

## مراجع و منابع

- هوش مصنوعی: رهیافت نوین (S. Russell & P. Norvig)
- برنامه نویسی پرولوگ برای هوش مصنوعی (I. Bratko)
- هوش مصنوعی (E. Rich)
- هوش مصنوعی (Luger)

# هوش مصنوعی معرفی

فصل اول

سید ناصر رضوی

Email: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۶

## وظایف و مسولیت ها

- **مربی**
  - رفتار محترمانه با دانشجویان
  - رفتار منصفانه
  - در دسترس بودن برای پاسخگویی
  - آموزش آنچه که دانشجو باید بداند (رعایت سرفصل)
  - بیان نحوه ارزیابی
- **دانشجو**
  - رفتار محترمانه
  - اجتناب از کلک زدن، دروغ گویی و سرقت آثار دیگران و یا کمک به افراد دیگر در انجام چنین کارهایی
  - عدم ایجاد مزاحمت و مداخله در فعالیت‌های دانشگاهی دیگران
- **عواقب: کاهش نمره و یا مردود شدن در درس**



N. Razavi - AI course - 2005

2

## نحوه ارزیابی

- سه الی چهار تمرین و کوئیز (10%)
- دو پروژه (5%)
- پرولوگ (15%)
- جستجو (15%)
- دو امتحان (30%)
- میان ترم (40%)
- پایان ترم (40%)
- نحوه ارزیابی تمرین‌ها و پروژه‌های معوقه:
  - تأخیر حداقل یک هفته: ۲۰ درصد جریمه
  - تأخیر بیش از یک هفته: صفر

## سرفصل

- مرور درس
- هوش مصنوعی چست؟
- یک تاریخچه مختصر
- وضعیت فعلی هوش مصنوعی

N. Razavi - AI course - 2005

5

## مژو) دلس

- فصل اول: معرفی
- فصل دوم: عامل ها (agents)
- فصل سوم و چهارم و پنجم و ششم: جستجو
- فصل هفتم و هشتم و نهم: منطق

N. Razavi - AI course - 2005

6

## هوش مصنوعی چیست؟

- تعاریف هوش مصنوعی در چهار دسته قرار می گیرند

منطقی فکر کردن	انسان گونه فکر کردن
منطقی عمل کردن	انسان گونه عمل کردن

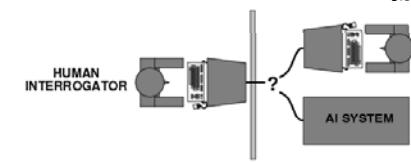
- کتاب درسی طرفدار «منطقی عمل کردن»

N. Razavi - AI course - 2005

7

## انسان گونه (فتار کردن: تست توپینگ

- تورینگ (۱۹۵۰) «ماشین های محاسباتی و هوشمندی»
- «آیا ماشین های توانند فکر کنند؟» ← «آیا ماشین های توانند هوشمندانه عمل کنند؟»
- یک آزمون عملی برای رفتار هوشمندانه



- وی پیش بینی نمود که تا سال ۲۰۰۰ ماشین های تا ۳۰ درصد شانس فریب دادن یک انسان عامی را خواهند داشت.
- قابلیت های مورد نیاز کامپیوتر

- پردازش زبان طبیعی
- بازنمایی داشش
- استدلال خودکار
- یادگیری

N. Razavi - AI course - 2005

8

## انسان گونه فکر کردن: مدلسازی شناختی

- در ک چگونگی تفکر انسانی و عملکرد مغز
  - درون گرایی
  - تجارت روانشناسی
- به دنبال ایجاد تئوری دقیقی درباره عملکرد ذهن انسان و تبدیل آن به برنامه کامپیووتری
- GPS: سیمون و نیوول، ۱۹۶۳
- در پی تعقیب مراحل استدلال برنامه و مقایسه آن با مراحل حل مسائل توسط انسان
- آنالیز هدف-وسیله

N. Razavi - AI course - 2005

9

## منطقی فکر کردن: < قوانین تفکر >

- ارسسطو: «فرآیند استدلال/تفکر درست چیست؟»
- مثال: "سقراط انسان است، تمام انسانها فانی هستند، پس سقراط فانی است.".
- پایه ریزی منطق (Logic)
- برنامه هایی براساس قوانین تفکر برای ایجاد سیستمهای هوشمند
- موانع اصلی
- دریافت دانش غیررسمی و تبدیل آن به دانش رسمی: «اکثر انسانها پرتلاش هستند.»
- تفاوت میان قادر به حل مسئله بودن در تئوری و در عمل (بن بست محاسباتی)

N. Razavi - AI course - 2005

10

## منطقی عمل کردن: عامل منطقی

- رفتار منطقی: انجام عمل درست
- عمل درست: عملی که با توجه به اطلاعات موجود، انتظار می رود شانس رسیدن به هدف را به حد اکثر برساند.
- لزوماً شامل تفکر نمی باشد - مانند پلک زدن - اما تفکر باید در خدمت عمل منطقی باشد.

N. Razavi - AI course - 2005

11

## عامل منطقی

- عامل: هر چیزی که قادر به در ک نمودن و عمل کردن باشد.
- این درس در مورد طراحی عامل های منطقی می باشد.
- به طور انتزاعی، عامل یک تابع از تاریخچه ادراکی بر روی اعمال می باشد:  
$$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$$
- ما برای هر دسته از محیط ها و وظایف مختلف، به دنبال عاملی (یا دسته ای از عامل ها) با بهترین کارایی می باشیم.
- هشدار: محدودیت های محاسباتی باعث شده اند که منطقی بودن به طور کامل، غیر قابل دسترس باشد.
- طراحی بهترین برنامه برای منابع ماشینی داده شده

N. Razavi - AI course - 2005

12

# علوم زیربنایی هوش مصنوعی

منطق، روش های استدلال، ذهن به عنوان سیستم فیزیکی زیربنایی در یادگیری، زبان، منطقی بودن	فلسفه
روش های بازنمایی رسمی و الگوریتم های اثبات، محاسبات، تصمیم(نا) پذیری، احتمالات	ریاضیات
بهره وری، نظریه تصمیم	اقتصاد
مواد فیزیکی برای فعالیت های ذهنی	عصب شناسی
پدیده ادراک و کنترل، تکنیک های آزمایشگاهی	روان شناسی
مهندسی کامپیوتر ایجاد کامپیوترهای سریع طراحی سیستم هایی به منظور بیشینه سازی یک تابع هدف در طول زمان	نظریه کنترل
بازنمایی دانش، گرامر	زبان شناسی

N. Razavi - AI course - 2005

13

# تاریخچه مختصر هوش مصنوعی (ادامه)

ورود هوش مصنوعی در عرصه های صنعتی	۱۹۸۰
محبوبیت مجدد شبکه های عصبی	۱۹۸۶
تبديل شدن هوش مصنوعی به یک علم	۱۹۸۷
ظهور عامل های هوشمند	۱۹۹۵

N. Razavi - AI course - 2005

15

# تاریخچه مختصر هوش مصنوعی

مک کالچ و پیتر: مدل مداری بولی از ذهن	۱۹۴۳
تورینگ: «ماشین های محاسباتی و هوشمندی»	۱۹۵۰
نشت در دارتموث: پیدایش «هوش مصنوعی»	۱۹۵۶
اولین برنامه های هوش مصنوعی شامل: برنامه ساموئل برای انجام بازی چکر، برنامه Logic Theorist سیمون و نیوول و ...	۵۰
الگوریتم کامل رابینسون برای استدلال منطقی	۱۹۶۵
پیدایش نظریه پیچیدگی محاسباتی، توقف تحقیقات بر روی شبکه های عصبی	۷۳-۱۹۶۶
پیدایش سیستم های اولیه مبتنی بر دانش	۷۹-۱۹۶۹

N. Razavi - AI course - 2005

14

# وضعيت فعلی هوش مصنوعی

- شکست قهرمان شطرنج (گری کاسپاروف) توسط Deep Blue در سال ۱۹۹۷
- اثبات یک حدس ریاضیاتی (حدس رابینز) که برای چندین دهه به صورت حل نشده باقی مانده بود.
- کنترل خودکار (هدایت اتوماتیک یک اتومبیل از پیتربرگ تا سن دیه گو در ۰.۹۸٪ از مسیر - سیستم بینایی ALVINN برنامه های زمان بندی خودکار ناسا
- حل بهتر جداول کلمات متقطع توسط PROVERB نسبت به انسان
- سیستم های خبره پزشکی
- ...

N. Razavi - AI course - 2005

16

## آلوس مطالب

- عامل ها و محیط ها
- منطقی بودن
- PEAS (معیار کارآیی، محیط، اثر کننده ها، حسگرها)
- انواع محیط ها
- انواع عامل ها

## عامل های هوشمند

فصل دوم  
سید ناصر رضوی

Email: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

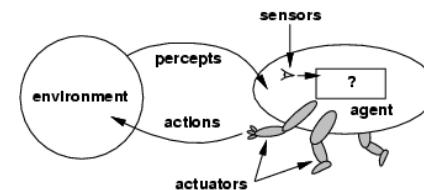
۱۳۸۴

## عامل ها

- **عامل:** هر چیزی که بتواند **محیط** پیرامونش را از طریق **حسگرها** درک کند و در آن محیط از طریق **اثر کننده ها** عمل کند.
- عامل انسانی:
  - چشم ها و گوش ها و سایر اندام های حسی به عنوان حسگرها
  - دست ها، پاهای، دهان و سایر اعضای بدن به عنوان اثر کننده ها
- عامل روبات:
  - دوربین ها و فاصله یاب مادون قرمز به عنوان حسگرها
  - انواع موتورها به عنوان اثر کننده ها

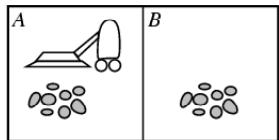
پرماش پرتاب آموزشی  
<http://Parmasoft.ir>  
<http://Facebook.com/Parmasoft>

## عامل ها و محیط ها



- **تابع عامل** تاریخچه ادراکی را به اعمال نگاشت می کند:  
 $[f: \mathcal{P}^* \rightarrow \mathcal{A}]$
- برنامه **عامل** برای ایجاد  $f$  بر روی **معماری** فیزیکی اجرا می شود.
- **عامل** = **معماری** + برنامه

## دنیای جارو برقی



- ادراک ها: مکان ها و محتويات آنها، مانند  $[A, Dirty]$
- اعمال: حرکت به چپ و راست، مکش و  $NoOp$

N. Razavi- AI course- 2005

5

## عامل های منطقی

- یک عامل باید بر اساس آنچه که می تواند در ک کند و اعمالی که می تواند انجام دهد، «کار درست را انجام دهد». عمل درست آن است که باعث شود عامل بیشترین موفقیت را بدست آورد.
- معیار کارآیی:** یک معیار هدف برای سنجش میزان موفقیت رفتار یک عامل
- مثال: معیار موفقیت عامل دنیای جارو برقی:
  - مقدار گرد و خاک تمیز شده
  - میزان زمان مصرف شده
  - مقدار برق مصرف شده
  - میزان سرو صدای تولید شده و ...

N. Razavi- AI course- 2005

6

## عامل های منطقی

- عامل منطقی:** برای هر دنباله ادراکی ممکن، یک عامل منطقی باید بر اساس شواهد دریافتی از **دنباله ادراکی** و **دانش درونی عملی** را انتخاب کند که انتظار می رود معیار **کارآیی** اش را به حداکثر برساند.

N. Razavi- AI course- 2005

7

## عامل های منطقی

- منطقی بودن با دانش کل بودن (دانستن همه چیز توسط دانش نامحدود) تفاوت دارد.
- عامل می تواند اعمالی را انجام دهد که از طریق تغییر در ادراک های آتی اطلاعات مفید بدست آورد (جمع آوری دانش، اکتشاف)
- یک عامل **خودمختار** است اگر رفتارش بر اساس تجربه اش تعیین شود (به همراه قابلیت یادگیری و تطبیق پذیری)

N. Razavi- AI course- 2005

8

## PEAS

- PEAS: Performance measure, Environment, Actuators, Sensors
  - در طراحی یک عامل ابتدا باید موارد بالا تعیین گردد.
  - مثال: طراحی یک راننده تاکسی اتوماتیک
    - معیار کارآیی: امنیت، سرعت، راحتی، سود و ...
    - محیط: خیابان ها، افراد پیاده، مشتری ها و ...
    - اثر کننده ها: فرمان، شتاب دهنده، ترمزها، بوق، چراغ ها و ...
  - حسگرهای صوتی (Sonar)، سرعت سنج، دوربین ها، GPS، کیلومتر شمار، حسگرهای موتور، صفحه کلید، میکروفون و ...

N. Razavi- AI course- 2005

9

## PEAS

- عامل: سیستم تشخیص پزشکی
  - معیار کارآیی: سلامتی بیمار، به حداقل رساندن هزینه و ...
  - محیط: بیمار، بیمارستان، کارمندان و ...
  - اثر کننده ها: صفحه نمایش (پرسش ها، آزمایش ها، تشخیص ها، مداوا)
  - حسگرهای کلید (دریافت علایم، یافته ها و پاسخ های بیمار)

N. Razavi- AI course- 2005

10

## PEAS

- عامل: روبات جابه جا کننده اشیاء
  - معیار کارآیی: درصد قطعاتی که در صندوق درست قرار می گیرند
  - محیط: نوار نقاله و اشیاء روی آن، صندوق ها
  - اثر کننده ها: بازوها و دست
  - حسگرهای دوربین، حسگر زاویه مفاصل

N. Razavi- AI course- 2005

11

## PEAS

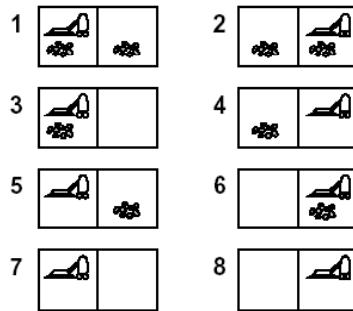
- آموزش دهنده زبان به صورت محاوره ای
  - معیار کارآیی: به حداقل رساندن نمره دانش آموز در امتحان
  - محیط: مجموعه دانش آموزان
  - اثر کننده ها: صفحه نمایش (تمرین ها، پیشنهادات و اصلاحات)
  - حسگرهای کلید

N. Razavi- AI course- 2005

12

## محیط

- هر محیط دارای مجموعه ای از حالت ها می باشد:
  - محیط در هر لحظه  **فقط** در یکی از این حالت ها می باشد.



- مثال: دنیای مکش

$$S = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

N. Razavi- AI course- 2005

13

## عامل و محیط

- در لحظه شروع، محیط در یکی از حالت های ممکن می باشد
  - عمل عامل در محیط، باعث **تغییر حالت** محیط می شود



- حالت فعلی:  $S_i$
- عمل عامل:  $\text{Action}$
- حالت بعدی:  $S_j$

- مثال: دنیای مکش



N. Razavi- AI course- 2005

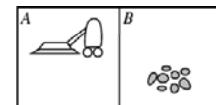
14

## انواع محیط

- **کاملا قابل مشاهده** (در مقابل مشاهده پذیر جزئی): محیطی که در آن در هر لحظه از زمان حسگرهای عامل به آن امکان دستیابی به حالت کامل محیط را می دهند.

- مثال: دنیای مکش - حسگرهای [location, status]:
  - تشخیص مکان: چپ یا راست
  - تشخیص وضعیت: تمیز یا کثیف

[ LEFT, [CLEAN, DIRTY] ] 



N. Razavi- AI course- 2005

15

## انواع محیط

- **قطعی:** (در مقابل اتفاقی): حالت بعدی محیط کاملا بوسیله حالت فعلی و عمل انجام شده توسط عامل قابل تعیین می باشد.
  - اگر محیط به جز در مورد عمل عامل های دیگر قطعی باشد، آنگاه محیط استراتژیک می باشد.

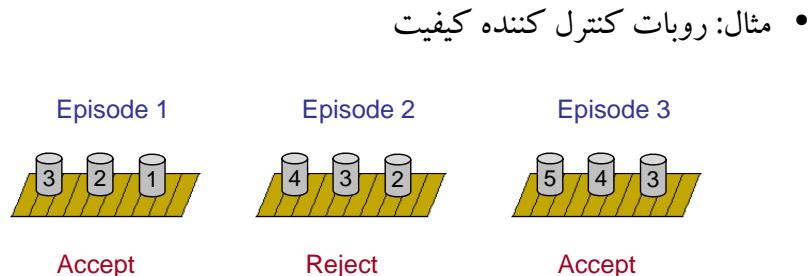


N. Razavi- AI course- 2005

16

## انواع محیط

- **پیزودیک** (در مقابل ترتیبی): تجربه عامل به «دوره های» غیرقابل تجزیه تقسیم می شود (هر دوره شامل ادراک عامل و سپس انجام یک عمل می باشد) و انتخاب عمل در هر دوره تنها به خود همان دوره بستگی دارد.

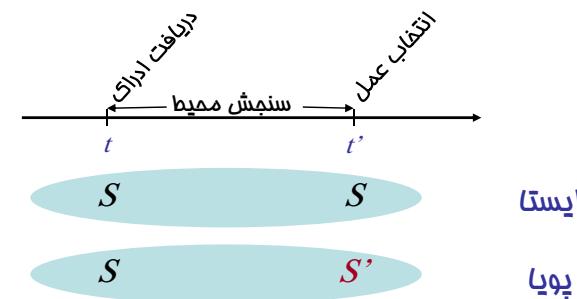


N. Razavi- AI course- 2005

17

## انواع محیط

- **ایستا** (در مقابل پویا): محیط در حین سنجش عامل (برای انتخاب عمل) تغییر نمی کند. اگر خود محیط با گذشت زمان تغییر نکند ولی معیار کارآیی عامل تغییر کند، آنگاه محیط **نیمه پویا** می باشد.



N. Razavi- AI course- 2005

18

## انواع محیط

- **گستته** (در مقابل پیوسته): محیطی که در آن تعداد محدود و متمایزی از درک ها و عمل های کاملاً واضح تعریف شده باشد.
- در محیط گستته، مجموعه حالات محیط یک مجموعه گستته می باشد و حالات بسادگی قابل تمایز می باشند.  
- مثال: محیط دنیای مکش
- $State = \{1, 2, \dots, 8\}$
- $Action = \{\text{Left}, \text{Right}, \text{Suck}, \text{NoOp}\}$
- $Percept = \{[\text{Left}, \text{Clean}], [\text{Left}, \text{Dirty}], [\text{Right}, \text{Clean}], \dots\}$

N. Razavi- AI course- 2005

19

## انواع محیط

- **تک عاملی** (در برابر چند عاملی): یک عامل خودش به تنها یی در محیط عمل می کند.
- مثال: محیط عامل حل کننده جدول کلمات متقطع و دنیای مکش
- **چند عاملی**: تعدادی عامل که با یکدیگر در تعامل می باشند.
- مثال: شطرنج (رقابتی)، روبوکاپ (بین اعضای یک تیم همیاری و بین اعضای دو تیم رقابتی)، محیط تاکسی خودکار (همیاری جزئی)

N. Razavi- AI course- 2005

20

## انواع محیط

رانندگی تاکسی	شرطنج بدون ساعت	شرطنج با ساعت	
خیر	بله	بله	کاملاً قابل مشاهده
خیر	استراتژیک	استراتژیک	قطعی
خیر	خیر	خیر	دورة ای
خیر	بله	نیمه پویا	ایستا
خیر	بله	بله	گستته
خیر	خیر	خیر	تک عاملی

- نوع محیط به میزان زیادی تعین کننده طراحی عامل می باشد.

- دنیای واقعی: مشاهده پذیر جزئی، اتفاقی، ترتیبی، پویا، پیوسته و چندعاملی

N. Razavi- AI course- 2005

21

## تابع و برنامه های عامل

- یک عامل کاملاً بوسیله تابع عامل مشخص می شود.

- یادآوری: تابع عامل دنباله ادراکی را به عمل نگاشت می کند.

- یک تابع عامل (یا یک کلاس هم ارزی کوچک) منطقی (rational) می باشد.

- هدف: یافتن روشی به منظور پیاده سازی تابع عامل منطقی به طور مختصر و مفید

N. Razavi- AI course- 2005

22

## عامل مبتنی بر جدول جستجو

- یک روش به منظور توصیف تابع عامل
- نشان دهنده فعالیت مناسب برای هر دنباله ادراکی ممکن
- مثال: جدول دنیای جاروبرقی

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
...	

N. Razavi- AI course- 2005

23

## برنامه عامل مبتنی بر جدول جستجو

```
function TABLE-DRIVEN-AGENT( percept) returns an action
```

```
static: percepts, a sequence, initially empty
```

```
table, a table of actions, indexed by percept sequence,  
initially fully specified
```

```
append percept to the end of percepts
```

```
action  $\leftarrow$  LOOKUP( percepts, table)
```

```
return action
```

N. Razavi- AI course- 2005

24

## عامل مبتنی بر جدول جستجو

- معایب:
  - جدول بسیار عظیم (مثلا در شطرنج ۱۰<sup>۱۵۰</sup> سطر)
  - زمان بسیار زیاد برای ایجاد جدول و احتمال بالای خطا
  - عدم خود مختاری
  - حتی با قابلیت یادگیری، نیاز به زمان بسیار زیادی برای یادگیری مداخل جدول دارد.

N. Razavi- AI course- 2005

25

## انواع عامل ها

- چهار نوع اصلی به ترتیب افزایش عمومیت (Generality)
  - (Simple reflex) - عامل های واکنشی ساده
  - (Model-based reflex) - عامل های واکنشی مبتنی بر مدل
  - (Goal-based) - عامل های مبتنی بر هدف
  - (Utility-based) - عامل های مبتنی بر سودمندی

N. Razavi- AI course- 2005

26

## عامل های واکنشی ساده

- ساده ترین نوع عامل
- در هر لحظه، عمل تنها بر اساس درک فعلی انتخاب می شود
- مثال:

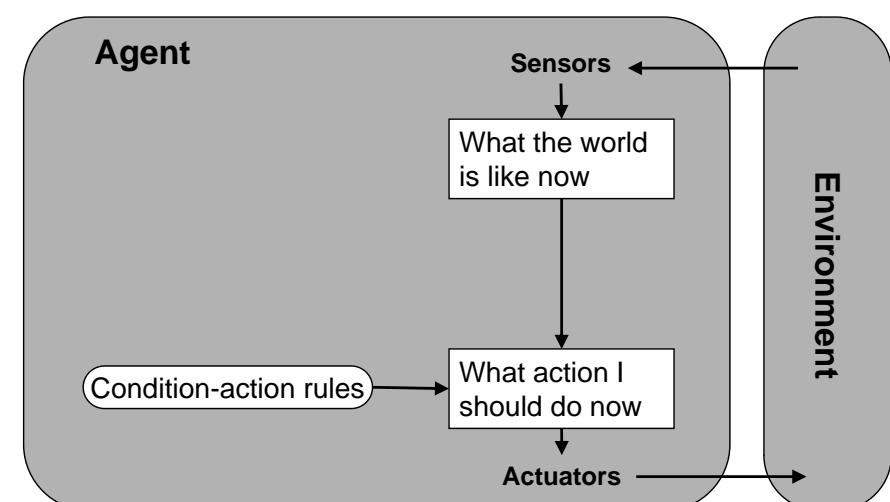
```
function REFLEX-VACCUM-AGENT( [location, status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

- شامل قوانین شرط-عمل مانند:
  - "اگر چراغ ترمز اتوموبیل جلویی روشن شد، آنگاه ترمز کن"

N. Razavi- AI course- 2005

27

## ساختار عامل های واکنشی ساده



N. Razavi- AI course- 2005

28

## برنامه عامل واکنشی ساده

```
function SIMPLE-REFLEX-AGENT( percept) returns an action
  static: rules, a set of condition-action rules

  state  $\leftarrow$  INTERPRET-INPUT( percept)
  rule  $\leftarrow$  RULE-MATCH( state, rules)
  action  $\leftarrow$  RULE-ACTION[ rule]
  return action
```

N. Razavi- AI course- 2005

29

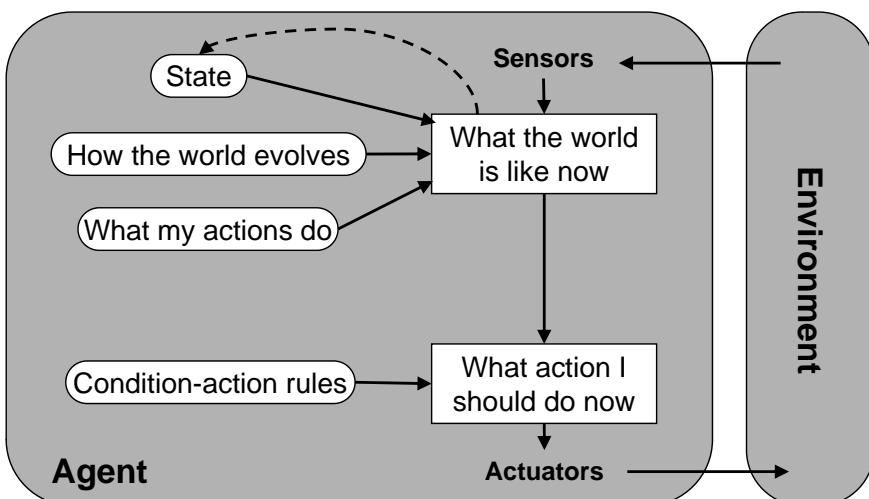
## عامل های واکنشی مبتنی بر مدل (حافظه دار)

- عامل واکنشی ساده در صورتی کار می کند که محیط کاملا قابل مشاهده باشد
- اگر محیط مشاهده پذیر جزئی باشد، پیگیری تغییرات دنیا لازم است
- مثال: تاکسی اتوماتیک
- مستلزم دو نوع دانش
  - نحوه تغییر دنیا
  - تاثیر اعمال عامل بر دنیا

N. Razavi- AI course- 2005

30

## عامل های واکنشی مبتنی بر مدل



N. Razavi- AI course- 2005

31

## برنامه عامل های واکنشی مبتنی بر مدل

