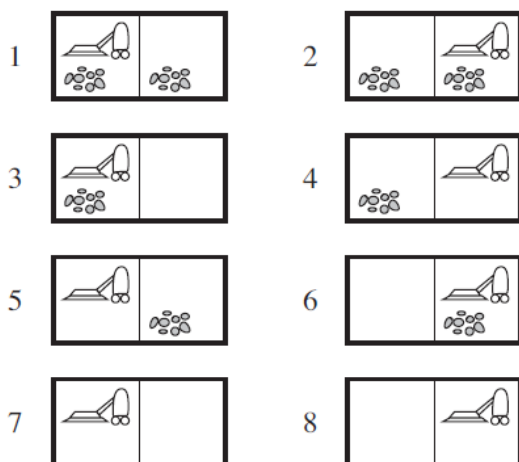
■ تعمیم مدل انتقال در فصل ۳

■ تابع `Result()` یک مجموعه از حالات قابل

دستیابی ممکن باز می گرداند.

■ بطور مثال: مکش در حالت ۱ به مجموعه

{۷و۵} می رود.



# جستجو با اعمال غیر قطعی

- اصلاح حل: بجای یک دنباله یک طرح شرطی (یا طرح اقتضائی یا راهبرد (استراتژی))
- بطور مثال با شروع از حالت ۱:  
 $[Suck, \text{if } State = 5 \text{ then } [Right, Suck] \text{ else } []]$  .
- حل مسائل همراه با عدم قطعیت شامل جملات اگر-آنگاه
- حل بصورت یک درخت بجای یک دنباله
- انتخاب عمل بر اساس اقتضاء بعد از عمل

# درخت جستجوی AND-OR

- جستجو مجدداً با ایجاد یک درخت جستجو
- در حالت قطعی شاخه ها با اعمالی که عامل می تواند انجام دهد ایجاد می شد.
- به چنین رئوس ایجاد شده، **رئوس-یا** (OR nodes) می گوئیم.
- در حالت غیرقطعی باید همه رئوسی که ممکن است نتایج انجام عمل باشند نیز ایجاد شود.
- این رئوس، **رئوس-و** (AND nodes) گفته می شوند.



# جستجو با اعمال غیر قطعی

- یک حل برای یک جستجو و-یا (AND-OR) یک زیر درخت از درخت جستجوی کامل است بطوری که:
- در هر برگ یک رأس هدف وجود دارد،
- در هر یک از رئوس یا در آن فقط یک عمل مشخص شده، و
- در هر یک از رئوس و در آن تمامی پیشامدها در نظر گرفته شده اند.

---

Figure 4.11

---

**function** AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*  
**return** OR-SEARCH(*problem*, *problem*.INITIAL, [])



---

Figure 4.11

---

**function** AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*  
    **return** OR-SEARCH(*problem*, *problem*.INITIAL, [])

**function** OR-SEARCH(*problem*, *state*, *path*) **returns** a conditional plan, or *failure*  
    **if** *problem*.IS-GOAL(*state*) **then return** the empty plan  
    **if** IS-CYCLE(*path*) **then return** *failure*  
    **for each** *action* **in** *problem*.ACTIONS(*state*) **do**  
        *plan* ← AND-SEARCH(*problem*, RESULTS(*state*, *action*), [*state*] + *path*)  
        **if** *plan* ≠ *failure* **then return** [*action*] + *plan*  
    **return** *failure*

چون عمق نخست است هر شاخه یا به جواب می رسد یا شکست می خورد

---

Figure 4.11

---

**function** AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*  
    **return** OR-SEARCH(*problem*, *problem*.INITIAL, [])

**function** OR-SEARCH(*problem*, *state*, *path*) **returns** a conditional plan, or *failure*  
    **if** *problem*.IS-GOAL(*state*) **then return** the empty plan  
    **if** IS-CYCLE(*path*) **then return** *failure*  
    **for each** *action* **in** *problem*.ACTIONS(*state*) **do**  
        *plan*  $\leftarrow$  AND-SEARCH(*problem*, RESULTS(*state*, *action*), [*state*] + *path*)  
        **if** *plan*  $\neq$  *failure* **then return** [*action*] + *plan*  
    **return** *failure*

**function** AND-SEARCH(*problem*, *states*, *path*) **returns** a conditional plan, or *failure*  
    **for each** *s<sub>i</sub>* **in** *states* **do**  
        *plan<sub>i</sub>*  $\leftarrow$  OR-SEARCH(*problem*, *s<sub>i</sub>*, *path*)  
        **if** *plan<sub>i</sub>* = *failure* **then return** *failure*  
    **return** [if *s<sub>1</sub>* **then** *plan<sub>1</sub>* **else if** *s<sub>2</sub>* **then** *plan<sub>2</sub>* **else ... if** *s<sub>n-1</sub>* **then** *plan<sub>n-1</sub>* **else** *plan<sub>n</sub>*]

An algorithm for searching AND-OR graphs generated by nondeterministic environments. A solution is a conditional plan that considers every nondeterministic outcome and makes a plan for each one.