```
function REFLEX-AGENT-WITH-STATE( percept) returns an action
  static: state, a description of the current world state
          rules, a set of condition-action rules
          action, the most recent action, initially none

  state  $\leftarrow$  UPDATE-STATE( state, action, percept)
  rule  $\leftarrow$  RULE-MATCH( state, rules)
  action  $\leftarrow$  RULE-ACTION[ rule]
  return action
```

N. Razavi- AI course- 2005

32

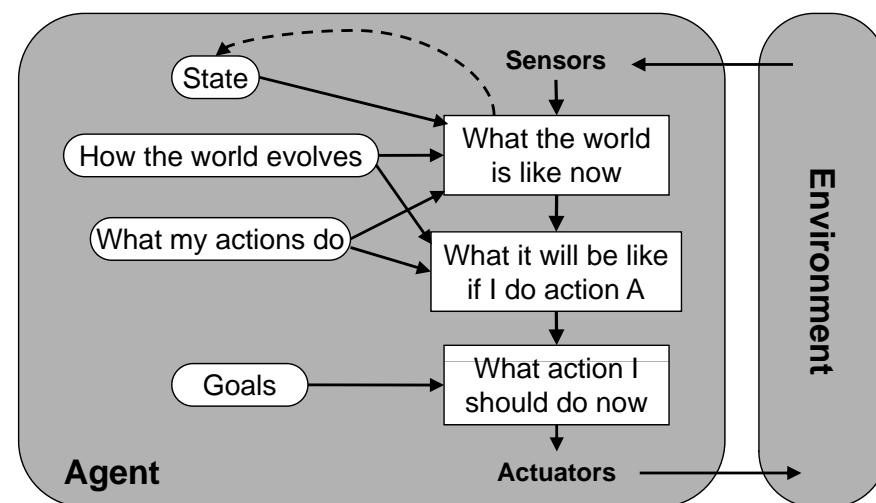
## عامل های مبتنی بر هدف

- اطلاعات لازم برای تصمیم گیری در مورد عملی که باید انجام شود:
  - اطلاعات مربوط به حالت فعلی
  - اطلاعات **هدف** (توصیف موقعیت مطلوب)
  - مثال: عمل مناسب برای تاکسی اتوماتیک در یک چهار راه کدام است؟ (بالا، پایین چپ، راست)
- اگر برای رسیدن به هدف نیاز به چندین عمل باشد
  - جستجو (search)
  - برنامه ریزی (planning)

N. Razavi- AI course- 2005

33

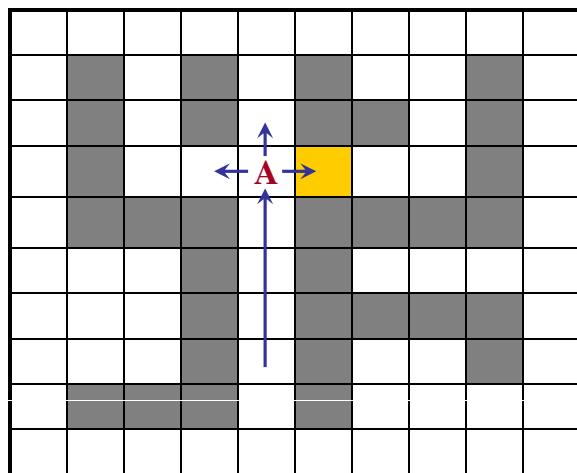
## عامل های مبتنی بر هدف



N. Razavi- AI course- 2005

34

## مثال: عامل هدف گرا



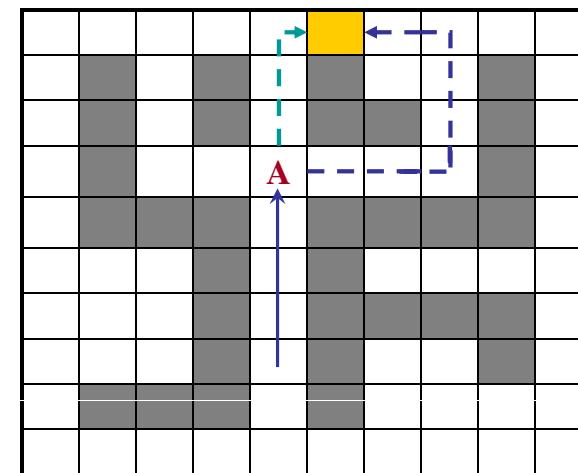
N. Razavi- AI course- 2005

35

## مثال: عامل هدف گرا

[UP, UP, UP, RIGHT]

[RIGHT, RIGHT, RIGHT, UP, UP, UP, LEFT, LEFT]



N. Razavi- AI course- 2005

36

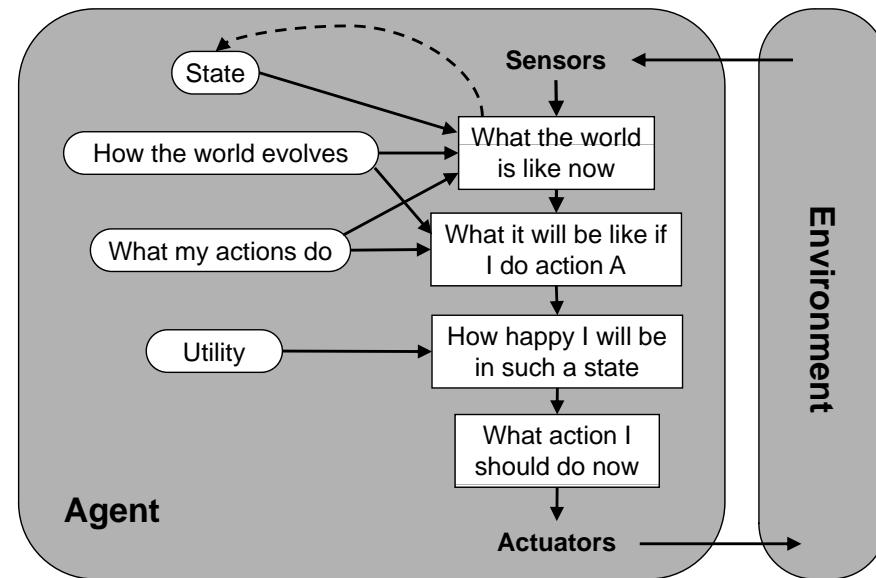
## عامل های سودمند

- در بسیاری از محیط ها اهداف برای تولید رفتاری با کیفیت بالا مناسب نیستند
- مثال: تاکسی اتوماتیک
  - ممکن است چندین مسیر برای رسیدن به مقصد موجود باشد، اما بعضی از آنها سریعتر، امن تر، مطمئن تر و یا ارزانتر از بقیه می باشند
  - اهداف ملاکی خام برای توصیف وضعیت ها هستند (مطلوب و نامطلوب)
- تابع سودمندی:** حالت ( یا دنباله ای از حالات) را به یک عدد حقیقی نگاشت می کند که درجه مطلوبیت آن را توصیف می کند
- امکان تصمیم گیری در مواردی که:
  - اهداف متناقض باشند
  - چندین هدف وجود دارد ولی رسیدن به هیچ یک قطعی نیست

N. Razavi- AI course- 2005

37

## عامل های هبتنی بلا سودمندی



N. Razavi- AI course- 2005

38

## عامل های یادگیرنده

- تورینگ (۱۹۵۰): ایده برنامه نویسی واقعی هوشمند به صورت دستی
  - ← نیاز به روش های سریعتر
  - ← ساخت ماشین های یادگیرنده و آموزش به آنها
- مؤلفه های عامل یادگیرنده
  - عنصر یادگیرنده: برای ایجاد بهبود
  - عنصر کارآیی: انتخاب فعالیت های خارجی
  - منتقد: تولید بازخورد با توجه به استاندارد کارآیی برای عنصر یادگیرنده
  - مولد مساله: پیشنهاد فعالیت های اکتشافی
- مثال: تاکسی اتوماتیک
  - عنصر کارآیی: حرکت سریع از خط ۳ به خط ۱
  - منتقد: دریافت شکایت راننده های دیگر
  - ایجاد قانونی بیانگر بد بودن این عمل و اصلاح عنصر کارآیی

N. Razavi- AI course- 2005

39

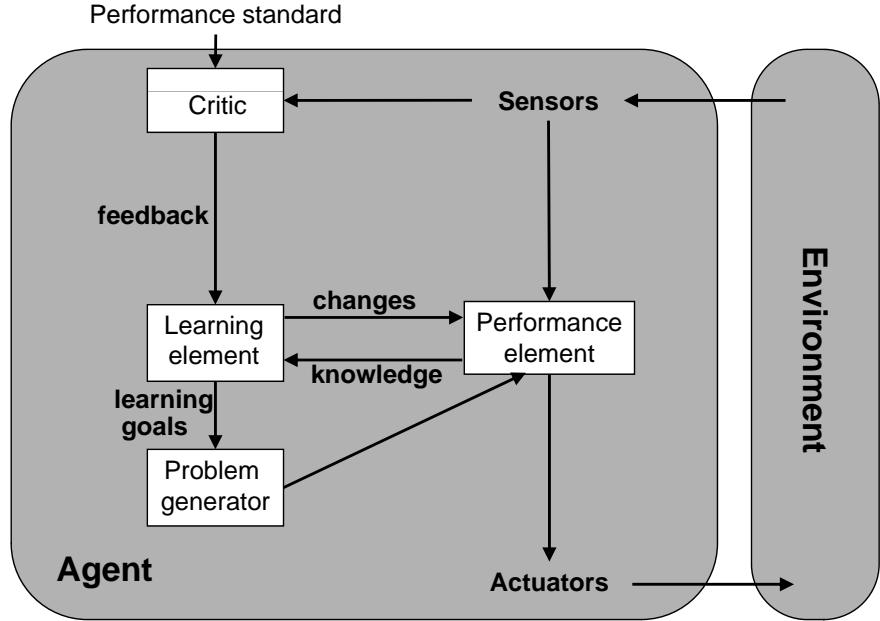
## عامل های یادگیرنده

- انواع دانشی که عنصر یادگیرنده می تواند یاد بگیرد:
  - یادگیری مستقیم از دنباله ادراکی
  - یادگیری نحوه تغییرات دنیا: مشاهده دو حالت متوالی
  - یادگیری در مورد تاثیر عمل عامل: مشاهده نتایج فعالیت عامل
- مثال: نحوه ترمز کردن در جاده های خیس
- پاداش و جریمه

N. Razavi- AI course- 2005

40

## عامل‌های یادگیری



N. Razavi- AI course- 2005

41

## مقدمه

- عامل های حل مسئله
- انواع مسئله
- فرموله سازی مسئله
- مسائل نمونه
- الگوریتم های ابتدایی جستجو

# حل مسائل توسط جستجو

فصل سوم

سید ناصر رضوی

Email: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۴

N.Razavi- AI course-2005

2

## عامل های حل مسئله

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
         state, some description of the current world state
         goal, a goal, initially null
         problem, a problem formulation
  state  $\leftarrow$  UPDATE-STATE(state, percept)
  if seq is empty then do
    goal  $\leftarrow$  FORMULATE-GOAL(state)
    problem  $\leftarrow$  FORMULATE-PROBLEM(state, goal)
    seq  $\leftarrow$  SEARCH(problem)
    action  $\leftarrow$  FIRST(seq)
    seq  $\leftarrow$  REST(seq)
  return action
```

فرضیات در مورد محیط: ایستا، قابل مشاهده، گسسته و قطعی

N.Razavi- AI course-2005

3

## مثال: رومانی

- یک روز تعطیل در رومانی؛ مکان فعلی شهر آراد
- پرواز فردا، بخارست را ترک می کند.
- فرموله سازی هدف:
  - بودن در بخارست
- فرموله سازی مسئله:
  - **حالت ها:** شهرهای مختلف
  - **عملیات:** رفتن از شهری به شهر دیگر
- یافتن پاسخ:
  - دنباله ای از شهرها، مانند:

Arad  $\rightarrow$  Sibiu  $\rightarrow$  Fagaras  $\rightarrow$  Bucharest

N.Razavi- AI course-2005

4

## انواع مسئله

- قطعی، کاملا مشاهده پذیر  $\leftarrow$  مسائل تک - حالت

- عامل دقیقا می داند در چه حالتی خواهد بود؛ راه حل یک دنباله می باشد.

- قطعی، مشاهده پذیر جزئی  $\leftarrow$  مسائل چند-حالت

- ممکن است عامل ایده ای درباره اینکه کجاست نداشته باشد؛ راه حل یک دنباله است.

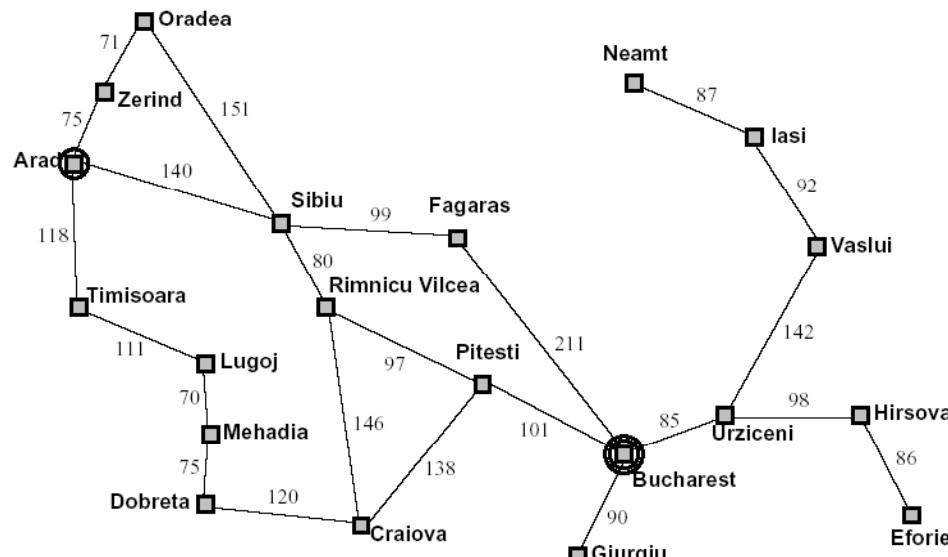
- غیر قطعی و/یا مشاهده پذیر جزئی  $\leftarrow$  مسائل احتمالی

- ادراک اطلاعات جدیدی درباره حالت فعلی فراهم می کند.

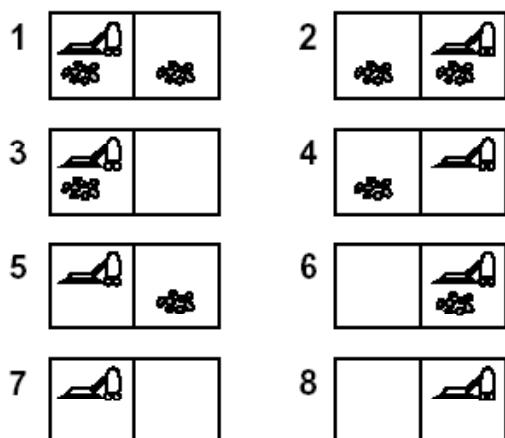
- در حین اجرا باید از حسگرها استفاده کند.

- راه حل به صورت یک درخت

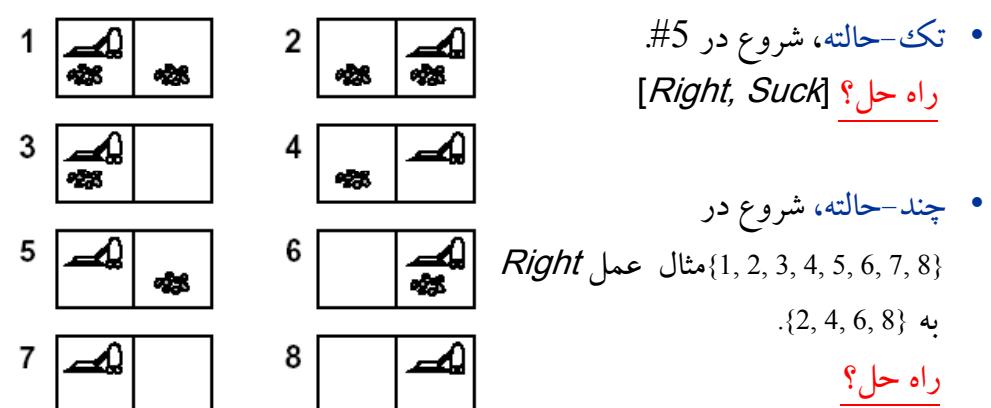
- اغلب جستجو و اجرا به صورت یک در میان (interleave) (online)  $\leftarrow$  مسائل اکتشافی



## مثال: دنیای مکش



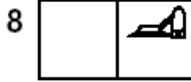
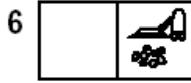
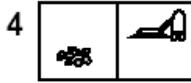
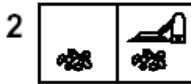
- تک-حالت، شروع در #5.  
راه حل؟



- تک-حالت، شروع در #5.  
راه حل؟ [Right, Suck]

- چند-حالت، شروع در {1, 2, 3, 4, 5, 6, 7, 8} عمل Right به {2, 4, 6, 8}.  
راه حل؟

## مثال: دنیای مکش



- چند-حالت، شروع در

*Right* {1, 2, 3, 4, 5, 6, 7, 8} مثال عمل

. {2, 4, 6, 8}

راه حل؟

[*Right, Suck, Left, Suck*]

احتمالی

- غیر قطعی: مکش می تواند یک فرش تمیز را کنیف کند.
- در گ محلی: گرد و خاک در محل فعلی
- ادراک: [L, Clean] #5 یعنی شروع در #5 یا #7

راه حل؟

[*Right, if dirt then Suck*]

N.Razavi- AI course-2005

9

## فرموله سازی مسائل تک-حالت

یک مسئله بوسیله چهار مورد تعریف می شود:

۱. حالت اولیه مثلاً بودن در شهر Arad

۲. عمل ها یا تابع حالت بعدی

(x) = S = مجموعه ای از زوج های عمل-حالت

•  $S(\text{Arad}) = \{\langle \text{Arad} \rightarrow \text{Zerind}, \text{Zerind} \rangle, \dots\}$

۳. تابع تست هدف

x = "at Bucharest": صریح

- ضمیم: NoDirt(x)

۴. تابع هزینه مسیر:

- مثال: مجموع فواصل، تعداد عمل های انجام شده و ...

- هزینه گام (Step cost):  $c(x, a, y) \geq 0$

• راه حل: دنباله ای از عملیات که از حالت اولیه شروع و به حالت هدف ختم می شود.

N.Razavi- AI course-2005

10

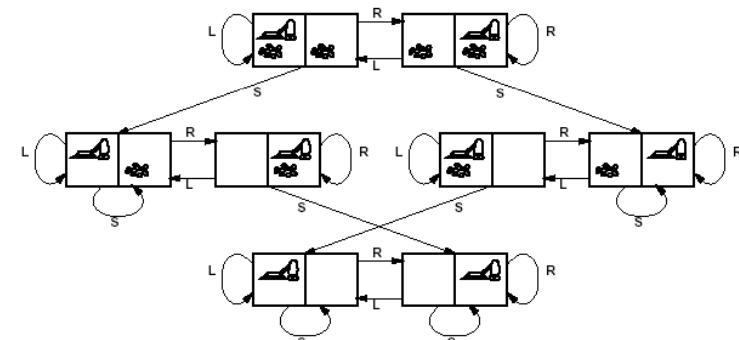
## انتخاب یک فضای حالت

- دنیای واقعی به شدت پیچیده می باشد
  - ← بنابراین، برای حل مسئله باید فضای حالت انتزاعی باشد.
- حالت (انتزاعی) = مجموعه ای از حالت های واقعی
- عمل (انتزاعی) = ترکیبی پیچیده از عمل های واقعی
  - مثلاً عمل Arad  $\rightarrow$  Zerind مجموعه ای پیچیده از اعمال باشد.
  - راه حل (انتزاعی)
    - مجموعه ای از مسیرهای واقعی که در دنیای واقعی راه حل می باشند.
    - هر عمل انتزاعی باید از مسئله اصلی ساده تر باشد!

N.Razavi- AI course-2005

11

## مثال: گراف فضای حالت دنیای مکش



• حالات؟

• اعمال؟

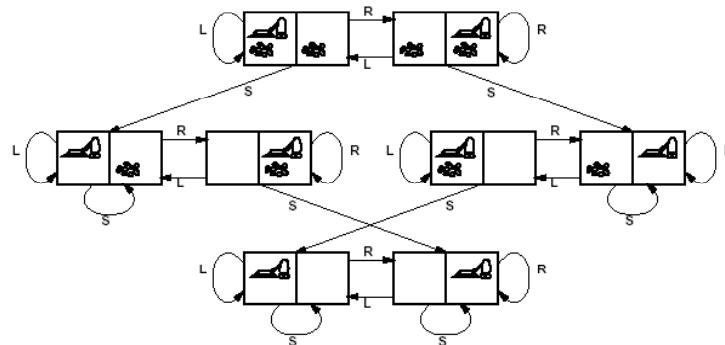
• تست هدف؟

• هزینه مسیر؟

N.Razavi- AI course-2005

12

## مثال: گراف فضای حالت دنیای مکش



- حالات؟ وجود گرد و خاک و مکان های عامل (بدون در نظر گرفتن مقدار گرد و خاک)
- اعمال؟ Left, Right, Suck
- تست هدف؟ نبودن گرد و خاک
- هزینه مسیر؟ بازاء هر عمل ۱

N.Razavi- AI course-2005

13

## مثال: محمای هشت

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- حالات؟
- اعمال؟
- تست هدف؟
- هزینه مسیر؟

N.Razavi- AI course-2005

14

## مثال: محمای هشت

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

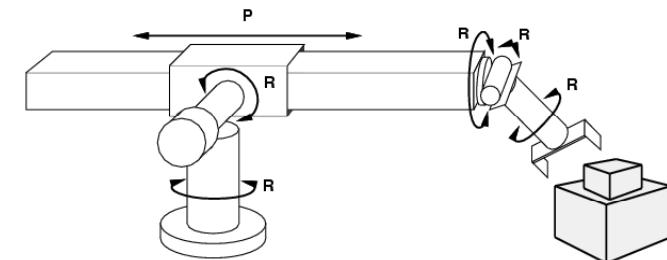
- حالات؟ اعداد صحیح یانگر محل کاشی ها
- اعمال؟ حرکت خانه خالی به چپ، بالا، راست و پایین
- تست هدف؟ حالت هدف (داده شده)
- هزینه مسیر؟ بازاء هر حرکت ۱

توجه: راه حل بهینه خانواده معماهای 7 یک مسئله NP-hard می باشد

N.Razavi- AI course-2005

15

## مثال: ربات اسمنبل کننده



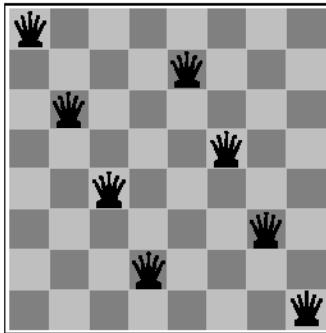
- حالات؟ زاویه مفاصل روبات، مختصات قطعات
- اعمال؟ حرکت پیوسته مفاصل روبات
- تست هدف؟ سرهم بندی کامل
- هزینه مسیر؟ زمان اجرا

N.Razavi- AI course-2005

16

## مسئله هشت وزیر

- قراردادن هشت وزیر در صفحه شطرنج به طوریکه هیچ وزیری نتواند به وزیر دیگری حمله کند.



- آزمون هدف:** ۸ وزیر روی صفحه شطرنج که با هم برخورد ندارند.
- هزینه مسیر:** صفر
- حالات:** ترتیب ۸ وزیر هر کدام در یک ستون مثال روبرو: {8, 6, 4, 2, 7, 5, 3, 1}
- عملگرهای انتقال:** یک وزیر دارای برخورد به مریع دیگری در همان ستون

N.Razavi- AI course-2005

17

## الگوریتم های جستجوی درخت

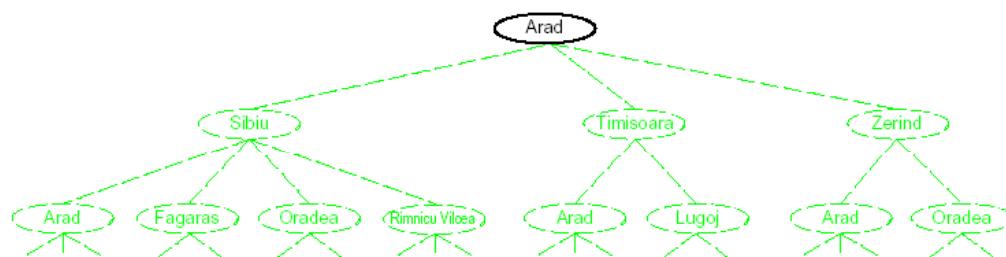
- ایده اصلی:** کاوش Offline و شبیه سازی شده فضای حالت بوسیله تولید حالات بعدی حالت هایی که تا کنون تولید شده اند.