---

# تلاش و تلاش

■ دنیای جاروی لغزنده را در نظر بگیرید،

همانند دنیای جاروی معمولی ولی

■ عمل راست یا چپ گاهی انجام می شود

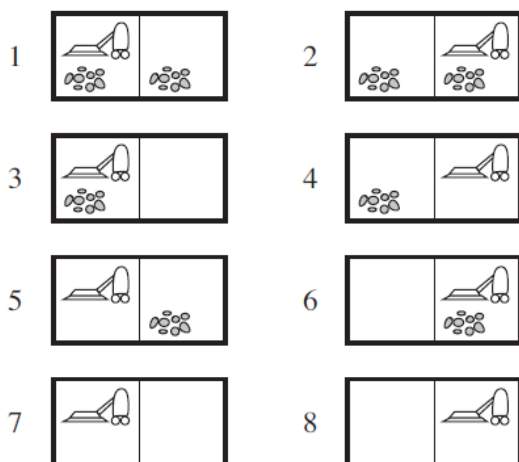
و گاهی در همان خانه ای که بودیم می مانیم.

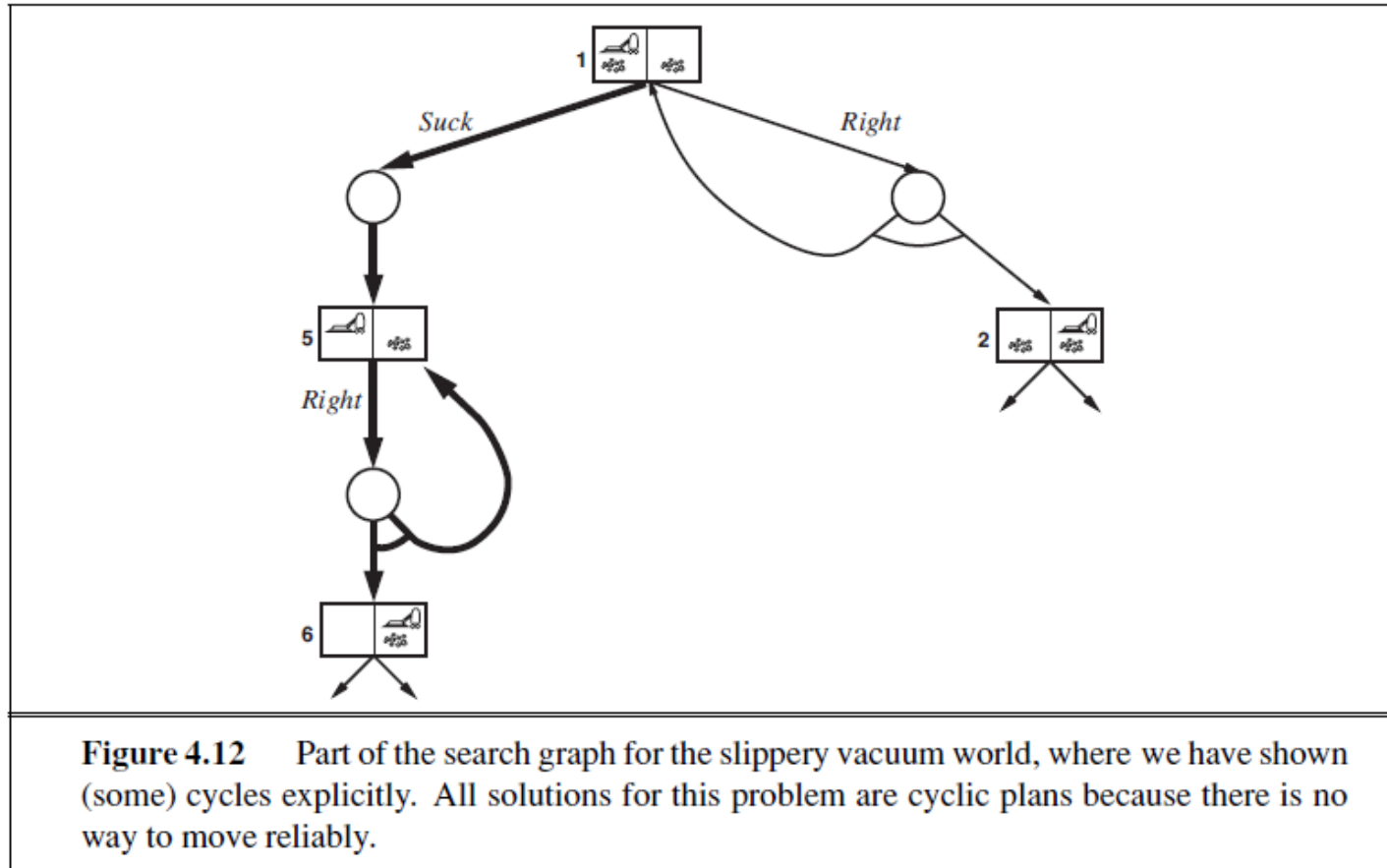
■ مثلاً اگر در حالت ۱، به راست برویم به

حالت ۱ یا ۲ ممکن است برویم.

■ در این وضعیت، حل بدون حالت

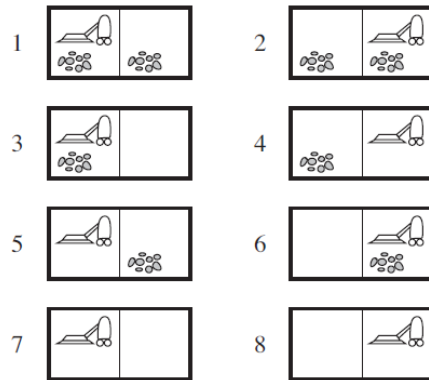
تکراری نخواهیم داشت.





- حل چرخشی وجود دارد.
- حداقل هر برگ در طرح یک حالت هدف بوده و برگ قابل رسیدن باشد.
- نمایش با اضافه کردن برچسب:

$[Suck, L_1 : Right, \text{if } State = 5 \text{ then } L_1 \text{ else } Suck]$



■ یا نمایش با حلقه while:

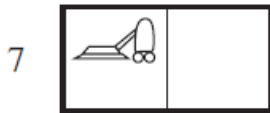
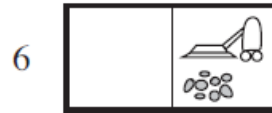
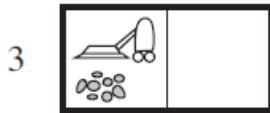
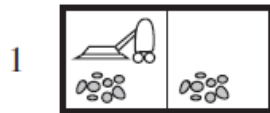
$[Suck, \text{while } State = 5 \text{ do } Right, Suck]$

# جستجو در محیط نیمه مشاهده پذیر

- عامل دقیق نمی داند در چه حالتی است.