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```

N.Razavi- AI course-2005

18

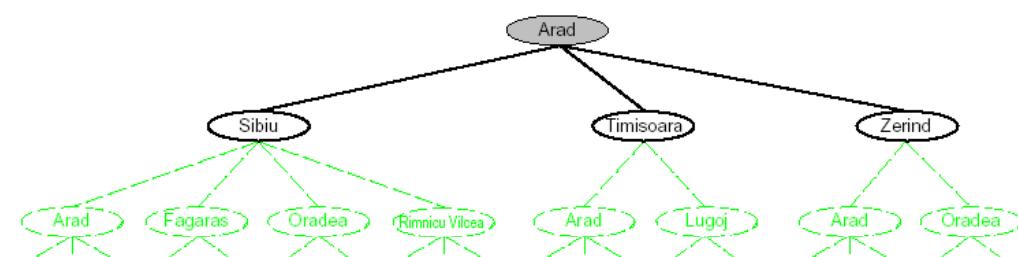
## مثال جستجوی درخت



N.Razavi- AI course-2005

19

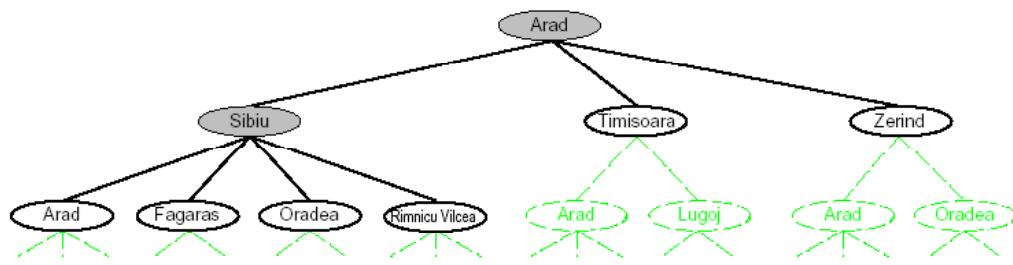
## مثال جستجوی درخت



N.Razavi- AI course-2005

20

## مثال جستجوی درخت



N.Razavi- AI course-2005

21

## پیاده سازی: جستجوی عمومی درخت

```

function TREE-SEARCH(problem, fringe) returns a solution, or failure
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
    fringe  $\leftarrow$  INSERT ALL(EXPAND(node, problem)), fringe

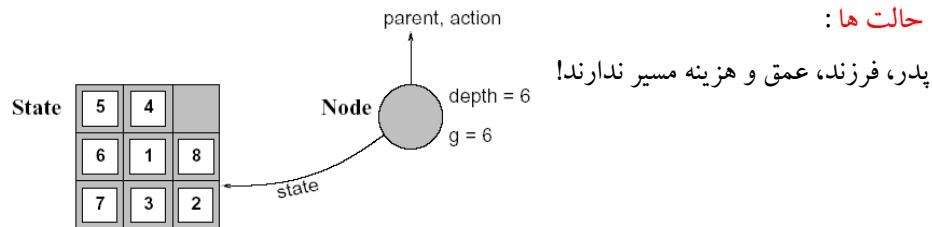
function EXPAND( node, problem) returns a set of nodes
  successors  $\leftarrow$  the empty set
  for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
    s  $\leftarrow$  a new NODE
    PARENT-NODE[s]  $\leftarrow$  node; ACTION[s]  $\leftarrow$  action; STATE[s]  $\leftarrow$  result
    PATH-COST[s]  $\leftarrow$  PATH-COST[node] + STEP-COST(node, action, s)
    DEPTH[s]  $\leftarrow$  DEPTH[node] + 1
    add s to successors
  return successors
  
```

N.Razavi- AI course-2005

22

## پیاده سازی: حالت و گره

- یک **حالت** (بیانگر) یک پیکره بندی فیزیکی می باشد
- یک **گره** یک ساختار داده ای تشکیل دهنده بخشی از درخت جستجو شامل: **پدر**، **فرزندها**، **عمق** و **هزینه مسیر** (*X*) **(*g*)** است.
- **حالت ها :** پدر، فرزند، عمق و هزینه مسیر ندارند!



- تابع EXPAND گره های جدید ایجاد می کند، فیلهای مختلف را مقدار می دهد و با استفاده از تابع SUCCESSORS-FN مسئله، حالت های مربوطه ایجاد می شود.

N.Razavi- AI course-2005

23

## استراتژی های جستجو

- یک استراتژی بواسیله **ترتیب گسترش گره ها** تعریف می شود.
- بعد از زیبایی استراتژی ها:

  - **کامل بودن** - آیا در صورت وجود راه حل، همیشه راه حلی پیدا می کند؟
  - **پیچیدگی زمانی** - تعداد گره های تولید شده / گسترش یافته
  - **پیچیدگی حافظه** - حداکثر تعداد گره ها در حافظه
  - **بهینگی** - آیا همیشه کم هزینه ترین راه حل را پیدا می کند؟

- **پیچیدگی زمان و فضا بر حسب پارامترهای زیر سنجیده می شوند:**
- **b** : حداکثر فاکتور انشعاب درخت جستجو
- **d** : عمق کم هزینه ترین راه حل
- **m** : حداکثر عمق فضای حالت (ممکن است  $\infty$  باشد)

N.Razavi- AI course-2005

24

## استراتژی های جستجوی ناآگاهانه

- استراتژی های **ناآگاهانه** تنها از اطلاعات موجود در تعریف مسئله استفاده می کنند.

- جستجوی اول-سطح (BFS)
- جستجوی هزینه-یکنواخت (UCS)
- جستجوی اول-عمق (عمقی) (DFS)
- جستجوی با عمق محدود (DLS)
- جستجوی عمیق کننده تکراری (IDS)

N.Razavi- AI course-2005

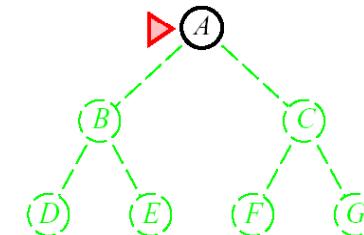
25

## جستجوی سطحی

- هر بار سطحی ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:**

- یک صف FIFO می باشد. یعنی، فرزندان جدید به انتهای صف اضافه می شوند.

0: [A]



N.Razavi- AI course-2005

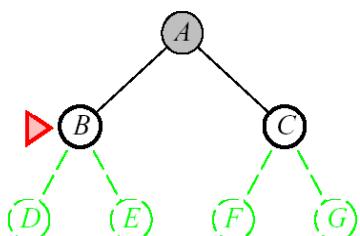
26

## جستجوی سطحی

- هر بار سطحی ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:**

- یک صف FIFO می باشد. یعنی، فرزندان جدید به انتهای صف اضافه می شوند.

1: [B, C]



N.Razavi- AI course-2005

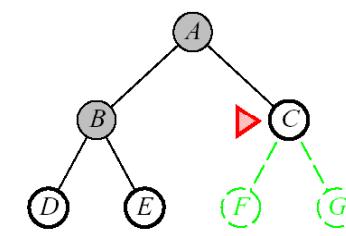
27

## جستجوی سطحی

- هر بار سطحی ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:**

- یک صف FIFO می باشد. یعنی، فرزندان جدید به انتهای صف اضافه می شوند.

2: [C, D, E]



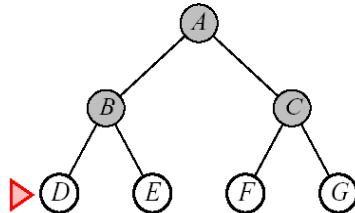
N.Razavi- AI course-2005

28

## جستجوی سطحی

- هر بار سطحی ترین گره گسترش نیافته را گسترش می دهد.
- **پیاده سازی:** یک صف FIFO می باشد. یعنی، فرزندان جدید به انتهای صف اضافه می شوند.

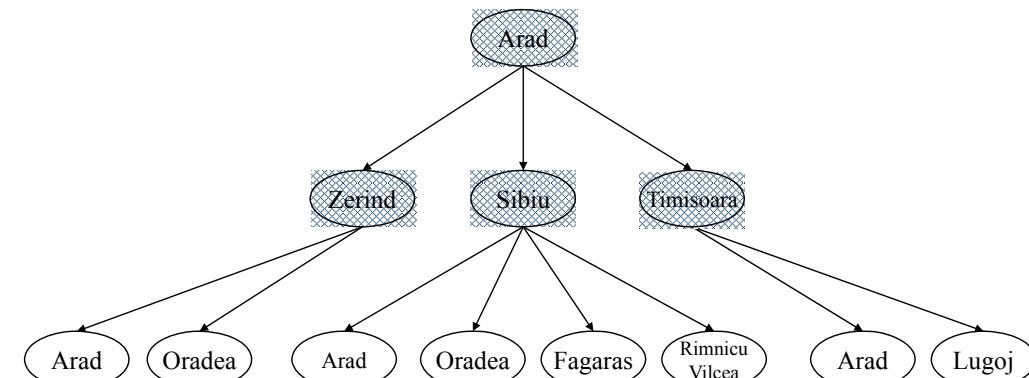
3: [D, E, F, G]



N.Razavi- AI course-2005

29

## مثال: جستجوی سطحی



N.Razavi- AI course-2005

30

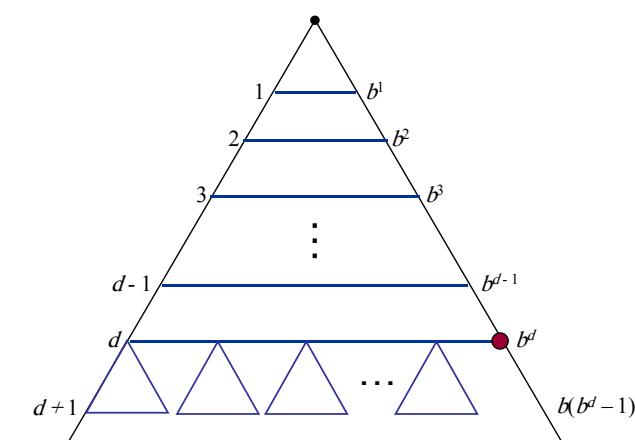
## خصوصیات جستجوی سطحی

- کامل؟ بله (به شرط محدود بودن  $b$ )
- $b + b^2 + \dots + b^d + b(b^d-1) = O(b^{d+1})$  پیچیدگی زمانی؟
- پیچیدگی حافظه؟  $O(b^{d+1})$  چون همه گره ها را در حافظه نگه می دارد
- بهینه؟ بله (مثلا اگر بازاء هر عمل، هزینه  $= 1$ )
- مشکل اصلی حافظه می باشد. (نسبت به زمان)

N.Razavi- AI course-2005

31

## پیچیدگی زمانی و حافظه جستجوی سطحی



$$b + b^2 + \dots + b^d + b(b^d-1) = O(b^{d+1})$$

N.Razavi- AI course-2005

32

## زمان و فضای لازم در جستجوی سطحی

حافظه	زمان	تعداد گره ها	عمق
۱ مگابایت	۱۱/۰ ثانیه	۱۱۰۰	۲
۱۰۶ مگابایت	۱۱ ثانیه	۱۱۱۱۰۰	۴
۱۰ گیگابایت	۱۹ دقیقه	۱۰۷	۶
۱ ترابایت	۳۱ ساعت	۱۰۹	۸
۱۰۱ ترابایت	۱۲۹ روز	۱۰۱۱	۱۰
۱۰ پتا بایت	۳۵ سال	۱۰۱۳	۱۲
۱ هیگرا بایت	۳۵۲۳ سال	۱۰۱۵	۱۴

$$b = 10 \quad \square$$

۱۰۰۰ گره در هر ثانیه

هر گره ۱۰۰۰ بایت

N.Razavi- AI course-2005

33

## جستجوی هزینه-یکنواخت

- هربار کم هزینه ترین گره گسترش نیافرته را گسترش می دهد.
- پیاده سازی:

$\text{fringe}$  = صفتی که براساس هزینه مسیر مرتب شده باشد.  
معادل جستجوی سطحی اگر هزینه گام ها مساوی باشند.

کامل؟ بله اگر هزینه گامها  $\leq \epsilon$   
پیچیدگی زمانی؟ تعداد گره هایی که هزینه مسیر آنها کوچکتر یا مساوی هزینه راه حل بهینه باشد.

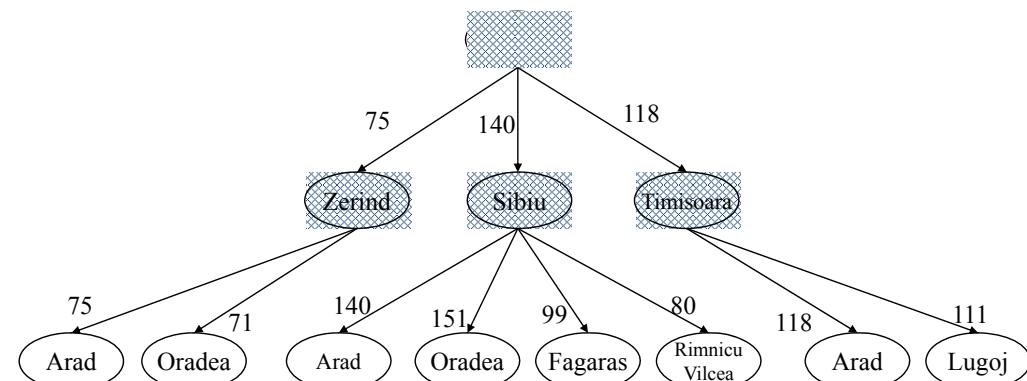
$$O(b^{\lceil C^{*/\epsilon} \rceil})$$

پیچیدگی حافظه؟ مانند پیچیدگی زمانی  
بهینه؟ بله (گره ها به ترتیب صعودی ( $n$ ) گسترش می یابند).

N.Razavi- AI course-2005

34

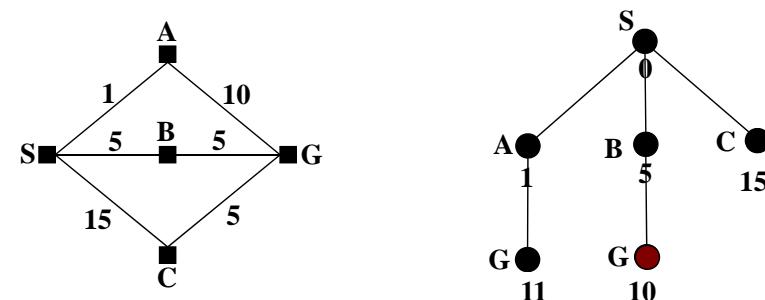
## مثال: جستجوی هزینه یکنواخت



N.Razavi- AI course-2005

35

## مثال: جستجوی هزینه یکنواخت



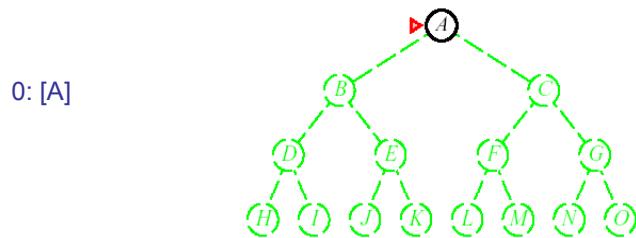
- 0: [S(0)]
- 1: [A(1), B(5), C(15)]
- 2: [B(5), G(11), C(15)]
- 3: [G(10), G(11), C(15)]
- 4: [G(11), C(15)]

N.Razavi- AI course-2005

36

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO = پشته fringe

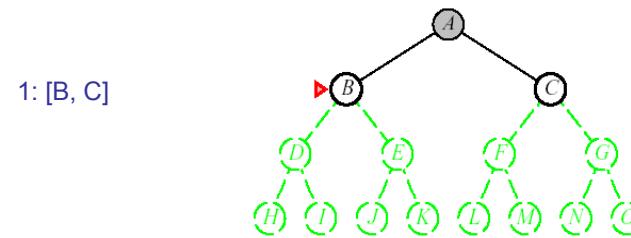


N.Razavi- AI course-2005

37

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO = پشته fringe

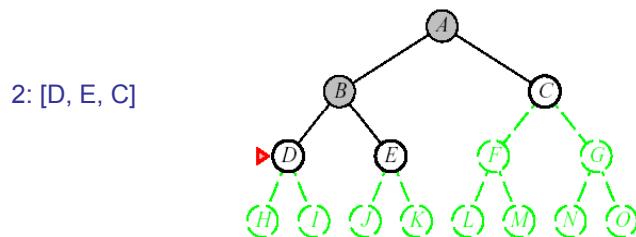


N.Razavi- AI course-2005

38

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO = پشته fringe

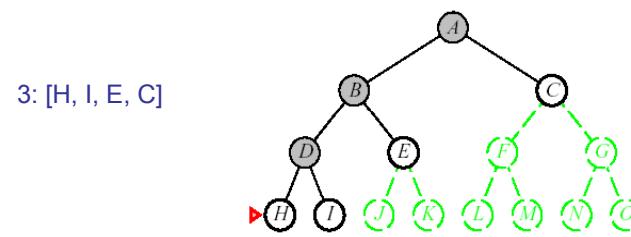


N.Razavi- AI course-2005

39

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO = پشته fringe

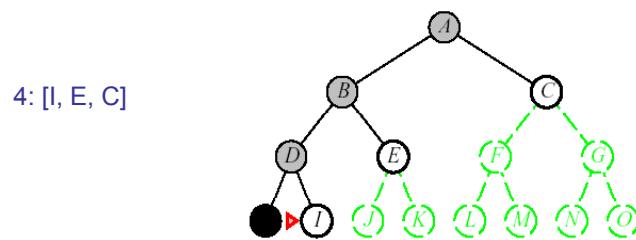


N.Razavi- AI course-2005

40

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
LIFO = fringe - پشته، فرزندان جدید را در ابتدا درج می کند.

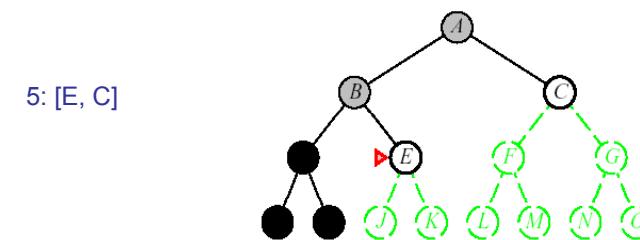


N.Razavi- AI course-2005

41

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
LIFO = fringe - پشته، فرزندان جدید را در ابتدا درج می کند.

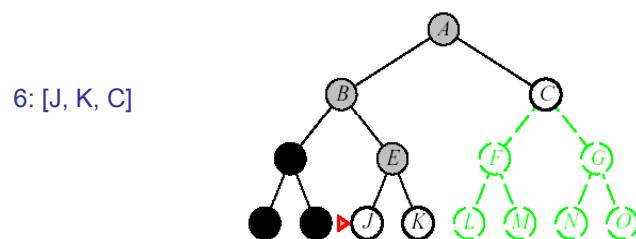


N.Razavi- AI course-2005

42

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
LIFO = fringe - پشته، فرزندان جدید را در ابتدا درج می کند.

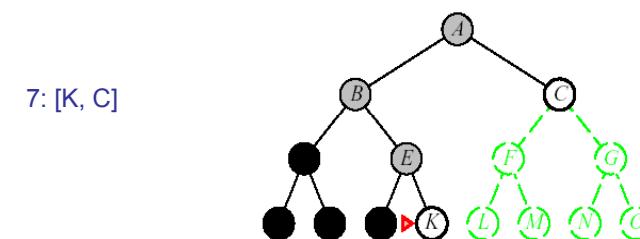


N.Razavi- AI course-2005

43

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
LIFO = fringe - پشته، فرزندان جدید را در ابتدا درج می کند.



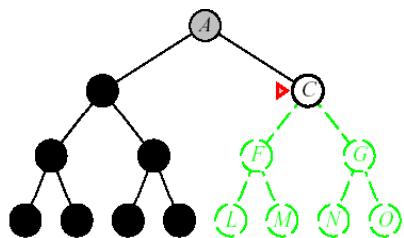
N.Razavi- AI course-2005

44

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO، فرزندان جدید را در ابتدا درج می کند.

8: [C]



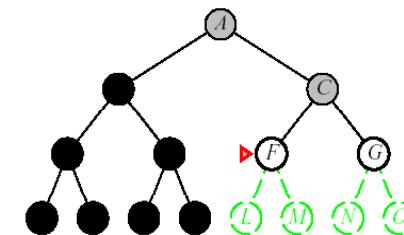
N.Razavi- AI course-2005

45

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO، فرزندان جدید را در ابتدا درج می کند.

9: [F, G]



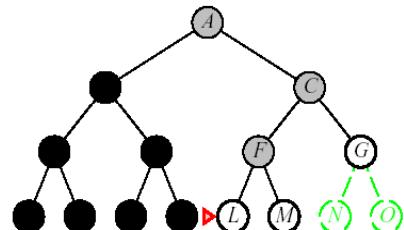
N.Razavi- AI course-2005

46

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO، فرزندان جدید را در ابتدا درج می کند.

10: [L, M, G]



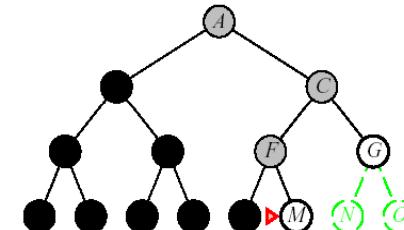
N.Razavi- AI course-2005

47

## جستجوی عمقی

- هر بار عمیق ترین گره گسترش نیافته را گسترش می دهد.
- پیاده سازی:  
– LIFO، فرزندان جدید را در ابتدا درج می کند.

11: [M, G]



N.Razavi- AI course-2005

48

## خصوصیات جستجوی عمقی

• کامل؟

- خیر (در فضاهای حالت با عمق نامحدود، دارای حلقه)
- برای اجتناب از حالات تکراری در طول مسیر، نیاز به اصلاح دارد.
- بنابراین، در فضای حالت محدود کامل است.

• پیچیدگی زمانی؟  $O(b^m)$

- در بدترین حالت تمام گره های درخت جستجو تولید می شوند
- اگر  $m$  خیلی بیشتر از  $d$  باشد، بسیار زیاد
- اگر تعداد راه حل ها زیاد باشد، می تواند بسیار سریعتر از جستجوی سطحی باشد

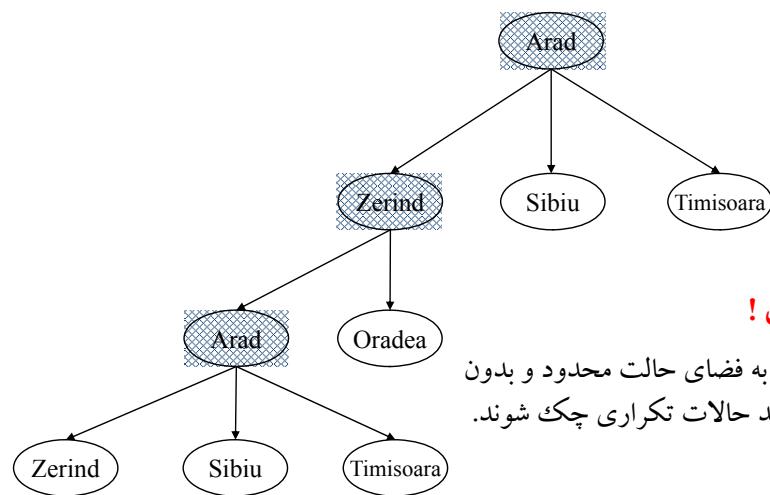
• پیچیدگی حافظه؟

- $O(bm)$ ، به صورت خطی!

• برهنه؟ خیر

N.Razavi- AI course-2005

50



N.Razavi- AI course-2005

49

## جستجوی با عمق محدود

= جستجوی عمقی با محدوده عمقی  $l$

یعنی، فرزندان گره های واقع در عمق  $l$  تولید نخواهند شد.

- در این استراتژی با در نظر گرفتن یک محدوده عمقی مانند  $l$  از به دام افتادن جستجوی عمقی در یک حلقه بی پایان جلوگیری می شود. (برش روی درخت جستجو)

- مثلا در نقشه رومانی چون ۲۰ شهر وجود دارد بنابراین طول راه حل باید حداقل ۱۹ باشد.

- بنابراین هیچ وقت گره ای با عمق بیش از ۱۹ بررسی نخواهد شد.
- اگر در محدوده عمقی  $l$  راه حلی وجود داشته باشد، بالاخره پیدا خواهد شد، اما هیچ تضمینی برای یافتن راه حل بهینه وجود ندارد.