- حتی اگر اعمال قطعی باشند، انجام عمل منجر به یک مجموعه حالات خواهد شد.
- در این حالت، بهتر است مسئله را در فضای باور بجای فضای فیزیکی حل کنیم.
- حالت باور شامل مجموعه ای از حالات فیزیکی است که عامل تصور می کند در آن قرار دارد.

# جستجو بدون مشاهده

■ بدترین حالت، عامل حسگر نداشته باشد.



■ عامل دانش محیطی که در آن هست را در اختیار دارد ولی اینکه در چه مکانی قرار دارد یا توزیع آشغال را نمی داند.  
■ اعمال قطعی

■ حالت اولیه {۱و۲و۳و۴و۵و۶و۷و۸}

■ [راست] به {۲و۴و۶و۸}

■ [راست، مکش] به {۸و۴}

■ نهایتاً [راست، مکش، چپ، مکش]

- جستجو در فضای باور
- حل در صورت وجود یک دنباله است.
- چون حسگری هم وجود ندارد مسئله اقتضائی نیست و می دانیم پس از حرکت در کی هم نخواهیم داشت.
- حتی اگر محیط غیرقطعی باشد.
- در این حالت در فضای باور محیط به نوعی مشاهده پذیر است چون عامل می داند به چه باوری می رود.
- می توان اجزائی که برای تعریف یک مسئله وجود دارد را برای فضای باور تعریف نمود.