## جستجوی با عمق محدود

• کامل؟

- بله (اگر  $d \leq l$ )

• پیچیدگی زمانی؟

- $O(bl)$

• پیچیدگی حافظه؟

- $O(bl)$

• برهنه؟

- خیر

N.Razavi- AI course-2005

52

N.Razavi- AI course-2005

51

## جستجوی با عمق محدود پیاده سازی بازگشتی

```

function DEPTH-LIMITED-SEARCH( problem, limit) returns soln/fail/cutoff
    RECURSIVE-DLS(MAKE-NODE(INITIAL-STATE[problem]), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns soln/fail/cutoff
    cutoff-occurred? ← false
    if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
    else if DEPTH[node] = limit then return cutoff
    else for each successor in EXPAND(node, problem) do
        result ← RECURSIVE-DLS(successor, problem, limit)
        if result = cutoff then cutoff-occurred? ← true
        else if result ≠ failure then return result
    if cutoff-occurred? then return cutoff else return failure

```

N.Razavi- AI course-2005

53

## جستجوی عمیق کننده تکراری

```

function ITERATIVE-DEEPENING-SEARCH( problem) returns a solution, or fail-
ure
    inputs: problem, a problem
    for depth ← 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH( problem, depth)
        if result ≠ cutoff then return result

```

N.Razavi- AI course-2005

55

## جستجوی عمیق کننده تکراری

- مشکل اصلی در جستجوی عمیق با عمق محدود شده (DLS) **انتخاب یک محدوده عمیق مناسب** است.
- در نقشه رومانی طول بزرگترین مسیر بین دو شهر ۹ می باشد(قطر)، و این محدوده عمیق مناسب تر از ۱۹ می باشد. اما در بیشتر فضاهای حالت انتخاب محدوده مناسب قبل از حل مسئله میسر نمی باشد.
- جستجوی عمیق کننده تکراری روی برای تعیین محدوده عمیق مناسب با امتحان کردن تمامی محدوده ها ( از صفر به بالا) می باشد. یعنی اول عمق صفر، بعد عمق ۱، بعد عمق ۲ و ...

N.Razavi- AI course-2005

54

## جستجوی عمیق کننده تکراری

- جستجوی عمیق کننده تکراری مزایای **جستجوی سطحی و عمیقی** را با هم ترکیب می کند:
  - مانند جستجوی سطحی **کامل** (اگر فاکتور انشعاب محدود باشد) و **بهینه** (اگر هزینه مسیر یک تابع غیر نزولی بر حسب عمق باشد) است.
  - جستجوی عمیق دارای **صرف حافظه خطی**  $O(bd)$  می باشد.

- این جستجو از نظر پیچیدگی زمانی مانند جستجوی محدود شده می باشد، به **جز اینکه برخی حالات چند بار بسط داده** می شوند.

N.Razavi- AI course-2005

56

## جستجوی عمیق کننده تکرای ( $I=0$ )

Limit = 0

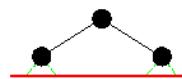
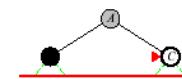
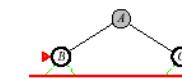
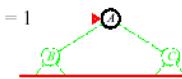


N.Razavi- AI course-2005

57

## جستجوی عمیق کننده تکرای ( $I=1$ )

Limit = 1

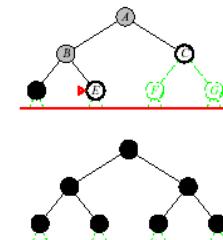
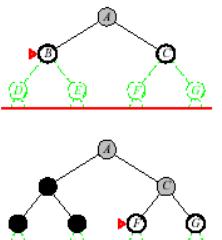
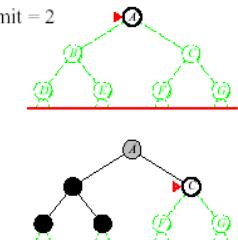


N.Razavi- AI course-2005

58

## جستجوی عمیق کننده تکرای ( $I=2$ )

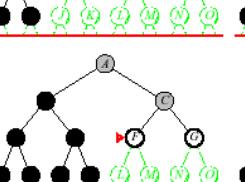
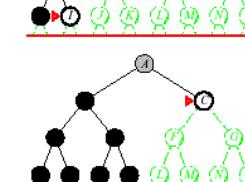
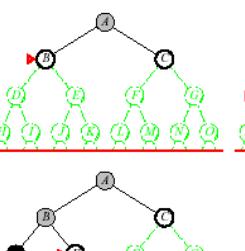
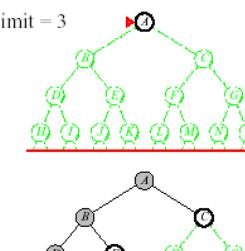
Limit = 2



59

## جستجوی عمیق کننده تکرای ( $I=3$ )

Limit = 3



N.Razavi- AI course-2005

60

## فواص جستجوی عمیق کننده تکراری

- کامل؟! بله (مانند جستجوی سطحی)
- پیچیدگی زمانی؟!

$$db^1 + (d-1)b^2 + \dots + b^d = O(b^d)$$

- پیچیدگی حافظه  $O(bd)$ ؟!
- بهینه؟! بله، (مانند جستجوی سطحی)

- می تواند برای کاوش درخت جستجوی هزینه یکنواخت اصلاح شود!!!

N.Razavi- AI course-2005

61

## IDS

- تعداد گره های تولید شده توسط DLS در عمق  $d$  با فاکتور انشعاب  $b$ :
- $$N_{DLS} = b + b^2 + \dots + b^{d-1} + b^d$$
- تعداد گره های تولید شده توسط IDS در عمق  $d$  با فاکتور انشعاب  $b$ :

$$N_{IDS} = db^1 + (d-1)b^2 + \dots + 2b^{d-1} + b^d$$

- اگر  $b = 10$  و  $d = 5$

$$N_{DLS} = 10 + 100 + 1,000 + 10,000 + 100,000 = 111,110$$

$$N_{IDS} = 50 + 400 + 3,000 + 20,000 + 100,000 = 123,450$$

- محاسبه میزان سربار:

$$((123450 - 111110)/111110) * 100 = 11\%$$

- با افزایش فاکتور انشعاب  $b$  میزان سربار کاهش می یابد.
- در بدترین حالت  $b = 2$ ، سربار 100% است و این جستجو دو برابر زمان می برد.

N.Razavi- AI course-2005

63

## پیمیدگی زمانی عمیق کننده تکراری

DLS ( $/= 0$ )	0	سربار $\leq N_{DLS} (/= d)$
DLS ( $/= 1$ )	$b^1$	
DLS ( $/= 2$ )	$b^1 + b^2$	
.	.	
.	.	
DLS ( $/= d-1$ )	$b^1 + b^2 + b^3 + \dots + b^{d-1}$	
DLS ( $/= d$ )	$b^1 + b^2 + b^3 + \dots + b^{d-1} + b^d$	

$$N_{IDS} = db^1 + (d-1)b^2 + (d-2)b^3 + \dots + 2b^{d-1} + b^d$$

N.Razavi- AI course-2005

62

## جستجوی دو طرفه

- ایده: انجام جستجو در دو جهت به طور همزمان
  - رو به جلو: از حالت اولیه به سمت حالت هدف
  - رو به عقب: از حالت هدف به سمت حالت اولیه
- انگیزه:  $b^{d/2} + b^{d/2}$  بسیار کمتر از  $b^d$  می باشد

- مثال: اگر راه حل یک مسئله در عمق  $d = 6$  باشد و  $b = 10$  آنگاه
  - جستجوی دو طرفه (در هر دو طرف جستجوی سطحی)  $\leftarrow 22,200$  گره
  - جستجوی سطحی  $\leftarrow 11,111,000$  گره

N.Razavi- AI course-2005

64

## جستجوی دوطرفه

- پیچیدگی زمانی:  $O(b^{d/2})$
- پیچیدگی حافظه:  $O(b^{d/2})$ 
  - به منظور بررسی تعلق حداقل یکی از درخت ها باید در حافظه نگهداری شود
  - مصرف حافظه نمایی، بزرگترین ضعف جستجوی دوطرفه می باشد
- کامل بودن و بهینگی (برای هزینه های گام یکسان):
  - اگر در هر دو طرف از جستجوی سطحی استفاده شود

N.Razavi- AI course-2005

65

## خلاصه الگوریتم ها

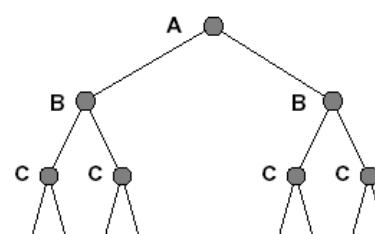
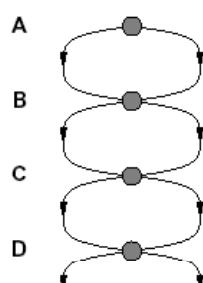
Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	$b^{d+1}$	$b^{\lceil C^*/\epsilon \rceil}$	$b^m$	$b^l$	$b^d$
Space	$b^{d+1}$	$b^{\lceil C^*/\epsilon \rceil}$	$bm$	$bl$	$bd$
Optimal?	Yes*	Yes	No	No	Yes*

N.Razavi- AI course-2005

66

## حالات تکرای

- شکست در تشخیص حالت های تکرای می تواند یک مسئله خطی را به یک مسئله نمایی تبدیل کند!



N.Razavi- AI course-2005

67

## جستجوی گراف

```

function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
  closed ← an empty set
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
    if STATE[node] is not in closed then
      add STATE[node] to closed
      fringe ← INSERTALL(EXPAND(node, problem), fringe)
  
```

N.Razavi- AI course-2005

68

## خلاصه

- فرموله سازی مسئله اغلب نیاز به انتزاع جزئیات مسئله دارد، تا بتوان فضای حالتی بدست آورده که به صورت مقرر و به صرفه ای قابل کاوش کردن و جستجو باشد.
- انواع استراتژی های ناآگاهانه وجود دارد.
- مصرف حافظه جستجوی عمیق کننده تکراری دارای مرتبه خطی می باشد و زمان خیلی بیشتری نسبت به سایر روش های ناآگاهانه مصرف نمی کند.

## مقدمه

- جستجوی اول-بهترین (Best-First Search)
- جستجوی حریصانه (Greedy search)
- جستجوی  $A^*$  و خصوصیات آن
- جستجوی عمیق کننده تکراری  $A^*$  و  $SMA^*$
- جستجوی اول بهترین بازگشتی (RBFA\*)
- هیوریستیک ها (Heuristics)
- الگوریتم های جستجوی محلی
- جستجوی تپه نوردی (Hill Climbing)
- جستجوی Simulated Annealing
- الگوریتم های ژنتیک

N. Razavi - AI course - 2007

2

## (وش) های جستجوی آگاهانه

فصل چهارم

سید ناصر رضوی

Email:[razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۶

## مشکلات (وش) های جستجوی نآگاهانه

- معیار انتخاب گره بعدی برای گسترش دادن تنها به شماره سطح آن بستگی دارد.
- از ساختار مسئله بهره نمی برند.
- درخت جستجو را به یک روش از پیش تعریف شده گسترش می دهند. یعنی قابلیت تطبیق پذیری با آنچه که تا کنون در مسیر جستجو دریافته اند و نیز حرکتی که می توانند خوب باشد ندارند.

## ۵(۹): جستجوی درخت

```
function GENERAL-SEARCH( problem, strategy) returns a solution , or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains the goal state then return the corresponding solution
        else expand the node and add the resulting node to the search tree
    end
```

- استراتژی توسط ترتیب گسترش یافتن گره ها تعریف می شود.

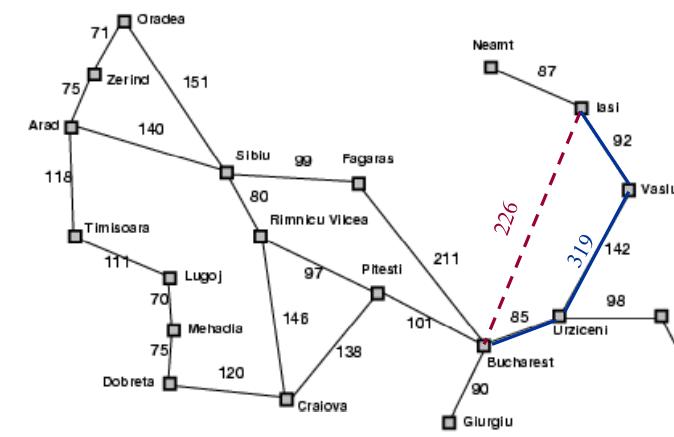
# جستجوی اول- بهترین

- ایده: برای هر گره از یک **تابع ارزیابی** (evaluation function) استفاده کن.
- تخمین "میزان مطلوب بودن" گره
- ← هر یار مطلوب ترین گره گسترش نیافته را بسط می دهد.
- پیاده سازی:
- یک صفت می باشد که بر اساس میزان مطلوب بودن گره ها مرتب می باشد.
- موارد خاص:
  - جستجوی حریصانه (Greedy search)
  - جستجوی  $A^*$

N. Razavi - AI course - 2007

5

# نقشه رومانی به همراه هزینه مراحل برمسب km



N. Razavi - AI course - 2007

6

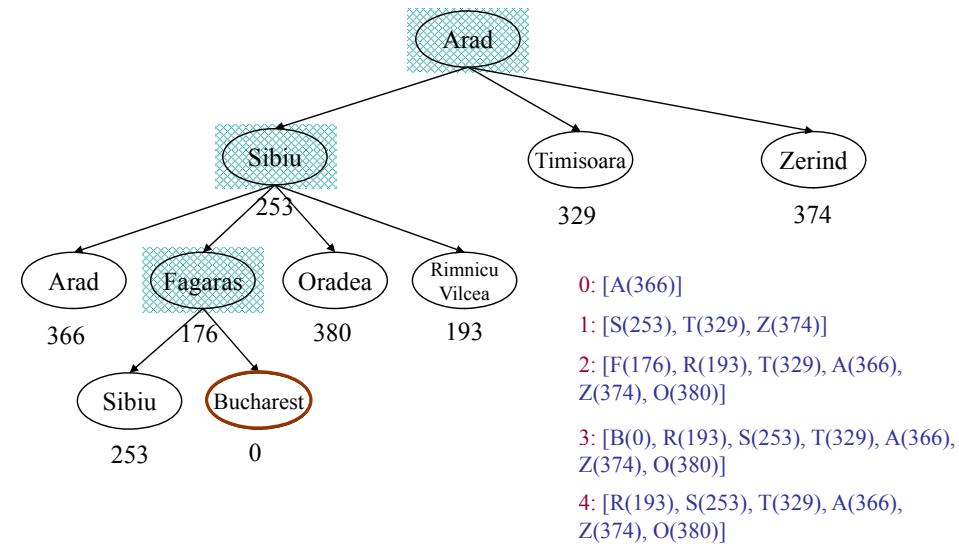
# مثال جستجوی حریصانه

- تابع ارزیابی  $f(n) = h(n)$  (هیوریستیک)
- = هزینه تخمینی از گره  $n$  تا نزدیکترین هدف
- مثال:  $h_{SLD}(n)$  = فاصله مستقیم از  $n$  تا بخارست.
- جستجوی حریصانه گره ای را گسترش می دهد که به نظر می رسد نزدیکترین گره به هدف (بخارست) باشد.

N. Razavi - AI course - 2007

7

# مثال جستجوی حریصانه



N. Razavi - AI course - 2007

8

## فواص جستجوی حریصانه

- کامل؟ خیر، ممکن است در حلقه گیر کند.  
- مثال: حالت اولیه Iasi، حالت هدف Fagaras
- Iasi → Neamt → Iasi → Neamt → ...
- پیچیدگی زمانی؟  $O(b^m)$ ، اما با یک هیوریستیک خوب می تواند بسیار بهبود یابد.
- پیچیدگی حافظه؟  $O(b^m)$ ، تمام گره ها را در حافظه نگه می دارد.
- بهینه؟ خیر، (با توجه به مثال قبل)

N. Razavi - AI course - 2007

9

## جستجوی A\*

- ایده: از گسترش مسیرهایی که تاکنون مشخص شده پژوهیه می باشد، انتخاب کن.
- ترکیب مزایای UCS و جستجوی حریصانه:  
- جستجوی حریصانه  $h(n)$  را حداقل می کند، نه کامل و نه بهینه  
- جستجوی UCS، هزینه مسیر را حداقل می کند؛ کامل و بهینه است؛ می تواند بسیار زمانبر باشد.

### تابع ارزیابی

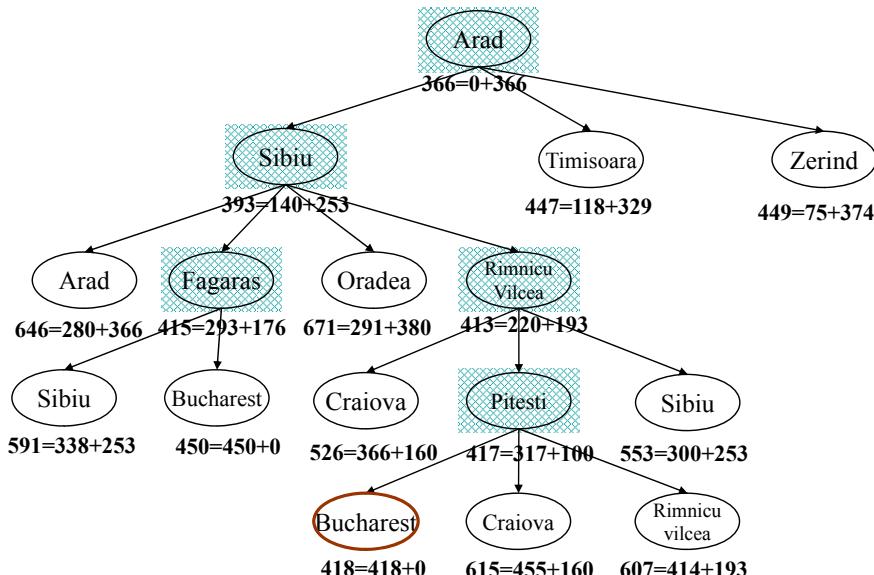
$$f(n) = g(n) + h(n)$$

- $g(n)$ : هزینه مسیر پیموده شده تا  $n$ .
- $h(n)$ : هزینه تخمینی ارزانترین مسیر راه حل از  $n$  تا هدف.
- $f(n)$ : هزینه تخمینی ارزانترین راه حل که از  $n$  می گذرد.

N. Razavi - AI course - 2007

10

## مثال جستجوی A\*



N. Razavi - AI course - 2007

11

## جستجوی A\*

از یک هیوریستیک قابل قبول (admissible) استفاده می کند،  
یعنی، همواره  $h(n) \leq h^*(n)$  که در آن  $h^*(n)$  هزینه واقعی  $n$  تا هدف  
می باشد.

هم چنین داریم  
 $h(n) \geq 0$

$h(G) = 0$  یک هدف می باشد.  
به طور کلی :

$$0 \leq h(n) \leq h^*(n)$$

مثال: در هیوریستیک  $h_{SLD}(n)$  هیچگاه هزینه تخمینی بیشتر از هزینه واقعی  
نخواهد بود. (کوتاهترین فاصله بین دو نقطه، فاصله مستقیم آن دو نقطه  
می باشد).

N. Razavi - AI course - 2007

12

## هیوریستیک قابل قبول

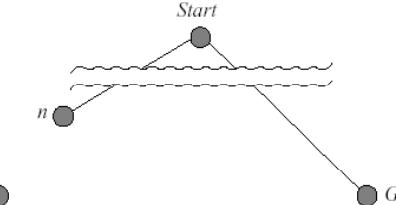
- یک هیوریستیک مانند  $h(n)$  قابل قبول است اگر برای هر گره  $n$  داشته باشیم:  $h(n) \leq h^*(n)$  که  $h^*(n)$  هزینه واقعی برای رسیدن به هدف از گره  $n$  می باشد.
- یک هیوریستیک قابل قبول هرگز هزینه رسیدن به هدف را بیش از حد تخمین نمی زند، یعنی خوش بینانه است.
- مثال: هیوریستیک  $h_{SLD}(n)$  (هیچگاه فاصله واقعی را بیش از حد تخمین نمی زند).
- قضیه: اگر  $h(n)$  قابل قبول باشد،  $A^*$  با استفاده از TREE-SEARCH بهینه است.

N. Razavi - AI course - 2007

13

## بهینگی $A^*$ (اثبات)

- فرض کنید یک جواب زیر بهینه مانند  $G_2$  تولید شده و در صفحه قرار دارد. هم چنین فرض کنید  $n$  یک گره گسترش نیافر را کوتاهترین مسیر به هدف بهینه  $G$  باشد (۹۹)



$$\begin{aligned} f(G_2) &= g(G_2) \\ &> g(G) \\ &\geq f(n) \end{aligned}$$

چون،  $f(G_2) \geq f(n)$   
چون،  $g(G_2) > g(G)$   
چون،  $g(G_2) \geq f(n)$

چون ( $f(G_2) \geq f(n)$ )، الگوریتم  $A^*$  هیچ وقت  $G_2$  را برای گسترش یافتن انتخاب نمی کند.

N. Razavi - AI course - 2007

14

## اثبات لم: سازگاری (*Consistency*)

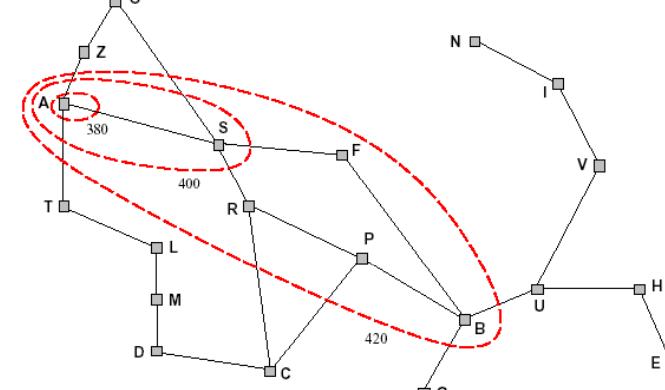
- $h(n) \leq c(n, a, n') + h(n')$  یک هیوریستیک سازگار است اگر:
  - اگر  $h$  سازگار باشد، داریم:
- $$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &\geq f(n) \end{aligned}$$
- (monotonicity) یعنی،  $f(n)$  در طول هر مسیری غیر کاهشی می باشد. (یکنواختی،
- قضیه: اگر  $h(n)$  سازگار باشد،  $A^*$  با استفاده از GRAPH-SEARCH بهینه است.

N. Razavi - AI course - 2007

15

## بهینگی $A^*$

- لم:  $A^*$  گره ها را براساس ترتیب صعودی مقادیر  $f$  گسترش می دهد.
- $A^*$  به تدریج  $f$ -contour ها را اضافه می کند.
- کانتور  $I$  شامل تمام گره ها با  $f = f_i$  می باشد، که  $f_i < f_{i+1}$  شامل تمام گره ها باشد، که  $f = f_i$  می باشد، که



16

## خصوصیات A\*

- کامل؟ بله، مگر اینکه تعدادی نامحدود گره با  $f \leq f(G)$  وجود داشته باشد.  $A^*$  در گراف های متناهی محلی (با فاکتور انشعاب محدود) کامل است به شرط آنکه هزینه تمام عملگرها مثبت باشد.
- گره ای با فاکتور انشعاب نامحدود وجود داشته باشد.
- مسیری با هزینه محدود اما با تعداد گره های نامحدود وجود داشته باشد.

- پیچیدگی زمانی؟ نمایی بر حسب [خطای نسبی  $h^*$  \* طول راه حل] مگر اینکه خطا درتابع کشف کننده رشدی سریعتر از لگاریتم هزینه مسیر واقعی نداشته باشد، به زبان ریاضی:

$$|h(n) - h^*(n)| \leq O(\log h^*(n))$$

N. Razavi - AI course - 2007

17

## جستجوی هیو(یستیک) با حافظه محدود

- چند راه حل برای مسئله حافظه در  $A^*$  (با حفظ خصوصیات کامل بودن و بهینگی):
  - جستجوی عمیق کننده تکرای  $A^*$  (IDA\*)
- مانند IDS ولی به جای محدوده عمقی از محدوده  $(g + h)$   $f\text{-cost}$  استفاده می شود
  - جستجوی اول-بهترین بازگشتی (RBFS)
- یک الگوریتم بازگشتی با فضای خطی که سعی می کند از جستجوی اول-بهترین استاندارد تقليد کند
- جستجوی (ساده شده)  $A^*$  با حافظه محدود ((S)MBA\*)
  - حذف بدترین گره وقتی که حافظه پر می باشد

N. Razavi - AI course - 2007

19

## خصوصیات A\*

- پیچیدگی فضای تمام گره ها در حافظه نگه می دارد.
- بهینه؟ بله - نمی تواند  $f_{i+1}$  را گسترش دهد مگر  $f_i$  تمام شده باشد.

- تمام گره ها با  $f^* < f(n)$  را گسترش می دهد.
- برخی از گره ها با  $f^* = f(n)$  را گسترش می دهد.
- گره ای با  $f^* > f(n)$  را هرگز گسترش نمی دهد.

- دارای کارآیی بهینه (optimally efficient) می باشد!

N. Razavi - AI course - 2007

18

## جستوى اول-بهترین بازگشتى (RBFS)

```

function RECURSIVE-BEST-FIRST-SEARCH(problem) return a solution or failure
  return RFBS(problem,MAKE-NODE(INITIAL-STATE[problem]),∞)

function RFBS(problem, node, f_limit) return a solution or failure and a new f-cost limit
  if GOAL-TEST[problem](STATE[node]) then return node
  successors  $\leftarrow$  EXPAND(node, problem)
  if successors is empty then return failure,  $\infty$ 
  for each s in successors do
    f[s]  $\leftarrow$  max(g(s) + h(s), f[node])
  repeat
    best  $\leftarrow$  the lowest f-value node in successors
    if f[best] > f_limit then return failure, f[best]
    alternative  $\leftarrow$  the second lowest f-value among successors
    result, f[best]  $\leftarrow$  RBFS(problem, best, min(f_limit, alternative))
    if result  $\neq$  failure then return result
  
```

N. Razavi - AI course - 2007

20

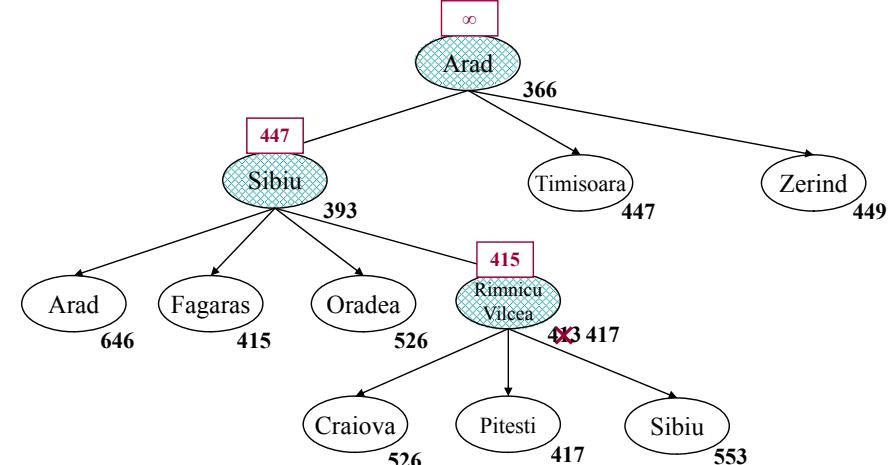
## RBFS جستجوی

- مقدار f-value مربوط به بهترین مسیر جایگزین موجود را نگهداری می کند.
- اگر f-value فعلی از f-value مسیر جایگزین تجاوز کند، آنگاه به این مسیر جایگزین بازگشت می کند.
- در بازگشت به عقب مقدار f-value مربوط به هر گره موجود در مسیر را با بهترین مقدار f-value فرزندانش جایگزین می کند.
- بنابراین گسترش مجدد نتیجه فعلی هنوز ممکن می باشد.

N. Razavi - AI course - 2007

21

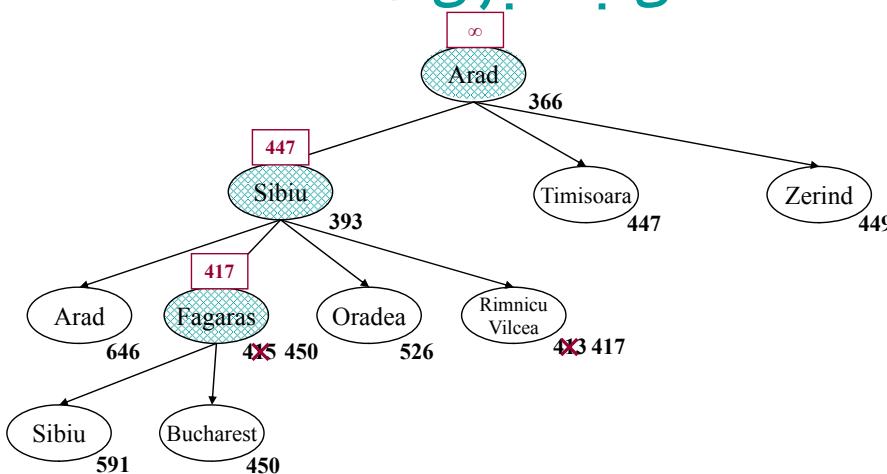
## مثال جستجوی RBFS



N. Razavi - AI course - 2007

22

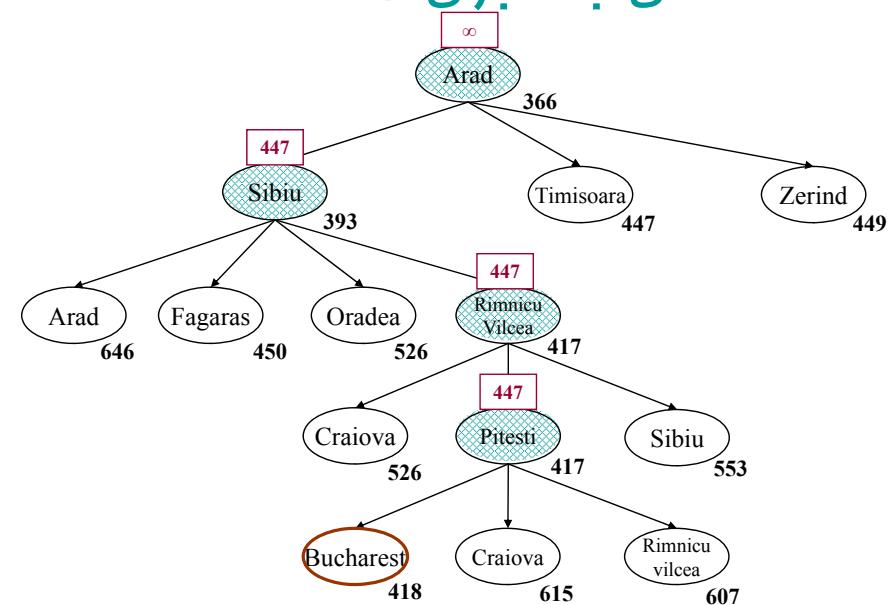
## مثال جستجوی RBFS



N. Razavi - AI course - 2007

23

## مثال جستجوی RBFS



N. Razavi - AI course - 2007

24

# ازیابی RBFS

- RBFS کمی کارآتر از IDA\* می باشد
- هنوز گسترش اضافی گره ها وجود دارد (تغییر عقیده)
- RBFS هم مانند A\*, اگر  $h(n) \leq f(n)$  قابل قبول باشد بهینه است
- پیچیدگی حافظه  $O(bd)$
- IDA\* فقط یک عدد رانگهداری می کند (حد فعلی  $f-cost$ )
- تعیین پیچیدگی زمانی مشکل است
- به دقت تابع هیوریستیک و میزان تغییر بهترین مسیر در اثر بسط گره ها بستگی دارد.
- مانند IDA\* در معرض افزایش نمایی پیچیدگی زمانی قرار دارد.
- IDA\* هر دو از **میزان بسیار کم حافظه** رنج می برند.
- با اینکه حافظه زیادی وجود دارد، نمی توانند از ان استفاده کنند.

N. Razavi - AI course - 2007

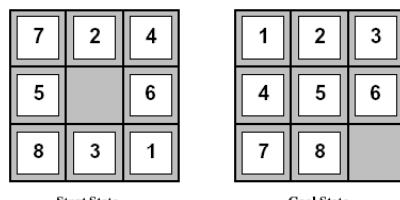
25

## هیوریستیک های قابل قبول

مثال برای معماهی هشت:

$h_1(n)$  = number of misplaced tiles

$h_2(n)$  = total Manhattan distance



$$h_1(S) = 6$$

$$h_2(S) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1 = 14$$

N. Razavi - AI course - 2007

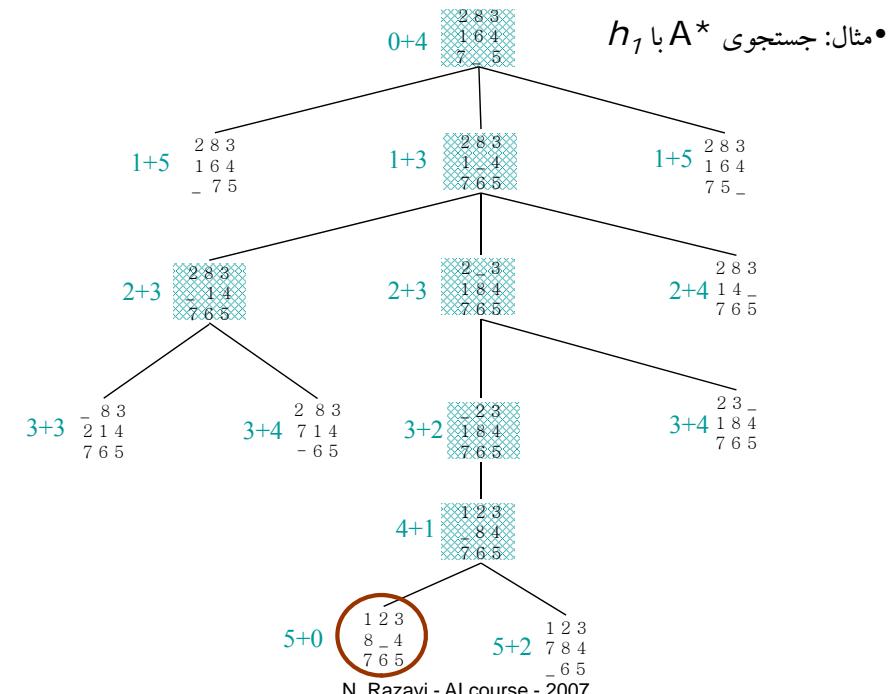
27

## (Simplified) Memory Bounded A\*

- **ایده:** استفاده از تمامی حافظه موجود
- یعنی، گسترش بهترین گره های برگی تا زمانی که حافظه موجود پر شود
- در صورت پر شدن حافظه، SMA\* بدترین گره برگ (با بیشترین مقدار *f-value*) را از حافظه حذف می کند
- مانند RBFS اطلاعات گره های فراموش شده را در پدرشان ذخیره می کند
- اگر تمام برگ ها دارای *f-value* برابر باشند چه می شود؟
- ممکن است یک گره را هم برای گسترش و هم برای حذف انتخاب کند
- راه حل :SMA\*
- گسترش: بهترین گره برگ که از همه جدید تر است
- حذف: بدترین گره که از همه قدیمی تر است
- SMA\* کامل است اگر راه حل قابل دستیابی وجود داشته باشد و بهینه است اگر راه حل بهینه قابل دستیابی باشد

N. Razavi - AI course - 2007

26



N. Razavi - AI course - 2007

28

## تسلط (Dominance)

- اگر بازاء هر  $n$  داشته باشیم،  $h_2(n) \geq h_1(n)$  و هر دو هیوریستیک قابل قبول باشند)

آنگاه  $h_2$  بر  $h_1$  تسلط دارد و برای جستجو بهتر می باشد.

- مثال هزینه جستجو: تعداد گره های تولید شده در عمق های مختلف:

$$d = 14 \rightarrow \text{IDS} = 3,473,941$$

$$A^*(h_1) = 539$$

$$A^*(h_2) = 113$$

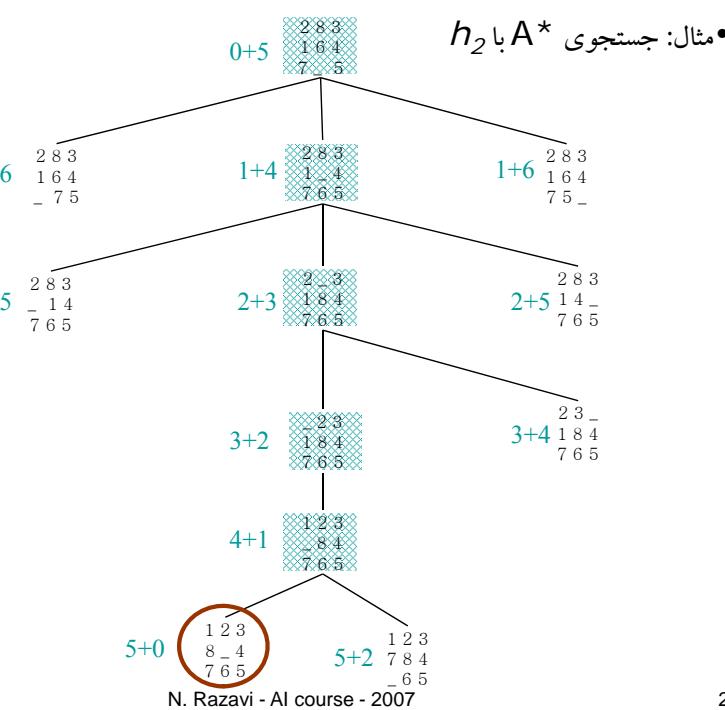
$$d = 24 \rightarrow \text{IDS} = 54,000,000,000$$

$$A^*(h_1) = 39,135$$

$$A^*(h_2) = 1641$$

N. Razavi - AI course - 2007

30



N. Razavi - AI course - 2007

29

## فاکتور انشعاب موثر ( $b^*$ )

### فاکتور انشعاب موثر (EBF):

- اگر تعداد گره های گسترش یافته توسط روال جستجو  $N$  باشد و عمق راه حل  $d$  باشد،  $b^*$  به صورت زیر محاسبه می شود:

$$N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$$

- مثال:** اگر  $A^*$  راه حلی را در عمق 5 با استفاده از 52 گره پیدا کند، فاکتور انشعاب موثر را محاسبه کنید.
- پاسخ:

$$53 = 1 + b^* + (b^*)^2 + \dots + (b^*)^5 \Rightarrow b^* = 1.92$$

- در یک هیوریستیک هر چه فاکتور انشعاب به یک نزدیکتر باشد، بهتر است و آن هیوریستیک **کیفیت کشف کنندگی** بیشتری دارد.

N. Razavi - AI course - 2007

31

## مقایسه بین هزینه جستجو و فاکتور انشعاب موثر

$d$	Search Cost			Effective Branching Factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6,384	39	25	2.80	1.33	1.24
10	47,127	93	39	2.79	1.38	1.22
12	364,404	227	73	2.78	1.42	1.24
14	3,473,941	539	113	2.83	1.44	1.23
16	-	1,301	211	-	1.45	1.25
18	-	3,056	363	-	1.46	1.26
20	-	7,276	679	-	1.47	1.27
22	-	18,094	1,219	-	1.48	1.28
24	-	39,135	1,641	-	1.48	1.26

N. Razavi - AI course - 2007

32

## ابداع توابع هدویستیک

- می توان هیوریستیکهای قابل قبول را برای یک مسئله، از **هزینه دقیق** راه حل یک نسخه **راحت (relaxed)** از مسئله بدست آورد.
- مثال: قانون معماهی هشت. یک کاشی می تواند از خانه A به خانه B برود، اگر A مجاور B باشد و B خالی باشد.
- اگر قوانین معماهی هشت به گونه ای راحت شوند که یک کاشی بتواند **به هر خانه ای** حرکت کند، هیوریستیک  $h_1(n)$  کوتاهترین راه حل را می دهد.
- اگر قوانین به گونه ای راحت شوند که یک کاشی بتواند **به هر خانه مجاور** حرکت کند، هیوریستیک  $h_2(n)$  کوتاهترین راه حل را می دهد.
- **نکته کلیدی:** هزینه راه حل بهینه یک مسئله راحت، بیشتر از هزینه راه حل بهینه در مسئله واقعی نیست.

N. Razavi - AI course - 2007

33

## الگوریتم های جستجوی محلی و مسائل بهینه سازی

- در بسیاری از مسائل بهینه سازی، **مسیر** راه حل اهمیت ندارد؛ خود حالت هدف پاسخ مسئله می باشد.
- فضای حالت = مجموعه پیکره بندی های «کامل»؛
- یافتن پیکره بندی بهینه، مانند TSP
- یافتن یک پیکره بندی که محدودیت های مسئله را ارضاء کند، مانند مسئله  $n$ -وزیر
- در چنین مواردی می توان از **الگوریتم های جستجوی محلی** بهره گرفت.
- یک حالت « فعلی » را به تنها یک در نظر بگیر؛ سعی کن آن را بهبود بخشی.

N. Razavi - AI course - 2007

34

## الگوریتم های جستجوی محلی و مسائل بهینه سازی

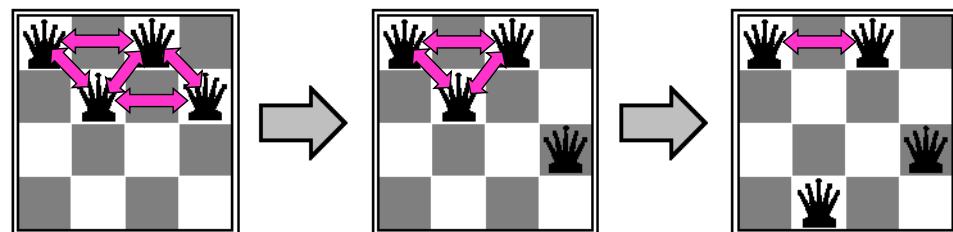
- جستجوی محلی = استفاده از یک حالت فعلی و حرکت به حالت های همسایه
- مزایا:
  - استفاده از حافظه بسیار کم
  - یافتن راه حل های معقول در اغلب موارد در فضاهای حالت بزرگ و یا نامحدود
- مفید برای مسائل بهینه سازی محض (objective function)
  - یافتن بهترین حالت بر طبق تابع هدف

N. Razavi - AI course - 2007

35

## مثال: $n$ -وزیر

- مسئله  $n$ -وزیر را در یک صفحه شطرنج  $n \times n$  به گونه ای قرار بده که هیچ دو وزیری در یک سطر، ستون و یا قطر قرار نگیرند.
- یکی از وزیرها را در ستون خودش به گونه ای جایه جا کن که تعداد برخورد ها کاهش یابد.



N. Razavi - AI course - 2007

36

# تپه نوادی (یا گرادیان صعودی/نزولی)

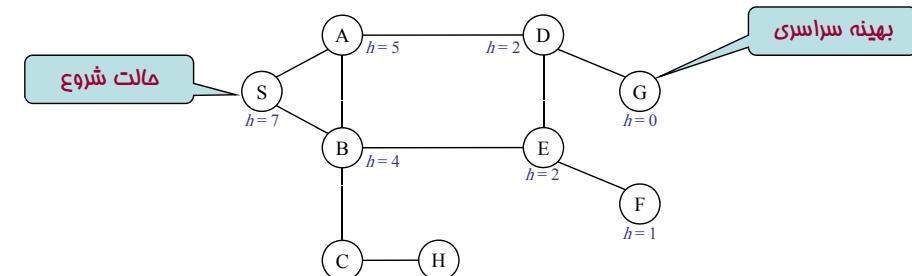
«مانند بالا رفتن از کوه اورست در مه غلیظ با ضعف حافظه»

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                  neighbor, a node
  current  $\leftarrow$  MAKE-NODE(INITIAL-STATE[problem])
  loop do
    neighbor  $\leftarrow$  a highest-valued successor of current
    if VALUE[neighbor]  $\leq$  VALUE[current] then return STATE[current]
    current  $\leftarrow$  neighbor
```

N. Razavi - AI course - 2007

37

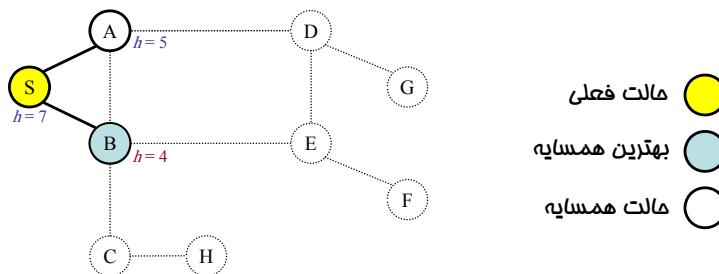
# مثال: جستجوی تپه نوادی



N. Razavi - AI course - 2007

38

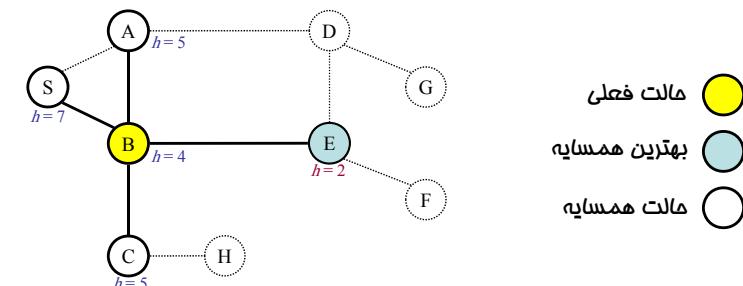
# مثال: جستجوی تپه نوادی



N. Razavi - AI course - 2007

39

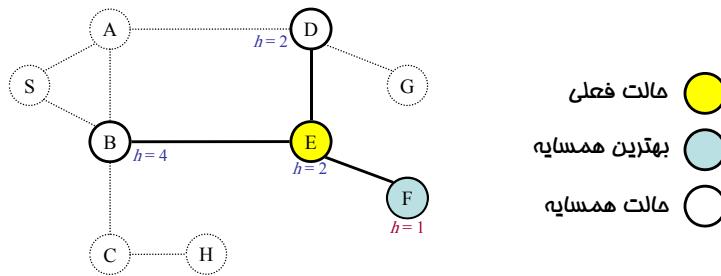
# مثال: جستجوی تپه نوادی



N. Razavi - AI course - 2007

40

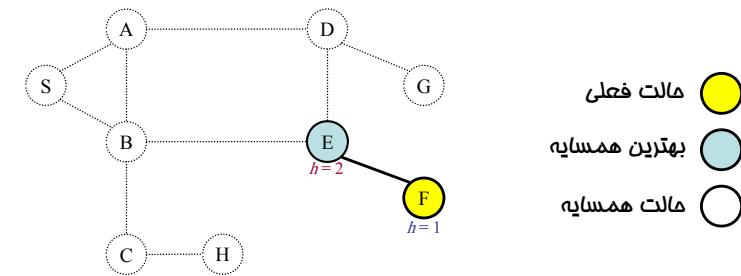
## مثال: جستجوی تپه نوادی



N. Razavi - AI course - 2007

41

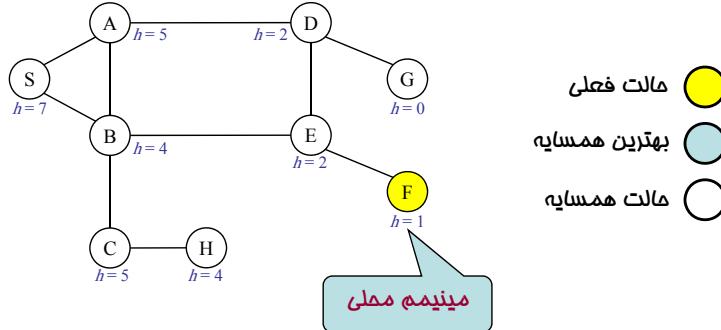
## مثال: جستجوی تپه نوادی



N. Razavi - AI course - 2007

42

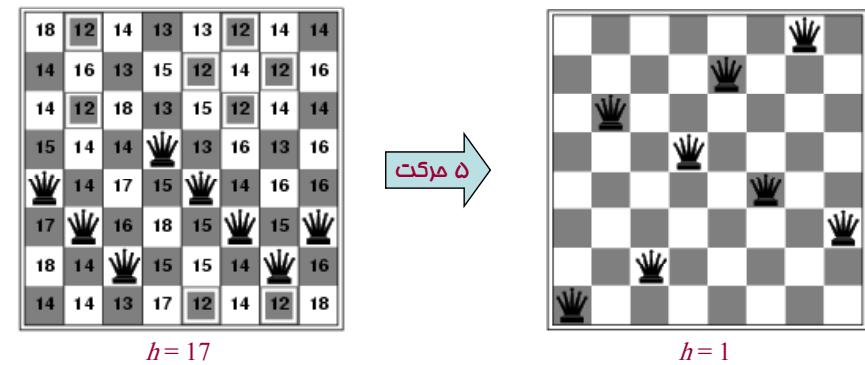
## مثال: جستجوی تپه نوادی : مثال ۸ وزیر



N. Razavi - AI course - 2007

43

## جستجوی تپه نوادی : مثال ۸ وزیر



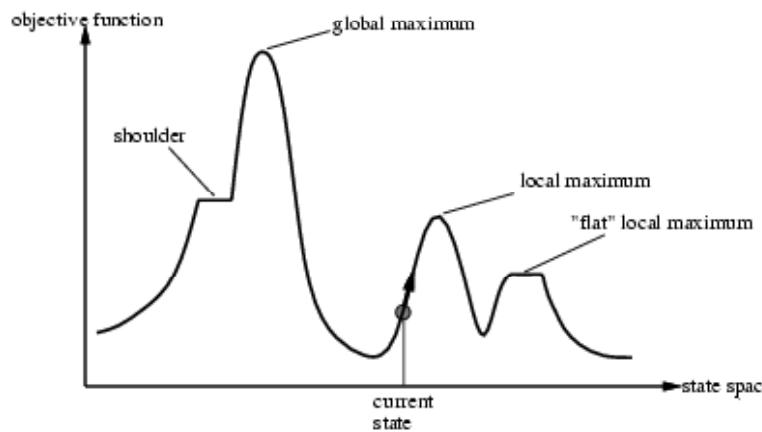
$h$  = تعداد جفت وزیرهایی که بطور مستقیم و یا بطور غیر مستقیم یکدیگر را تهدید می کنند.

N. Razavi - AI course - 2007

44

## تپه نوردهی (ادامه)

- مشکل: بسته به حالت اولیه مسئله ممکن است در **ماکریم محلی** گیر کند.



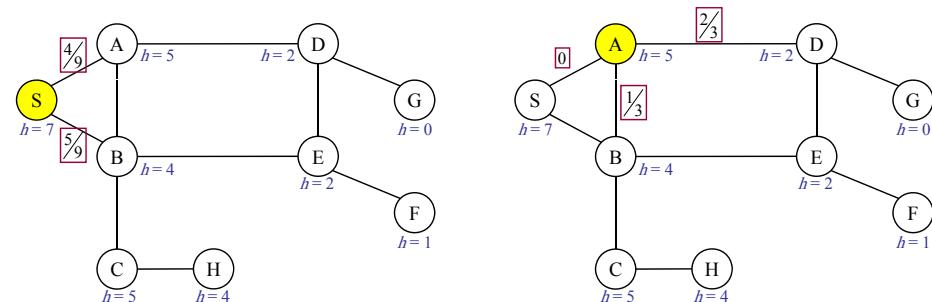
N. Razavi - AI course - 2007

45

## انواع دیگر تپه نوردهی (تپه نوردهی اتفاقی)

- تپه نوردهی اتفاقی

- انتخاب تصادفی در میان حرکت های رو به بالا
- احتمال انتخاب می تواند مناسب با شب حرکت تغییر کند



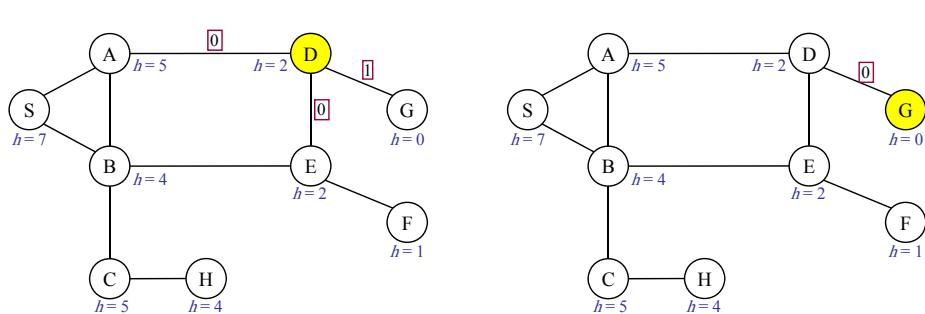
N. Razavi - AI course - 2007

46

## انواع دیگر تپه نوردهی (تپه نوردهی اولین انتخاب)

### تپه نوردهی اتفاقی

- انتخاب تصادفی در میان حرکت های رو به بالا
- احتمال انتخاب می تواند مناسب با شب حرکت تغییر کند



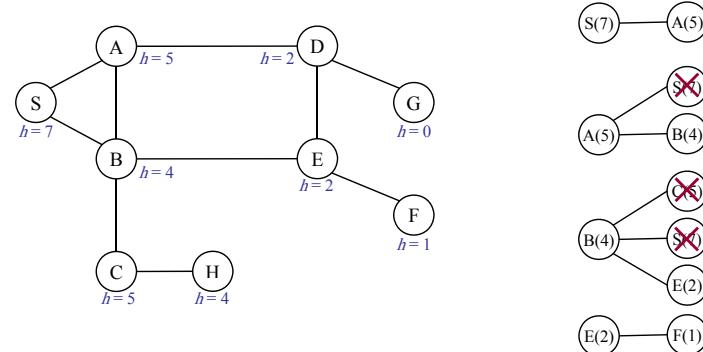
N. Razavi - AI course - 2007

47

## انواع دیگر تپه نوردهی (تپه نوردهی اولین انتخاب)

### تپه نوردهی اولین انتخاب

- همان تپه نوردهی اتفاقی که حالت های بعدی را به طور تصادفی تولید می کند تا یکی از آنها بهتر از حالت فعلی باشد

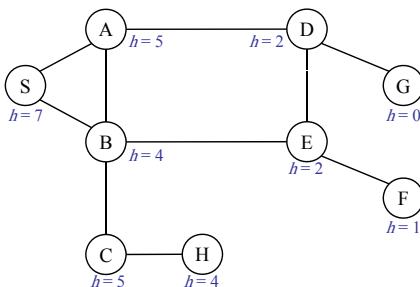


N. Razavi - AI course - 2007

48

## انواع دیگر تپه نوردی (تپه نوردی با شروع مجدد تصادفی)

- تپه نوردی با شروع مجدد تصادفی (ایده: اگر شکست خوردن دوباره سعی کن)
  - سعی می کند که از گیرافتادن در ماکریم محلی اجتناب کند



$S \rightarrow B \rightarrow E \rightarrow F \times$   
 $C \rightarrow B \rightarrow E \rightarrow F \times$   
 $H \times$   
 $C \rightarrow H \times$   
 $A \rightarrow D \rightarrow G$

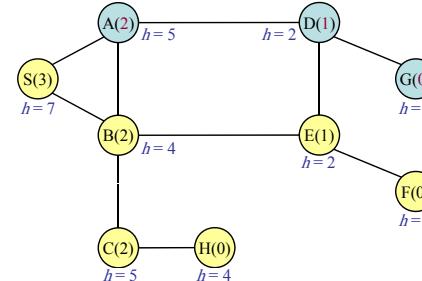
مشکل: معمولاً در یک مسئله در دنیای واقعی، قبل از حل مسئله جواب بهینه را نمی دانیم!  
 راه حل: تکرار تپه نوردی تا زمانی که وقت داریم.

N. Razavi - AI course - 2007

49

## انواع دیگر تپه نوردی (تپه نوردی با شروع مجدد تصادفی)

- محاسبه تعداد متوسط مراحل در تپه نوردی با شروع تصادفی مجدد
  - تعداد تکرارها = عکس احتمال موفقیت
  - تعداد متوسط مراحل = تعداد شکست ها \* تعداد متوسط مراحل شکست + ۱ \* تعداد متوسط مراحل موفقیت



$$\begin{aligned} \text{احتمال موفقیت} &= \frac{1}{3} \\ \text{تعداد تکرارها} &= 3 \\ \text{تعداد متوسط مراحل موفقیت} &= 1 \\ \text{تعداد متوسط مراحل شکست} &= \frac{8}{6} \\ (3 - 1) \times \frac{8}{6} + 1 \times 1 &\approx 4 \end{aligned}$$

N. Razavi - AI course - 2007

50

## Simulated Annealing

- ایده: از ماکریم های محلی با انجام حرکت های «بد» فرار کن  
 اما به تدریج اندازه و تعداد حرکات بد (به سمت پایین) را کم کن