- چون ۸ حالت داریم.
- تعداد حالات باور  $2^8=256$  است.
- برابر با مجموعه قوه یک مجموعه ۸ عضوی
- همه آنها در این مثال دسترسی نمی شوند.
- می توان از همان جستجوهای کلاسیک استفاده کرد،
- با تعریف اجزاء مسئله برای فضای باور

■ اجزاء مسئلہ اولیہ P را با اندیس P نمایش می دهیم، مثل  $\text{Result}_P, \text{Actions}_P$

■ **حالات:** اگر P شامل N حالت باشد، مسئلہ فضای باور دارای  $2^N$  باور است.

■ **حالت اولیہ:** یکی از باورها، معمولاً آنکه شامل همه حالات P است.

■ **اعمال:** ممکن است:  $b = \{s_1, s_2\} \quad \text{ACTIONS}_P(s_1) \neq \text{ACTIONS}_P(s_2)$

■ اگر انجام عملی مجاز روی حالتی دیگر مضر نباشد:

$$\text{ACTIONS}(b) = \bigcup_{s \in b} \text{ACTIONS}_P(s) .$$

■ اگر مضر باشد شاید بهتر باشد اشتراک اعمال مجاز روی هر حالت را در نظر بگیریم.

■ مدل انتقال:

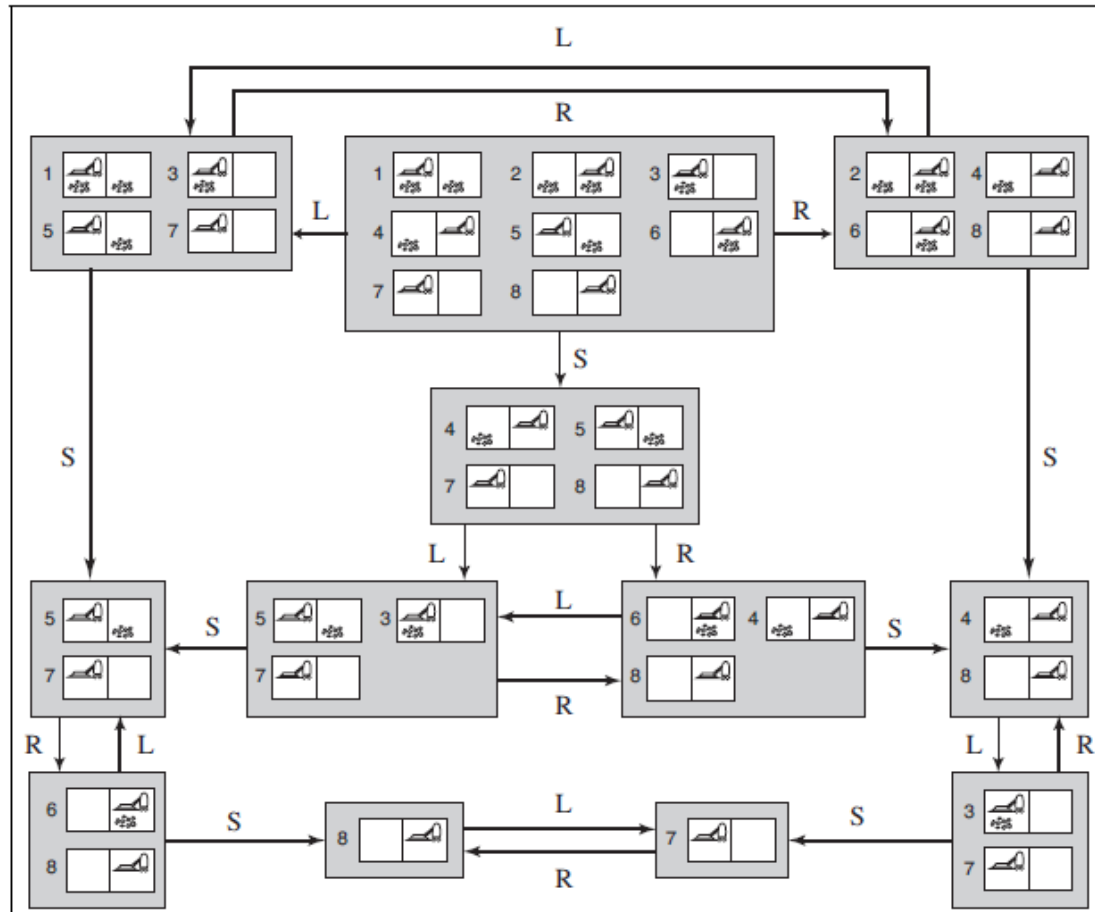
■ اعمال قطعی:  $b' = \text{RESULT}(b, a) = \{s' : s' = \text{RESULT}_P(s, a) \text{ and } s \in b\}$ .