```
function SIMULATED_ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"
  local variables: current, a node
                    next, a node
                    T, a "temperature" controlling prob. of downward steps
  current  $\leftarrow$  MAKE-NODE(INITIAL-STATE[problem])
  for t  $\leftarrow$  1 to  $\infty$  do
    T  $\leftarrow$  schedule[t]
    if T = 0 then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow$  VALUE[next] - VALUE[current]
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 
```

N. Razavi - AI course - 2007

51

## آنالینگ شبیه سازی شده

- پیش روی مانند تپه نوردی می باشد، اما در هر مرحله حالت بعدی به طور تصادفی انتخاب می شود.
- اگر حالت بعدی انتخاب شده بهتر باشد، همواره به آن حالت بعدی خواهیم رفت.
- در غیر این صورت، تنها با یک احتمال به آن حالت خواهیم رفت و این احتمال به صورت نمایی کاهش می یابد.

$$e^{-\Delta E/T}$$

تابع احتمال:  
 $T$ : تعداد مراحل بهبود راه حل  
 $\Delta E$ : میزان کاهش در هر مرحله

N. Razavi - AI course - 2007

52

## خصوصیات آنلینگ شبیه سازی شده

- در مقادیر بالاتر  $T$ ، احتمال انجام حرکات بد (رفتن به سمت پایین) بیشتر است. (مانند جستجوی تصادفی رفتار می کند)
- با کاهش  $T$  این احتمال کاهش یافته و در  $T=0$  این احتمال به صفر می رسد. (مانند تپه نورده رفتار می کند)
- می توان ثابت کرد که اگر  $T$  به اندازه کافی آرام کاهش بیابد، جستجوی SA یک پاسخ بهینه (global optimum) با احتمالی که به سمت یک میل می کند خواهد یافت.
- کاربردها:
  - حل مسائل VLSI
  - برنامه ریزی
  - اعمال بهینه سازی
  - زمانبندی خطوط هواپی

N. Razavi - AI course - 2007

53

## Local beam search جستجوی محلی دسته ای

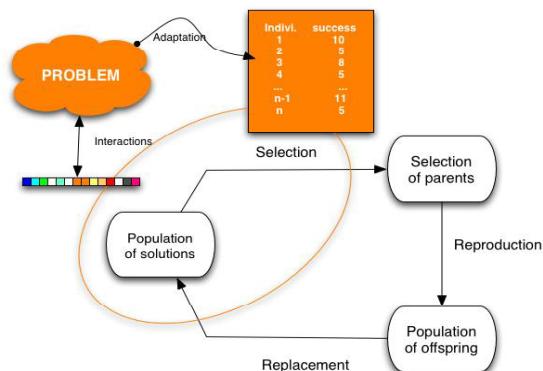
- شروع با  $K$  حالت که به طور تصادفی ایجاد شده اند.
- در هر تکرار، تمام فرزندان برای هر  $K$  حالت تولید می شوند.
- اگر یکی از آنها حالت هدف بود جستجو متوقف می شود و در غیر این صورت از میان لیست کامل فرزندان  $K$  تا از بهترین ها انتخاب می شوند و مرحله بالا تکرار می شود.

N. Razavi - AI course - 2007

54

## الگوریتم های ژنتیک

- نوعی از local beam search به همراه ترکیب جنسی



N. Razavi - AI course - 2007

55

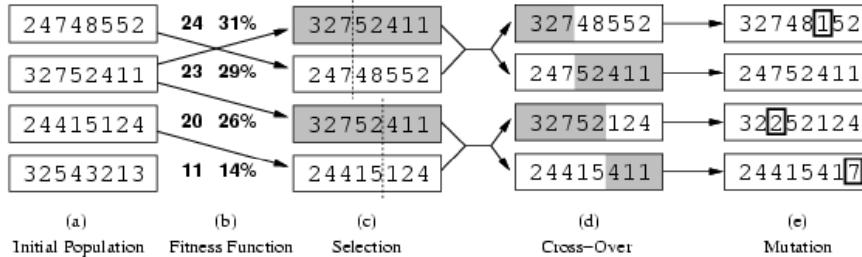
## الگوریتم های ژنتیک

- یک حالت بعدی با ترکیب دو حالت پدر ایجاد می شود.
- شروع با  $K$  حالت تصادفی (جمعیت)
- یک حالت با یک رشته بر روی یک مجموعه محدود بازنمایی می شود (غلب رشته ای از صفر و یک) – (کروموزوم)
- تابع ارزیابی (تابع برازنده‌گی – Fitness function) : مقادیر بالاتری را برای حالات بهتر ایجاد می کند.
- نسل بعدی جمعیت با انجام اعمال زیر روی جمعیت فعلی تولید می شود:
  - انتخاب (Selection)
  - آمیزش (Crossover)
  - جهش (Mutation)

N. Razavi - AI course - 2007

56

## الگوریتم های ژنتیک



- تابع برازنده‌گی: تعداد جفت وزیر هایی که یکدیگر را تهدید نمی‌کنند  
(مینیمم: صفر، ماکزیمم:  $8 * 7/2 = 28$ )

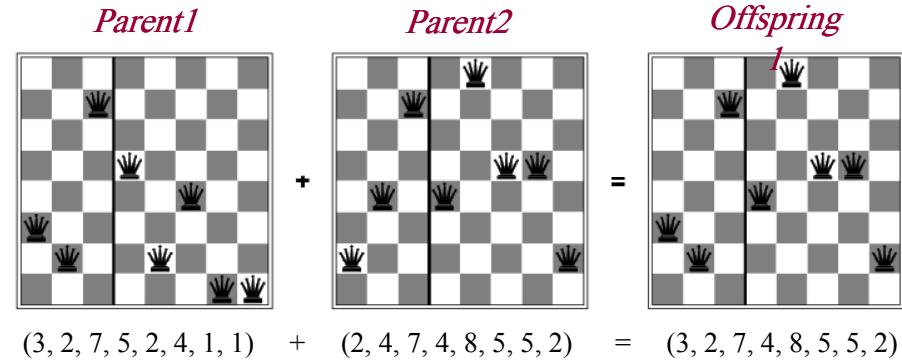
- $24 / (24+23+20+11) = 31\%$
- $23 / (24+23+20+11) = 29\%$

N. Razavi - AI course - 2007

57

## الگوریتم های ژنتیک

- یک مثال برای عمل Crossover



N. Razavi - AI course - 2007

58

## الگوریتم ژنتیک

```

function GENETIC-ALGORITHM( population, FITNESS-FN ) returns an individual
  input: population, a set of individuals
        FITNESS-FN, a function that measures the fitness of an individual

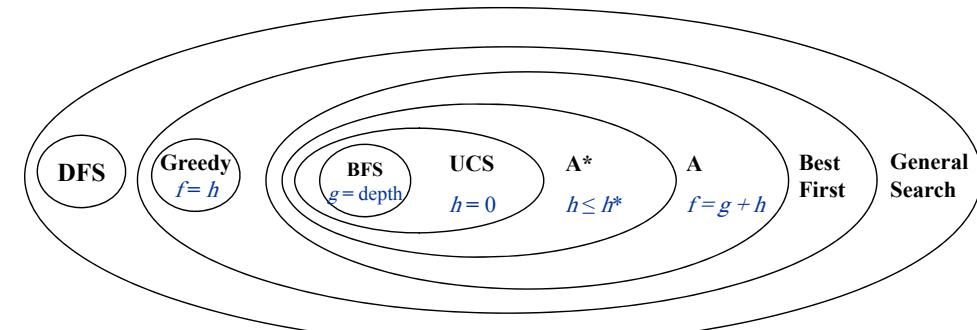
  repeat
    new_population ← empty set
    loop for i from 1 to SIZE(population) do
      x ← RANDOM SELECTION(population, FITNESS_FN)
      y ← RANDOM_SELECTION(population, FITNESS_FN)
      child ← REPRODUCE(x,y)
      if (small random probability) then child ← MUTATE(child)
      add child to new_population
    population ← new_population
  until some individual is fit enough or enough time has elapsed
  return the best individual in population, according to FITNESS-FN

```

N. Razavi - AI course - 2007

59

## دسته بندی استراتژی های جستجو



N. Razavi - AI course - 2007

60

## مسائل اکتشافی

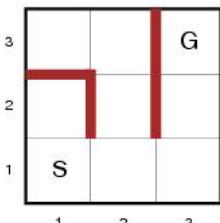
- تا کنون تمام الگوریتم ها offline بودند
  - Offline = راه حل قبل از اجرای آن معین است
  - Online = محاسبه و عمل بصورت یک در میان
  - جستجوی online برای محیط های پویا و نیمه پویا ضروری می باشد
  - در نظر گرفتن تمامی امکان های مختلف غیر ممکن است
  - استفاده شده در مسائل اکتشافی
    - حالت ها و اعمال ناشناخته
    - مثال: یک روبات در یک محیط جدید، یک نوزاد و ...

N. Razavi - AI course - 2007

61

## مسائل جستجوی online

- هدف: رسیدن به حالت هدف با حداقل هزینه
  - هزینه = کل هزینه مسیر پیموده شده
  - نسبت رقابتی = مقایسه هزینه با هزینه مسیر راه حل در حالتی که فضای جستجو شناخته شده باشد
  - می تواند نامحدود باشد
  - در مواردی که عامل به طور تصادفی
    - به یک بن بست می رسد



N. Razavi - AI course - 2007

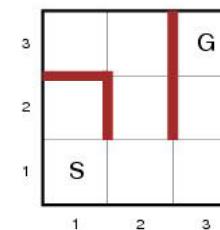
63

## online جستجوی

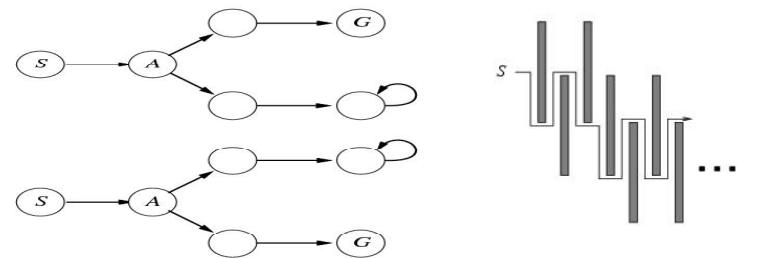
- دانش عامل:
  - عمل (ها): لیست اعمال مجاز در حالت  $S$
  - $C(S, a, S')$ : تابع هزینه گام (پس از تعیین  $S'$ )
  - GOAL-TEST( $s$ )
  - عامل می تواند حالت قبلی را تشخیص دهد
  - اعمال قطعی هستند
  - دستیابی به هیوریستیک قابل قبول ( $h(s)$ )
  - مانند فاصله مانهاتنی

N. Razavi - AI course - 2007

62



## استدلال دشمنانه



- یک دشمن را تصور کنید که قادر است در حین کاوش عامل فضای حالت را بازد
- حالت های ملاقات شده  $S$  و  $A$ . حالت بعدی؟
  - در یکی از فضاهای حالت شکست می خورد
  - هیچ الگوریتمی نمی تواند از بن بست ها در تمامی فضاهای حالت اجتناب کند

N. Razavi - AI course - 2007

64

## Online DFS-Agent

```

function ONLINE-DFS-AGENT( $s'$ ) returns an action
input:  $s'$ , a percept identifying current state
static:  $result$ , a table indexed by action and state, initially empty
         $unexplored$ , a table that lists for each visited state, the action not yet tried
         $unbacktracked$ , a table that lists for each visited state, the backtrack not yet tried
         $s, a$ , the previous state and action, initially null

if GOAL-TEST( $s'$ ) then return stop
if  $s'$  is a new state then  $unexplored[s'] \leftarrow ACTIONS(s')$ 
if  $s$  is not null then do
     $result[a, s] \leftarrow s'$ 
    add  $s$  to the front of  $unbacktracked[s']$ 
if  $unexplored[s']$  is empty then
    if  $unbacktracked[s']$  is empty then return stop
    else  $a \leftarrow$  an action  $b$  such that  $result[b, s'] = POP(unbacktracked[s'])$ 
else  $a \leftarrow POP(unexplored[s'])$ 
 $s \leftarrow s'$ 
return  $a$ 

```

## عامل های جستجوی online

- عامل یک نقشه از محیط نگهداری می کند
  - نقشه بر اساس ورودی ادراکی بهنگام می شود
  - از این نقشه برای انتخاب عمل بعدی استفاده می شود
- به تفاوت مثلا با  $A^*$  دقت کنید
  - یک نسخه online تنها می تواند گرهی را گسترش دهد که به طور فیزیکی در آن قرار داشته باشد

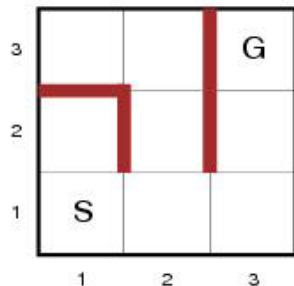
N. Razavi - AI course - 2007

65

N. Razavi - AI course - 2007

66

## جستجوی عمقی online، مثال

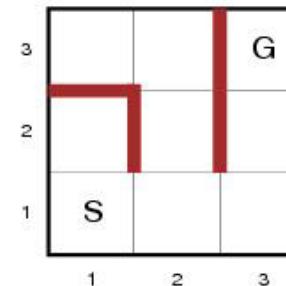


- مساله مسیر پر پیچ و خم بر روی یک صفحه  $3 \times 3$
- حالت اولیه:  $s' = (1, 1)$
- Unexplored(UX) و Result
- Unbacktracked(UB) و ...
- تهی هستند
- $S$  و  $a$  نیز تهی هستند

N. Razavi - AI course - 2007

67

## جستجوی عمقی online، مثال

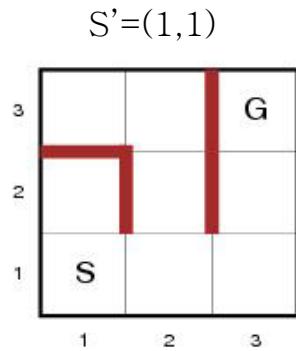


- GOAL-TEST((1, 1))?
  - $S$  not =  $G$  thus false
- (1,1) a new state?
  - True
  - ACTION((1,1)) -> UX[(1,1)]
    - {RIGHT,UP}
- $s$  is null?
  - True (initially)
- UX[(1,1)] empty?
  - False
- POP(UX[(1,1)]) ->  $a$ 
  - $a = UP$
- $s = (1,1)$
- Return  $a$

N. Razavi - AI course - 2007

68

## جستجوی عمیق online، مثال

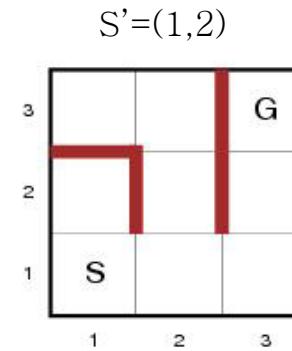


- GOAL-TEST((1,1))?  
–  $S \neq G$  thus false
- (1,1) a new state?  
– false
- $s$  is null?  
– false ( $s = (2, 1)$ )  
– result [DOWN, (2, 1)]  $\leftarrow (1, 1)$   
–  $UB[(1,1)] = \{(2,1)\}$
- $UX[(1,1)]$  empty?  
– False
- $a = RIGHT, s = (1,1)$
- return  $a$

N. Razavi - AI course - 2007

69

## جستجوی عمیق online، مثال

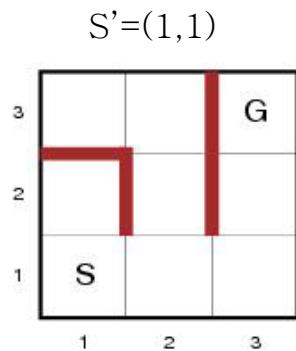


- GOAL-TEST((1,2))?  
–  $S \neq G$  thus false
- (1,2) a new state?  
– True,  
 $UX[(1,2)] = \{RIGHT, UP, LEFT\}$
- $s$  is null?  
– false ( $s = (1,1)$ )  
– result [RIGHT, (1,1)]  $\leftarrow (1, 2)$   
–  $UB[(1,2)] = \{(1,1)\}$
- $UX[(1, 2)]$  empty?  
– False
- $a = LEFT, s = (1, 2)$
- return  $a$

N. Razavi - AI course - 2007

70

## جستجوی عمیق online، مثال



- GOAL-TEST((1, 1))?  
–  $S \neq G$  thus false
- (1, 1) a new state?  
– false
- $s$  is null?  
– false ( $s = (1, 2)$ )  
– result [LEFT, (1, 2)]  $\leftarrow (1, 1)$   
–  $UB[(1, 1)] = \{(1, 2), (2, 1)\}$
- $UX[(1, 1)]$  empty?  
– True  
–  $UB[(1, 1)]$  empty? False
- $a = b$  for  $b$  in result [ $b, (1,1)$ ] = (1, 2)  
–  $b = RIGHT$
- $a = RIGHT, s = (1, 1) \dots$

N. Razavi - AI course - 2007

71

## جستجوی عمیق online، مثال

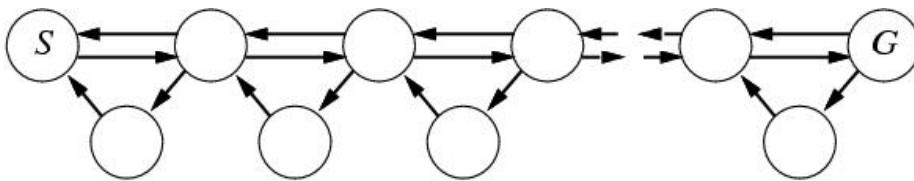
- بدترین حالت: هر گره دو بار ملاقات شده است
- یک عامل ممکن است در حالی که به پاسخ نزدیک است، یک راه طولانی را بپیماید
- یک روش عمیق کننده تکرای online می تواند این مشکل را حل کند
- جستجوی عمیق online فقط وقتی کار می کند که اعمال قابل برگشت باشند

N. Razavi - AI course - 2007

72

# جستجوی محلی online

- تپه نورده online می باشد
  - یک حالت ذخیره شده است
- عملکرد بد به دلیل وجود ماکریزم های محلی
  - شروع مجدد تصادفی غیر ممکن است
- راه حل: حرکت تصادفی باعث کاوش می شود ( می تواند حالات بسیاری را به صورت نمایی تولید کند)

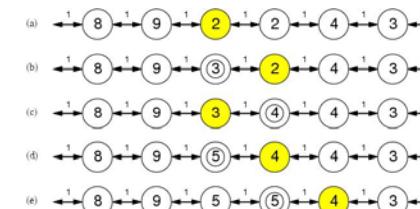


N. Razavi - AI course - 2007

73

# جستجوی محلی online

- راه حل ۲: اضافه نمودن حافظه به تپه نورده
- ذخیره بهترین تخمین فعلی ( $H(S)$ ) از هزینه رسیدن به هدف
- ( $H(S)$ ) در ابتدا تخمین هیوریستیک ( $h(S)$ ) می باشد
- پس از آن بر اساس تجربه بهنگام می شود (شکل پایین)



N. Razavi - AI course - 2007

74

## Learning real-time A\*