■ اعمال غیرقطعی:  $b' = \text{RESULT}(b, a) = \{s' : s' \in \text{RESULTS}_P(s, a) \text{ and } s \in b\}$   
 $= \bigcup_{s \in b} \text{RESULTS}_P(s, a),$

■ تست هدف: هر حالت در باور یک حالت هدف است.

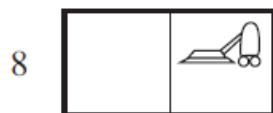
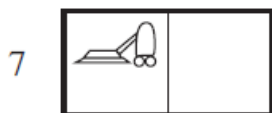
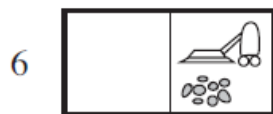
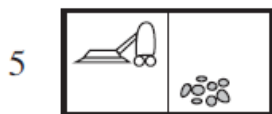
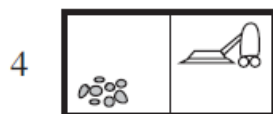
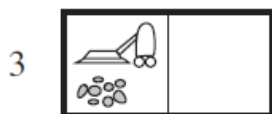
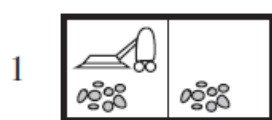
■ هزینه عمل:

- فرض می کنیم هزینه اعمال یک عمل روی حالات مختلف در یک باور یکسان باشد، در این صورت همان منظور خواهد شد.
- حالتی که تفاوت هزینه وجود دارد فعلاً مورد بحث ما نیست.



**Figure 4.14** The reachable portion of the belief-state space for the deterministic, sensor-less vacuum world. Each shaded box corresponds to a single belief state. At any given point, the agent is in a particular belief state but does not know which physical state it is in. The initial belief state (complete ignorance) is the top center box. Actions are represented by labeled links. Self-loops are omitted for clarity.

# جستجو با مقداری درک



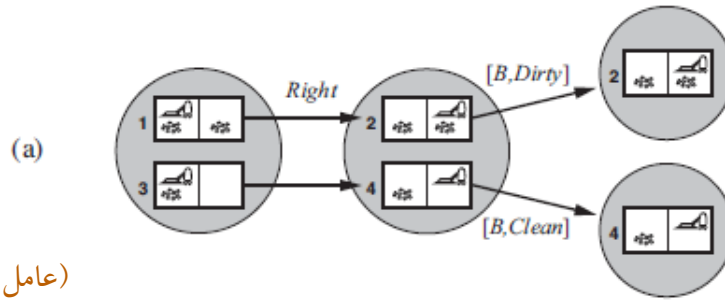
■ فرض درک فقط شامل مکان و وجود آشغال در خانه فعلی

■ درک [چپ، تمیز] در حالت {۷و۵}

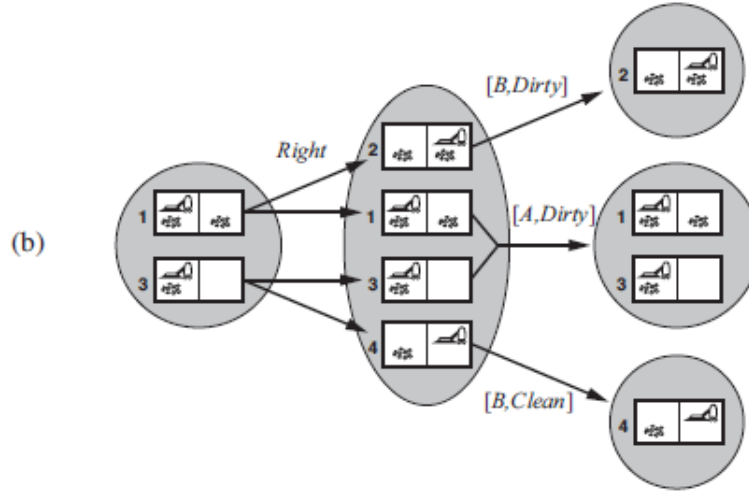
■ حل [راست اگر آشغال مکش]

■ مسئله شرطی (اقتضائی)

قطعی  
(عامل با درک محلی)