```

function LRTA*-COST ( $s, a, s', H$ ) return an cost estimate
  if  $s'$  is undefined return  $h(s)$ 
  else return  $c(s, a, s') + H[s]$ 

function LRTA*-AGENT ( $s$ ) return an action
  input:  $s$ , a percept identifying current state
  static:  $result$ , a table indexed by action and state, initially empty
   $H$ , a table of cost estimates indexed by state, initially empty
   $s, a$ , the previous state and action, initially null

  if GOAL-TEST ( $s$ ) then return stop
  if  $s'$  is a new state (not in  $H$ ) then  $H[s'] \leftarrow h(s')$ 
  unless  $s$  is null
     $result[a, s] \leftarrow s'$ 
     $H[s] \leftarrow \text{MIN LRTA*-COST } (s, b, result[b, s], H)$ 
     $b \in \text{ACTIONS } (s)$ 
     $a \leftarrow \text{an action } b \text{ in ACTIONS } (s) \text{ that minimizes LRTA*-COST } (s', b, result[b, s], H)$ 
     $s \leftarrow s'$ 
  return  $a$ 

```

N. Razavi - AI course - 2007

75

## مقدمه

# مسائل ارضاء محدودیت (CSP)

فصل پنجم  
سید ناصر رضوی

Email: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۴

- چند مثال از مسائل ارضاء محدودیت
- کاربرد General Search در حل CSP
- جستجوی عقبگرد (backtracking) برای CSP
- کنترل روبه جلو (forward checking) برای CSP ها
- استفاده از هیوریستیک ها در CSP ها
- جستجوی محلی برای مسائل ارضاء محدودیت

N. Razavi - AI course - 2005

2

# مسائل ارضاء محدودیت

- مسائل جستجوی استاندارد
- **حال** یک "جهبه سیاه" است - هر ساختار داده ای که از تابع حالات بعدی، تابع هیوریستیک و تست هدف پشتیبانی کند.
- :CSP
  - **حال** توسط متغیرهای  $X_i$  تعریف می شود که هر یک مقادیرشان را از یک دامنه  $D_i$  اختیار می کنند.
  - **تست هدف** مجموعه ای از محدودیت ها می باشد که ترکیبات مجاز مقادیر برای زیرمجموعه ای از متغیرها را مشخص می کند.
- مثال ساده ای از یک **زبان بازنمایی رسمی**
- امکان استفاده از الگوریتم های **همه منظوره** که نسبت به الگوریتم های استاندارد قدرت بیشتری دارند.

N. Razavi - AI course - 2005

3

## مثال: رنگ آمیزی نقشه

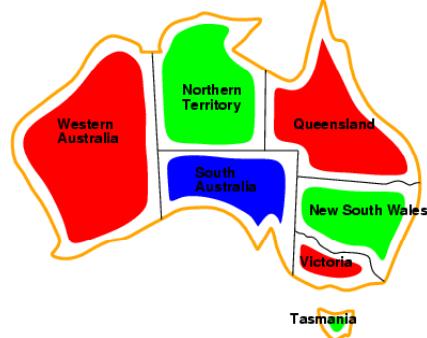


- متغیرها: WA, NT, Q, NSW, V, SA, T
- دامنه ها: {red, green, blue}
- محدودیت ها: نواحی همسایه باید رنگ متفاوتی داشته باشند
- مثلا: WA ≠ NT ، به بیان دیگر:  $\{(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)\}$

N. Razavi - AI course - 2005

4

## مثال: رنگ آمیزی نقشه



- راه حل ها انتساب های کامل و سازگار می باشند.
- مثال:

{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green}

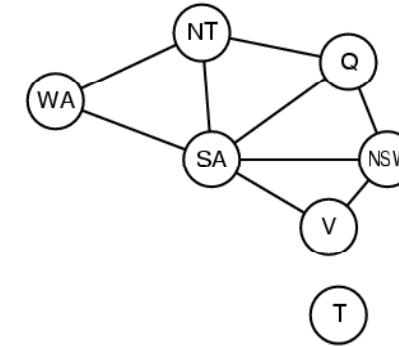
N. Razavi - AI course - 2005

5

## گراف محدودیت

### • گراف محدودیت:

- گره ها متغیرها را نشان می دهند.
- یالهای گراف محدودیت های بین متغیرها را نشان می دهند.



N. Razavi - AI course - 2005

6

## مزایای بیان مسئله به صورت CSP

- به دلیل نمایش استاندارد حالت ها (مجموعه متغیرهایی با مقدارشان)، می توان تابع Successor و آزمون هدف را به شکل کلی نوشت به طوریکه برای هر CSP قابل اعمال باشد.
- می توان هیوریستیک های کلی و کارایی ایجاد کرد که نیاز به تخصص اضافی در دامنه خاص مسئله نداشته باشند.

N. Razavi - AI course - 2005

7

## انواع مسائل CSP

### • متغیرهای گستته

#### - دامنه های محدود:

- n متغیر، اندازه هر دامنه d  $\leftarrow$  تعداد انتساب های کامل ( $O(d^n)$ )
- مثال: CSP های بولین، n- وزیر، رنگ آمیزی نقشه و ...

#### - دامنه های نامحدود:

- اعداد صحیح، رشته ها و ...
- مثال: زمانبندی کارها- متغیرها، زمان شروع/پایان هر کار هستند.
- نیاز به زبان محدودیت دارند. مثال:  $StartJob_1 + 5 \leq StartJob_3$

### • متغیرهای پیوسته

- مثال: زمان های شروع و پایان مشاهدات تلسکوپ فضایی هابل
- محدودیت های خطی که توسط برنامه ریزی خطی قابل حل در زمان چندجمله ای می باشند.

N. Razavi - AI course - 2005

8

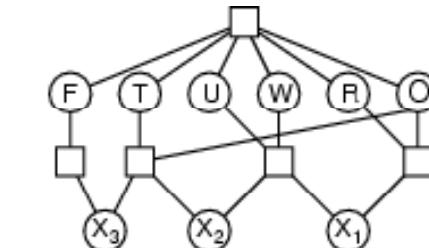
## انواع محدودیت ها

- یکانی (Unary):** روی یک متغیر تعریف می شود،  
مثال:  $SA \neq \text{green}$
- دودویی (Binary):** محدودیت شامل یک زوج از متغیرها می باشد،  
مثال:  $SA \neq WA$
- مرتبه بالاتر (Higher-order):** محدودیت شامل سه یا بیشتر متغیر است،  
مثال: محدودیت های موجود در ستون های مسائل ریاضیات رمزي  
در صورت متناهی بودن دامنه، می تواند به تعدادی محدودیت دودویی کاهش یابد.
- محدودیت های بالا همگی از نوع **مطلق** می باشند. نقض یک محدودیت مطلق به معنای حذف یک راه حل بالقوه می باشد.  
**محدودیت اولویت دار (نرم):**  
مثال، قرمز بر سبز ارجحیت دارد. اغلب بوسیله در نظر گرفتن هزینه برای انتساب متغیرها قابل بازنمایی می باشند.  $\leftarrow$  Constraint Optimization Problem

N. Razavi - AI course - 2005

9

## مثال: ریاضیات رمزي



- Variables:**  $FTUW$   
 $R O X_1 X_2 X_3$
- Domains:**  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Constraints:**  $\text{Alldiff}(F, T, U, W, R, O)$
- 
- 
- 

$$O + O = R + 10 \cdot X_1$$

-

$$X_1 + W + W = U + 10 \cdot X_2$$

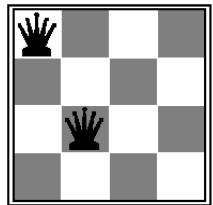
-

$$X_2 + T + T = O + 10 \cdot X_3$$

$\nabla \quad \nabla \quad \nabla \quad \nabla \quad \nabla \quad \nabla$

10

## مثال: ۴-وزیر به عنوان CSP



در هر ستون یک وزیر فرض کنید.

**متغیرها:**  $Q_1, Q_2, Q_3, Q_4$

**دامنه ها:**  $D_i = \{1, 2, 3, 4\}$

**محدودیت ها:**

$Q_1 = 1$   $Q_2 = 3$   $Q_i \neq Q_j$  (دو وزیر نمی توانند در یک سطر قرار بگیرند)

$|Q_i - Q_j| \neq |i - j|$  (یا در یک قطر)

هر محدودیت به مجموعه ای از مقادیر مجاز برای متغیرهایش ترجمه می شود،  
مثال:

مقادیر  $(Q1, Q2)$  می توانند مانند زیر باشند:

$(1, 3), (1, 4), (2, 4), (3, 1), (4, 1), (4, 2)$

N. Razavi - AI course - 2005

11

## مسائل اضافه محدودیت در دنیای واقعی

- مسائل انتسابی (assignment):**  
مثال، چه کسی چه کلاسی را درس می دهد.
- مسائل تعیین جدول زمانبندی (Timetabling):**  
مثال، کدام کلاس، کجا و کی ارائه می شود؟
- زمانبندی حمل و نقل (transportation):**
- زمانبندی کارخانه (factory scheduling):**
- توجه:** بسیاری از مسائل در دنیای واقعی شامل متغیرهایی با مقادیر حقیقی می باشند.

N. Razavi - AI course - 2005

12

## فرموله سازی جستجوی استاندارد (افزایشی)

- حالات توسط مقادیری که تا کنون انتساب یافته اند، تعریف می شوند.
- **حالت اولیه:** تمام متغیرها بدون مقدار، یعنی انتساب تهی  $\{\}$
- **عملگرها:** انتساب مقدار به یک متغیر بدون مقدار به طوری که با انتساب فعلی درگیری ایجاد نکند.  $\leftarrow$  در صورت عدم وجود انتساب های مجاز شکست می خورد.
- **تست هدف:** انتساب فعلی کامل باشد.
- **هزینه مسیر:** هزینه یکسان برای تمام مراحل

- 1) برای تمام  $CSP$  ها یکسان است!
  - 2) هر پاسخ در عمق  $n$  با  $n$  متغیر ظاهر می شود.  $\leftarrow$  استفاده از جستجوی عمقی
  - 3) در عمق  $l$ ، فاکتور انشعاب ( $b$ ) برابر است با  $(n-l)^*$ .
- بنابراین تعداد برگهای درخت جستجو برابر است با  $!!!! n! * d^n$
- $$(nd) * [(n-1)d] * [(n-2)d] * \dots * d = n! d^n$$
- مثالا در ۸ وزیر:  $8! 8^8$

N. Razavi - AI course - 2005

13

## پیاده سازی

### datatype CSP-STATE

**components:** UNASSIGNED, a list of variables not yet assigned  
ASSIGNED, a list of variable that have value

### datatype CSP-VAR

**components:** NAME, for i/o purpose  
DOMAIN, a list of possible values  
VALUE, current value (if any)

- محدودیت ها را می توان به دو روش نمایش داد:

- **صريح:** به عنوان مجموعه ای از مقادیر مجاز، با
- **ضمنی:** بواسطه تابعی که ارضاء محدودیت ها را بررسی می کند.

N. Razavi - AI course - 2005

14

## پیمیدگی این (هیافت

- حداکثر عمق فضا ( $m = n!!$ ) (تعداد متغیرها)
- عمق حالت پاسخ ( $d = n!!$ ) (تمام متغیرها مقدار دارند)
- الگوریتم جستجو؟؟ جستجوی اول-عمق
- فاکتور انشعاب  $b = \sum_i |D_i|$  (در بالای درخت)
- این ها می توانند با توجه به نکات زیر بهبود یابند:
  - 1) ترتیب انتساب مقادیر به متغیرها اهمیت ندارد، پس بسیاری از مسیرها معادل یکدیگر می باشند. (در ۸ وزیر اندازه فضای حالت از  $8! 8^8$  به  $8^8$  کاهش می یابد)
  - 2) انتساب های بعدی نمی توانند یک محدودیت نقض شده را تصحیح کنند. (مثلا اگر در ۸ وزیر دو وزیر اول در یک ردیف قرار بگیرند، جستجوی عمقی  $8^6$  حالت باقیمانده را امتحان می کند تا بفهمد پاسخی وجود ندارد)  $\leftarrow$  **جستجوی عقب گرد**.

N. Razavi - AI course - 2005

15

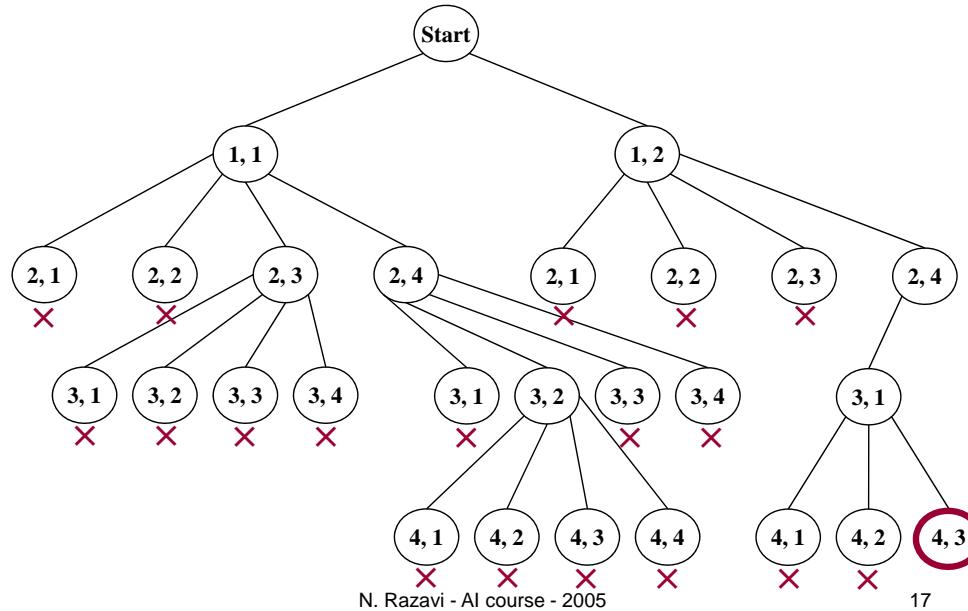
## Backtracking

- انتساب متغیرها جایه **پذیر** است مثلا  $[Q_1=1, Q_2=3]$  برابر است با  $[Q_2=3, Q_1=1]$
- در هر گره تنها باید انتساب های یک متغیر در نظر گرفته شود؛ پس  $b=d$  و تعداد برگهای  $d^n$
- جستجوی اول-عمق با انتساب های یک متغیر جداگانه در  $CSP$  ها، **backtracking** نام دارد.
- قرار دادن یک آزمون قبل از بسط گره ها که بررسی می کند آیا تاکنون محدودیتی نقض شده باشد، دیگر این گره بسط داده نمی شود و جستجو به عقب باز می گردد.
- می تواند  $n$ -وزیر را تا حدود  $n=25$  حل کند.

N. Razavi - AI course - 2005

16

## مثال: جستجوی عقبگرد برای ۴-وزیر



## جستجوی عقبگرد

```

function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING({}, csp)
function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment according to Constraints[csp] then
            add { var = value } to assignment
            result  $\leftarrow$  RECURSIVE-BACKTRACKING(assignment, csp)
            if result  $\neq$  failure then return result
            remove { var = value } from assignment
    return failure
  
```

N. Razavi - AI course - 2005

18

## جستجوی عقبگرد



N. Razavi - AI course - 2005

19

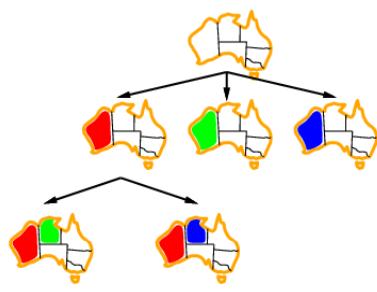
## جستجوی عقبگرد



N. Razavi - AI course - 2005

20

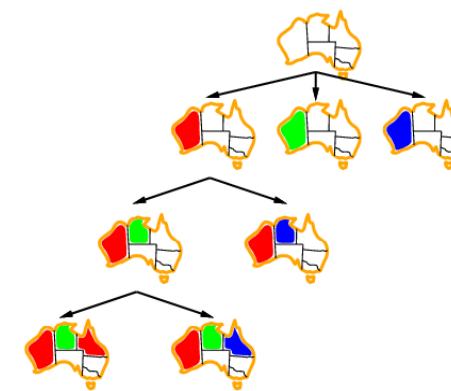
## جستجوی عقبگرد



N. Razavi - AI course - 2005

21

## جستجوی عقبگرد



N. Razavi - AI course - 2005

22

## بهبود کارآیی *backtracking*

- افزایش سرعت توسط روش های جستجوی **همه منظوره**:

- در مرحله بعد باید به کدام متغیر مقدار مداده شود؟
- مقادیر آن باید به چه ترتیبی امتحان شوند؟
- آیا می توان شکست های حتمی را زودتر تشخیص داد؟

فرض کنید در عقبگرد در مسأله ۸-وزیر، ۶ وزیر اول به گونه ای قرار گرفته اند که قرار دادن هشتمن وزیر را غیر ممکن می سازند. عقب گرد تمام ممکنهای ممکن برای وزیر هفتم را چک می کند، اگرچه مسأله غیر قابل حل می باشد.

- آیا می توان از ساختار مسأله بهره گرفت؟

N. Razavi - AI course - 2005

23

## متغیر با بیشترین محدودیت

- متغیر با بیشترین محدودیت :

- متغیر را انتخاب کن که کمترین مقادیر معتبر را دارد.



- هیوریستیک **کمترین مقادیر باقیمانده** (MRV)

N. Razavi - AI course - 2005

24

## متغیر با بیشترین محدودیت

- حل درگیری (tie) میان متغیرهایی که بیشترین محدودیت را دارند
- متغیر با بیشترین محدودیت: **هیوریستیک درجه**
  - متغیر با بیشترین محدودیت را روی مقادیر باقیمانده انتخاب کن



N. Razavi - AI course - 2005

25

## بررسی (و) به جلو (Forward checking)

- ایده:**
- مراقب مقادیر مجاز باقیمانده برای متغیرهای بدون مقدار باش.
- زمانی که یک متغیر هیچ مقدار مجاز باقیمانده ای ندارد، به جستجو پایان بده.



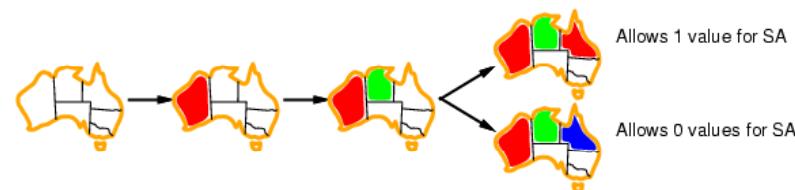
WA	NT	Q	NSW	V	SA	T
■■■	■■■	■■■	■■■	■■■	■■■	■■■

N. Razavi - AI course - 2005

27

## مقدار با کمترین محدودیت (Least constraining value)

- برای یک متغیر، مقداری را که کمترین محدودیت را ایجاد می کند انتخاب کن.
- مقداری که کمترین مقادیر را از متغیرهای باقیمانده حذف می کند



- با ترکیب این هیوریستیک ها مسئله ۱۰۰۰-وزیر قابل حل می شود.

N. Razavi - AI course - 2005

26

## بررسی (و) به جلو (Forward checking)

- ایده:**

- مراقب مقادیر مجاز باقیمانده برای متغیرهای بدون مقدار باش.
- زمانی که یک متغیر هیچ مقدار مجاز باقیمانده ای ندارد، به جستجو پایان بده.



WA	NT	Q	NSW	V	SA	T
■■■	■■■	■■■	■■■	■■■	■■■	■■■
■■■	■■■	■■■	■■■	■■■	■■■	■■■

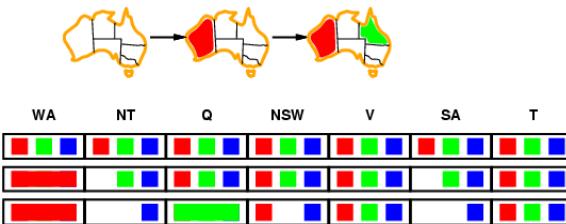
N. Razavi - AI course - 2005

28

## بررسی و به جلو (Forward checking)

### • ایده:

- مراقب مقادیر مجاز باقیمانده برای متغیرهای بدون مقدار باش.
- زمانی که یک متغیر هیچ مقدار مجاز باقیمانده ای ندارد، به جستجو پایان بده.



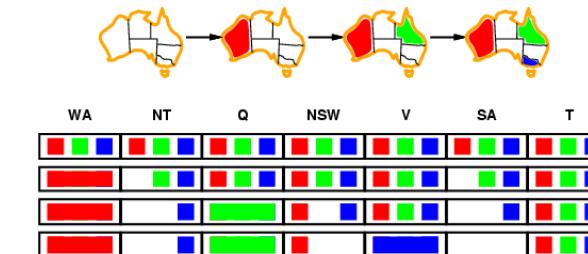
N. Razavi - AI course - 2005

29

## بررسی و به جلو (Forward checking)

### • ایده:

- مراقب مقادیر مجاز باقیمانده برای متغیرهای بدون مقدار باش.
- زمانی که یک متغیر هیچ مقدار مجاز باقیمانده ای ندارد، به جستجو پایان بده.

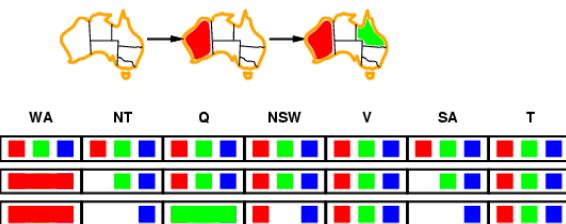


N. Razavi - AI course - 2005

30

## انتشار محدودیت

- بررسی رو به جلو محدودیت ها را از متغیرهای انتساب یافته به متغیرهای انتساب نیافته منتشر می کند، اما تمام شکست ها را نمی تواند در زودترین زمان ممکن تشخیص دهد.



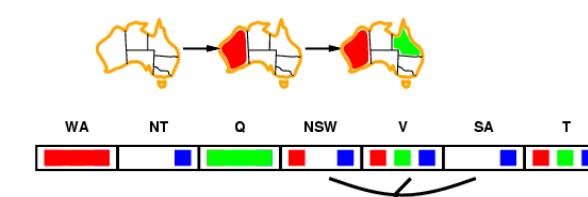
- NT و SA هر دو نمی توانند آبی باشند!
- انتشار محدودیت به طور مکرر بر محدودیت ها به طور محلی تأکید دارد.

N. Razavi - AI course - 2005

31

## سازگاری کمان (Arc consistency)

- ساده ترین شکل انتشار، هر کمان را **سازگار** می کند.
- $Y \rightarrow X$  سازگار است اگر و فقط اگر بازه هر بُرخی مقدار  $X$  مقدار مجاز  $Y$  موجود باشند.



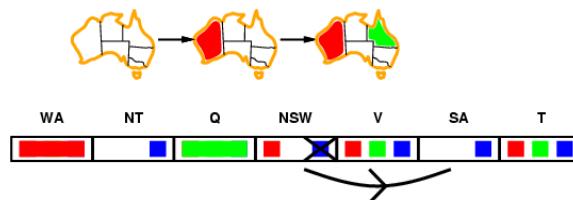
N. Razavi - AI course - 2005

32

## سازگاری کمان

### (Arc consistency)

- ساده ترین شکل انتشار، هر کمان را **سازگار** می کند.
- $X \rightarrow Y$  سازگار است اگر و فقط اگر بازه هر مقدار  $X$  برخی مقادیر مجاز  $Y$  موجود باشند.



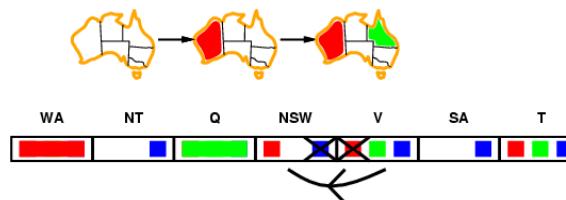
N. Razavi - AI course - 2005

33

## سازگاری کمان

### (Arc consistency)

- ساده ترین شکل انتشار، هر کمان را **سازگار** می کند.
- $X \rightarrow Y$  سازگار است اگر و فقط اگر بازه هر مقدار  $X$  برخی مقادیر مجاز  $Y$  موجود باشند.



اگر  $X$  مقداری را از دست بدهد، همسایه های  $X$  نیاز به بررسی مجدد دارند.

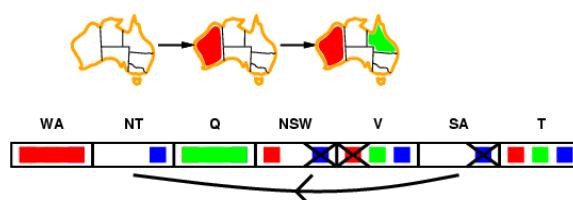
N. Razavi - AI course - 2005

34

## سازگاری کمان

### (Arc consistency)

- ساده ترین شکل انتشار، هر کمان را **سازگار** می کند.
- $X \rightarrow Y$  سازگار است اگر و فقط اگر بازه هر مقدار  $X$  برخی مقادیر مجاز  $Y$  موجود باشند.



سازگاری کمان شکست ها را زودتر از FC تشخیص می دهد.  
و می تواند به عنوان پیش پردازنده و یا بعد از هر انتساب اجرا شود.

N. Razavi - AI course - 2005

35

## الگوریتم سازگاری کمان AC-3

function **AC-3(*csp*)** returns the CSP, possibly with reduced domains

inputs: *csp*, a binary CSP with variables  $\{X_1, X_2, \dots, X_n\}$

local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty do

$(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\text{queue})$

if **RM-INCONSISTENT-VALUES( $X_i, X_j$ )** then

for each  $X_k$  in **NEIGHBORS[ $X_i$ ]** do

add  $(X_k, X_i)$  to *queue*

function **RM-INCONSISTENT-VALUES( $X_i, X_j$ )** returns true iff remove a value

*removed*  $\leftarrow$  false

for each  $x$  in **DOMAIN[ $X_i$ ]** do

if no value  $y$  in **DOMAIN[ $X_j$ ]** allows  $(x, y)$  to satisfy **constraint( $X_i, X_j$ )**

then delete  $x$  from **DOMAIN[ $X_i$ ]**; *removed*  $\leftarrow$  true

return *removed*

• پیچیدگی زمانی:  $O(n^2d^3)$

N. Razavi - AI course - 2005

36

# جستجوی محلی برای CSP

- تپه نورده و SA با حالات کامل کار می کنند؛ یعنی تمام متغیرها باید دارای مقدار باشند.
- برای اعمال آنها بر مسائل CSP:
  - داشتن حالتی با محدودیتهای نقض شده باید مجاز باشد.
  - عملگرها باید بتوانند به متغیرها مقادیر جدید بدهند.
- انتخاب متغیر: به صورت تصادفی یک متغیر در گیر را انتخاب کن
  - هیوریستیک *:min-conflicts*
  - مقداری را انتخاب کن که کمترین تعداد محدودیت ها را نقض می کند، یعنی، تپه نورده با هیوریستیک "تعداد کل محدودیت های نقض شده"

N. Razavi - AI course - 2005

37

## مثال : ۸- وزیر

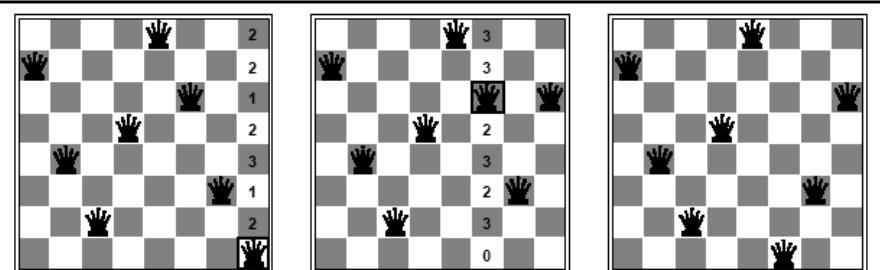


Figure 5.9 A two-step solution for an 8-queens problem using min-conflicts. At each stage, a queen is chosen for reassignment in its column. The number of conflicts (in this case, the number of attacking queens) is shown in each square. The algorithm moves the queen to the min-conflict square, breaking ties randomly.

N. Razavi - AI course - 2005

39

# الگوریتم Min-Conflicts