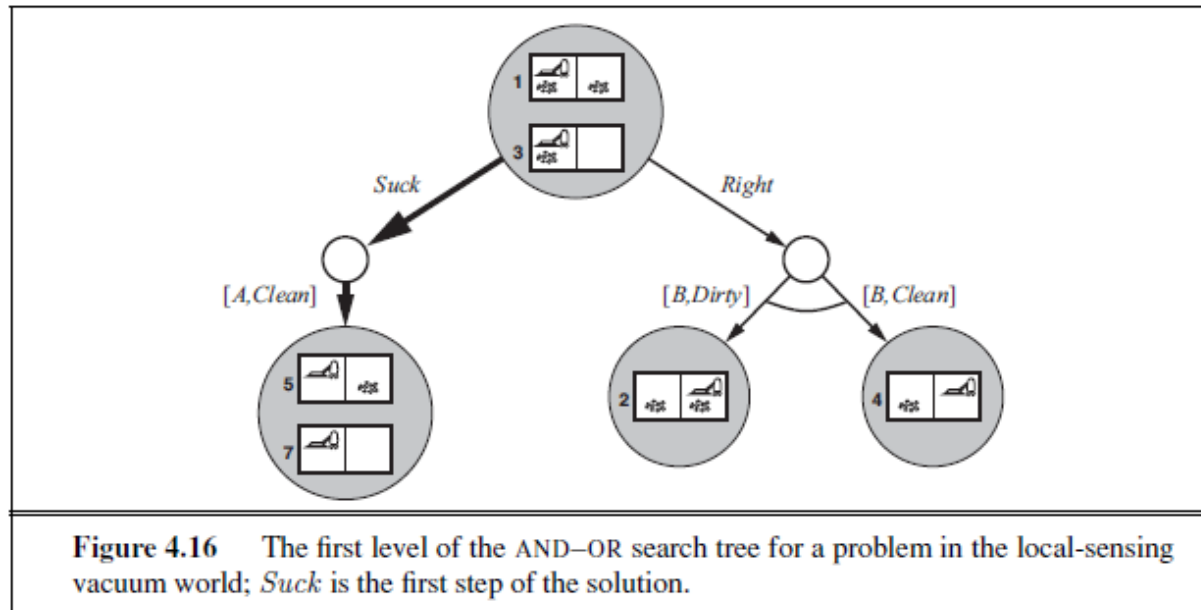
غير قطعی  
(دنیای لغزنده)



**Figure 4.15** Two example of transitions in local-sensing vacuum worlds. (a) In the deterministic world, *Right* is applied in the initial belief state, resulting in a new belief state with two possible physical states; for those states, the possible percepts are  $[B, Dirty]$  and  $[B, Clean]$ , leading to two belief states, each of which is a singleton. (b) In the slippery world, *Right* is applied in the initial belief state, giving a new belief state with four physical states; for those states, the possible percepts are  $[A, Dirty]$ ,  $[B, Dirty]$ , and  $[B, Clean]$ , leading to three belief states as shown.

# حل مسائل نیمه مشاهده پذیر

■ استفاده از الگوریتم جستجوی AND-OR



■ حل یک طرح شرطی .  $[Suck, Right, \text{if } Bstate = \{6\} \text{ then } Suck \text{ else } []]$



# عاملی برای محیط نیمه مشاهده پذیر

- دو تفاوت با عاملی که در محیط مشاهده پذیر عمل می کند.
- اول: حل یک طرح شرطی است بجای دنباله
- دوم: عامل باید یک باور در خود ذخیره نماید.
- فرآیند پیش بینی، مشاهده، بروز رسانی

$$b' = \text{UPDATE}(\text{PREDICT}(b, a), o)$$

- شناخته شده تحت نام نظارت کردن، صافی کردن (filtering)،  
یا تخمین حالت (state estimation)

# خلاصه

- جستجو با اعمال غیرقطعی
  - حل یک طرح شرطی، استفاده از درخت AND-OR با استفاده از فضای حالت
- جستجو برای عامل بدون حسگر
  - جستجو در فضای باور همانند حالت مشاهده پذیر با تعمیم تعریف اجزاء مسئله
- جستجو در محیط نیمه مشاهده پذیر
  - حل یک طرح شرطی، استفاده از درخت AND-OR با استفاده از فضای باور



- دقت نمائید که پاورپوینت ابزاری جهت کمک به یک ارائه شفاهی می باشد و به هیچ وجه یک جزوه درسی نیست و شما را از خواندن مراجع درس بی نیاز نمی کند.
- لذا حتماً مراجع اصلی درس را مطالعه نمائید.
- حضور فعال در کلاس دارای امتیاز است.