```

function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
  inputs: csp, a constraint satisfaction problem
          max_steps, the number of steps allowed before giving up

  current  $\leftarrow$  an initial complete assignment for csp
  for i = 1 to max_steps do
    if current is a solution for csp then return current
    var  $\leftarrow$  a randomly chosen, conflicted variable from VARIABLES[csp]
    value  $\leftarrow$  the value v for var that minimizes CONFLICTS(var, v, current, csp)
    set var = value in current
  return failure

```

Figure 5.8 The MIN-CONFICTS algorithm for solving CSPs by local search. The initial state may be chosen randomly or by a greedy assignment process that chooses a minimal-conflict value for each variable in turn. The CONFLICTS function counts the number of constraints violated by a particular value, given the rest of the current assignment.

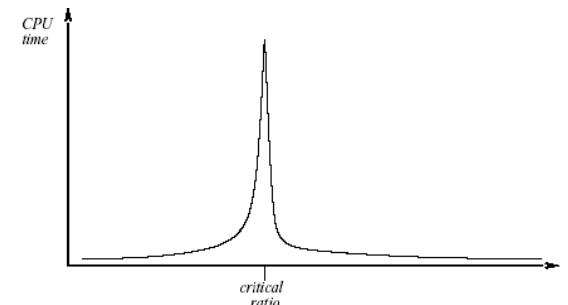
N. Razavi - AI course - 2005

38

## کارآیی min-conflicts

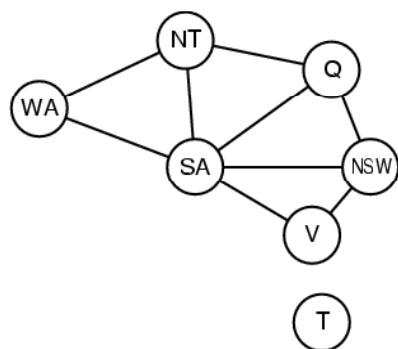
حل  $n$ -وزیر با  $n$  دلخواه (مثلا 10,000,000) در یک زمان تقریبا ثابت با شروع از یک وضعیت تصادفی با احتمال بالا (میانگین ۵۰ مرحله) همین موضوع به نظر می رسد برای هر CSP با شروع تصادفی درست باشد، به غیر از یک ناحیه باریک از نسبت  $R$ .

$$R = (\text{num. of constraints}) / (\text{num. of variables})$$



40

## ساختار مسئله



- و جزیره اصلی زیرمسایل مستقل هستند.
- و با توجه به مولفه های همبند در گراف محدودیت قابل شناسایی هستند.

N. Razavi - AI course - 2005

41

## ساختار مسئله (ادامه)

- فرض کنید هر زیرمسئله شامل  $c$  متغیر از کل  $n$  متغیر باشد
- هزینه راه حل در بدترین حالت خطی (برحسب  $n$ ) و برابر  $n/c$  می باشد

مثال:  $n = 80, d = 2, c = 20$

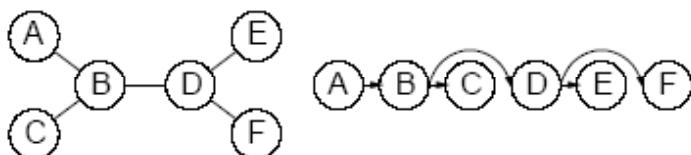
- $2^{80} = 4$  billion years at 10 million nodes/sec
- $4.2^{20} = 0.4$  seconds at 10 million nodes/sec

N. Razavi - AI course - 2005

42

## الgoritم برای CSP های دارای ساختار درختی

- یک متغیر را به عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگها به گونه ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.



- بازه  $\frac{n}{c}$  تا  $2^n$  عمل زیر را انجام بده

REMOVEINCONSISTENT( $Parent(X_j), X_j$ )

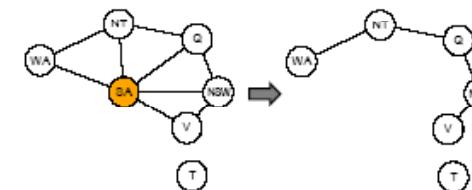
- برای  $j$  از یک تا  $n$  ، مقدار  $X_j$  را به طور سازگار با  $Parent(X_j)$  تعیین کن  
پیجیدگی زمانی:  $O(n \cdot d^2)$

N. Razavi - AI course - 2005

43

## CSP های دارای ساختار شبیه درختی

- شرطی سازی: به یک متغیر مقدار بده؛ دامنه همسایه های آن را هرس کن



- شرطی برشی: مجموعه ای از متغیرها (در تمام حالات ممکن) را مقدار دهی کن به طوریکه گراف محدودیت باقیمانده یک درخت شود.

اندازه برش  $c = d^2 \leftarrow$  زمان اجرا  $O(d^c \cdot (n - c) \cdot d^2)$

N. Razavi - AI course - 2005

44

## (هیافت دو): تجزیه درختی

## (هیافت دو): تجزیه درختی

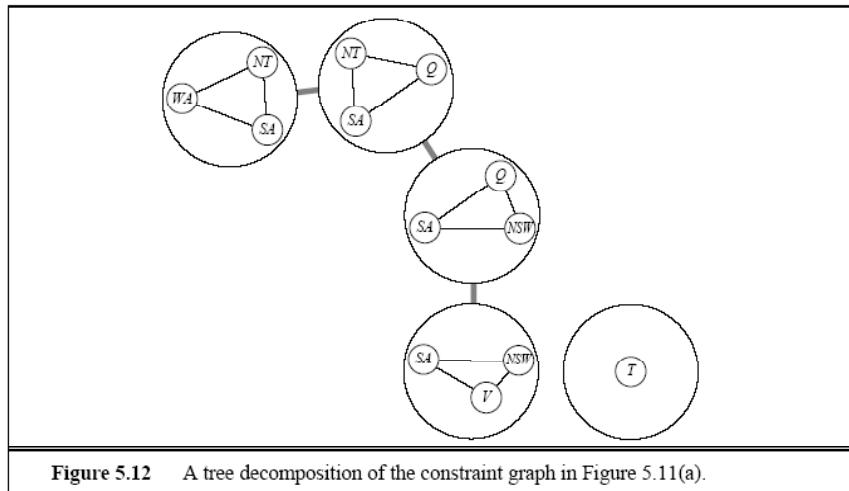


Figure 5.12 A tree decomposition of the constraint graph in Figure 5.11(a).

N. Razavi - AI course - 2005

45

N. Razavi - AI course - 2005

46

## (هیافت دو): تجزیه درختی

### • حل تجزیه درختی:

- هر زیر مسئله را به صورت یک **mega-variable** در نظر می گیریم که دامنه آن مجموعه تمام راه حل های ممکن آن زیر مسئله می باشد.
- سپس محدودیت های میان زیر مسایل را با استفاده از الگوریتم کارای درخت حل می کنیم. مثلاً اگر پاسخ اولین زیر مسئله انتساب زیر باشد:

$$\{ WA = red, SA = blue, NT = green \}$$

آنگاه تنها پاسخ سازگار برای زیر مسئله بعدی انتساب زیر می باشد:

$$\{ SA = blue, NT = green, Q = red \}$$

**پیچیدگی زمانی:**  $O(n \cdot d^{w+1})$

N. Razavi - AI course - 2005

47

## خلاصه

- مسائل CSP نوع خاصی از مسائل می باشند:
  - حالات توسط مقادیر مجموعه ای از متغیرها تعريف می شوند.
  - تست هدف توسط محدودیت ها روی مقادیر متغیرها تعريف می شود.
  - جستجوی عقبگرد = جستجوی اول-عمق که در هر گره یک متغیر مقدار گرفته باشد.
  - هیوریستیک های ترتیب متغیرها و انتخاب مقادیر بسیار کمک کننده می باشند.
  - بررسی رو به جلو از انتساب های منجر به شکست جلوگیری می کند.
  - انتشار محدودیت کارهای بیشتری برای محدود کردن مقادیر و تشخیص ناسازگاری ها انجام می دهد.
  - در عمل معمولاً جستجوی محلی min-conflicts مؤثر می باشد.

N. Razavi - AI course - 2005

48

## مقدمه

- تصمیم های بهینه
- هرس آلفا-بتا
- تصمیم های بلادرنگ و غیر کامل

## جستجوی (قابلی

فصل ششم

سید ناصر رضوی

Email: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۴

N. Razavi - AI course - 2005

2

## تصمیم های بهینه در بازی ها

- بازی های دو نفره (MIN و MAX)
- ابتدا MAX بازی می کند، سپس MIN و ...
- بازی به عنوان یک نوع از مسئله جستجو  
- **حالت اولیه:** شامل موقعیت صفحه و نوبت بازیکن
- **تابع جانشین:** لیستی از زوج های (move, state) (را بر می گرداند)
- **تست ترمینال:** تعیین کننده پایان بازی (حالات های ترمینال)
- **تابع سودمندی:** به هر حالت پایانی یک مقدار عددی می دهد
- درخت بازی: توسط حالت اولیه و حرکت های قانونی برای هر طرف مشخص می شود

N. Razavi - AI course - 2005

3

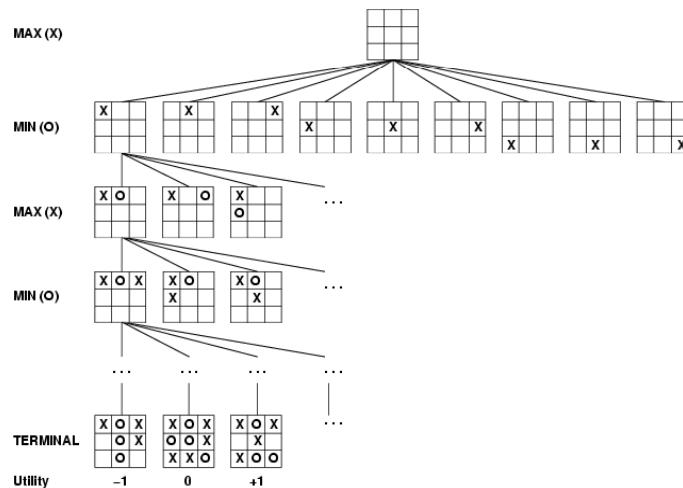
## بازی در مقایسه با مسائل جستجو

- رقیب «غیر قابل پیش بینی» ← مشخص کردن یک حرکت برای هر پاسخ ممکن از طرف رقیب
- محدودیت های زمانی ← مetasfane برای یافتن هدف باید تقریب زد

N. Razavi - AI course - 2005

4

## دخت بازی (۲-نفره، قطعی)

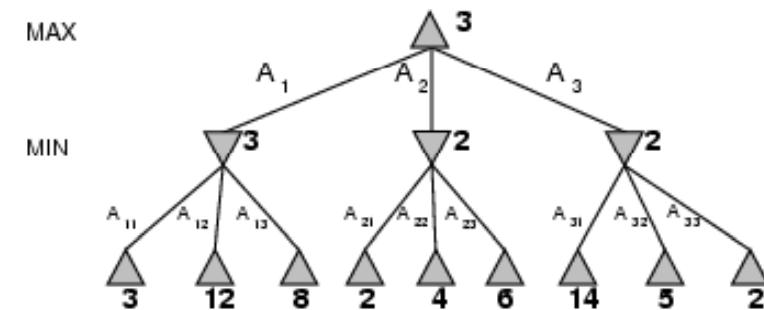


N. Razavi - AI course - 2005

5

## Minimax

- بازی عالی در بازی های قطعی
- ایده: انتخاب حرکت به موقعیتی با بیشترین مقدار **minimax**
- = بهترین امتیاز قابل دستیابی در برابر بهترین بازی
- مثال:



N. Razavi - AI course - 2005

6

## استراتژی بهینه

- با داشتن درخت بازی، استراتژی بهینه را می توان با در نظر گرفتن مقدار گره ها تعیین نمود: **minimax**

$\text{MINMAX-VALUE}(n) =$

$$\begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in \text{Successors}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MAX node} \\ \min_{s \in \text{Successors}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MIN node} \end{cases}$$

N. Razavi - AI course - 2005

7

## الگوریتم Minimax

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v
```

```
function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

N. Razavi - AI course - 2005

8

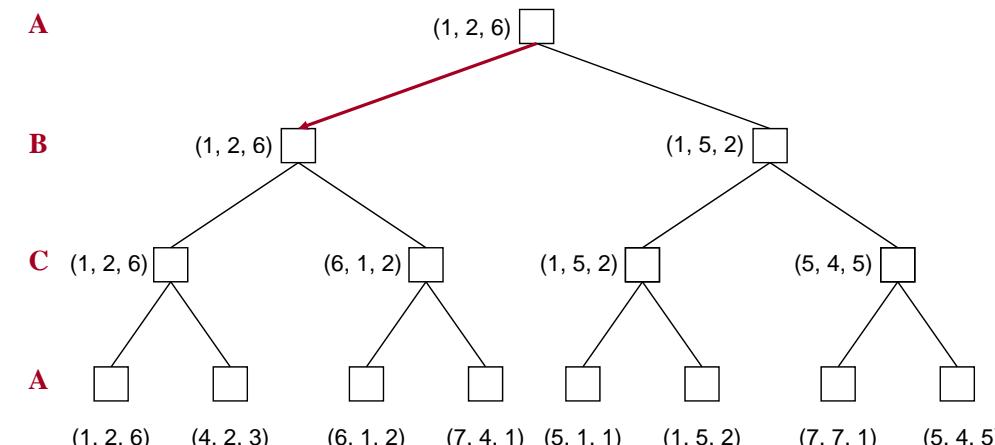
## خصوصیات Minimax

- کامل؟ بله (به شرط محدود بودن درخت)
- بهینه؟ بله (در مقابل رقیبی که بهینه بازی می کند)
- پیچیدگی زمانی؟  $O(b^m)$
- پیچیدگی حافظه؟  $O(bm)$  (کاوش اول-عمق)
- در شترنج  $b \approx 35$  و  $m \approx 100$  (در بازی های معقول)
- راه حل دقیق کاملا مقرن به صرفه نمی باشد.

N. Razavi - AI course - 2005

9

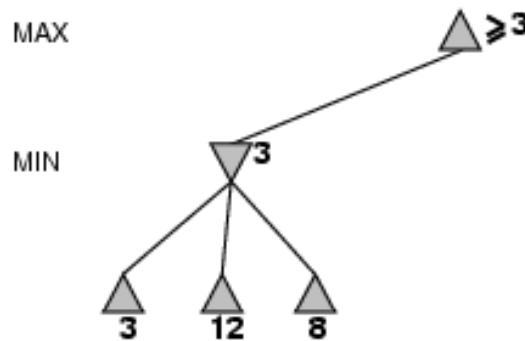
## تصمیمات بهینه در بازی های چند نفره



N. Razavi - AI course - 2005

10

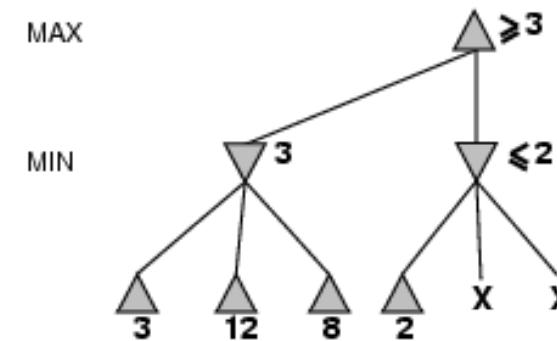
## مثال هرس آلفا-بتا



N. Razavi - AI course - 2005

11

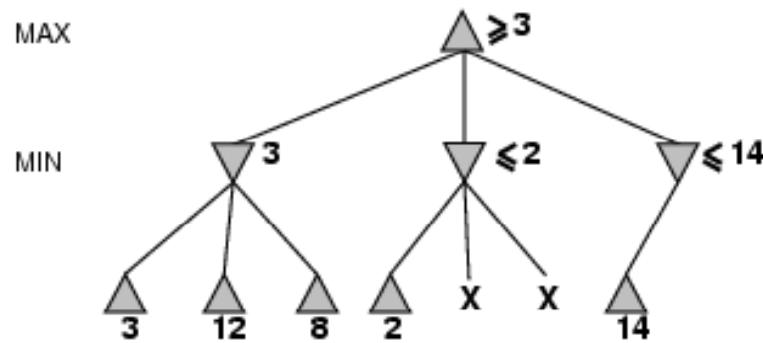
## مثال هرس آلفا-بتا



N. Razavi - AI course - 2005

12

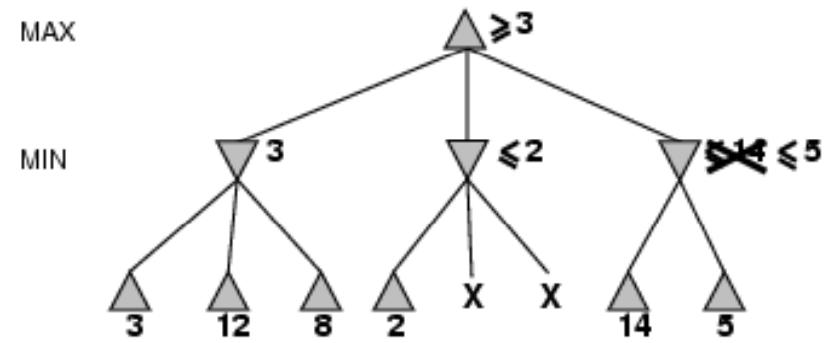
## مثال هرس آلفا-بتا



N. Razavi - AI course - 2005

13

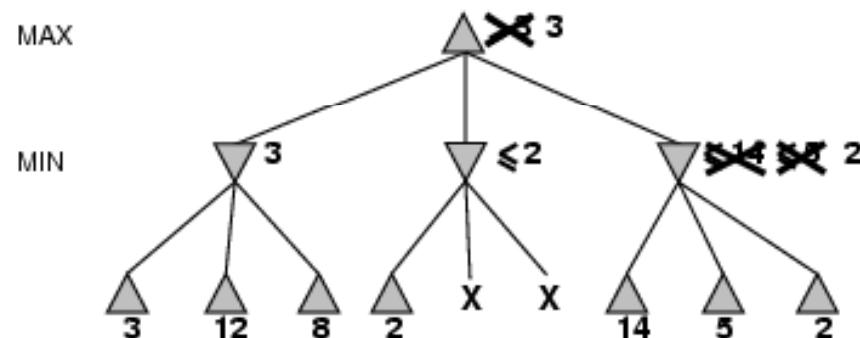
## مثال هرس آلفا-بتا



N. Razavi - AI course - 2005

14

## مثال هرس آلفا-بتا



MINIMAX-VALUE (root) =  $\max(\min(3, 12, 8), \min(2, x, y), \min(14, 5, 2))$   
 $= \max(3, \min(2, x, y), 2)$   
 $= \max(3, z, 2) \quad \text{where } z \leq 2$   
 $= 3$

N. Razavi - AI course - 2005

15

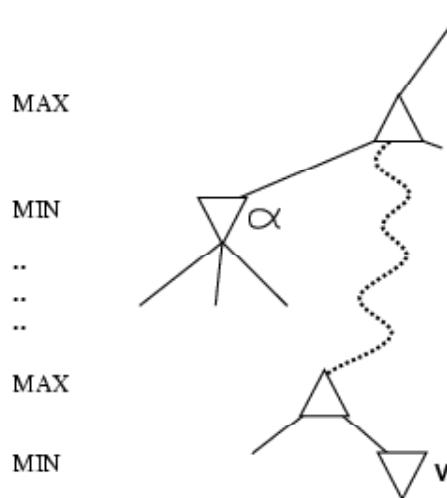
## خواص آلفا-بتا

- هرس کردن بر روی نتیجه نهایی **تاثیر ندارد**
- ترتیب خوب حرکت ها میزان تاثیر الگوریتم را بهبود می بخشد
- با «بهترین ترتیب»، پیچیدگی زمانی  $\mathcal{O}(b^{m/2})$  با «بهترین ترتیب»، پیچیدگی زمانی  $\mathcal{O}(b^m)$  است
- عمق جستجوی دو برابر
- یک مثال ساده از ارزش استدلال در مورد محاسبات مرتبط ( شکلی از **استدلال**)

N. Razavi - AI course - 2005

16

## مجه تسمیه



N. Razavi - AI course - 2005

- آلفا مقدار بهترین انتخاب (یعنی بالاترین مقدار) یافته شده تا کنون در هر نقطه انتخاب در طول مسیر برای *max* می باشد
- اگر *V* بدتر از آلفا باشد، *max* از آن اجتناب می کند
- آن شاخه را هرس می کند
- بنا نیز برای *min* مانند آلفا تعریف می شود

17

## الگوریتم آلفا-بتا

```
function ALPHA-BETA-SEARCH(state) returns an action
    inputs: state, current state in game
    v ← MAX-VALUE(state, -∞, +∞)
    return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
        α, the value of the best alternative for MAX along the path to state
        β, the value of the best alternative for MIN along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← -∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s, α, β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v
```

N. Razavi - AI course - 2005

18

## الگوریتم آلفا-بتا

```
function MIN-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
        α, the value of the best alternative for MAX along the path to state
        β, the value of the best alternative for MIN along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s, α, β))
        if v ≤ α then return v
        β ← MIN(β, v)
    return v
```

N. Razavi - AI course - 2005

19

## محدودیت های منابع

فرض کنید ۱۰۰ ثانیه زمان داریم و در هر ثانیه می توان  $10^4$  گره گسترش داد.

← در هر حرکت  $10^6$  گره

روش استاندارد:

- تست برش: مانند محدوده عمقی
- تابع ارزیابی:

= میزان تخمینی مطلوب بودن موقعیت

N. Razavi - AI course - 2005

20

## تابع ارزیابی

- در شطرنج، معمولاً مجموع وزن دار ویژگی ها

$$\text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

- مثال:  $w_1 = 9$  و

$$f_1(s) = (\# \text{ of white queens}) - (\# \text{ of black queens})$$

## برش جستجو

- MinimaxValue مانند MinimaxCutoff باشد جز اینکه:
- 1. Cutoff? جایگزین شده و Terminal? باشد
- 2. Utility جایگزین شده است Eval با

- آیا در عمل کار می کند؟

$$b^m = 10^6 \rightarrow m = 4$$

در شطرنج پیش بینی ۴ لایه نا امید کننده است!

- ۴ لایه = انسان مبتدی

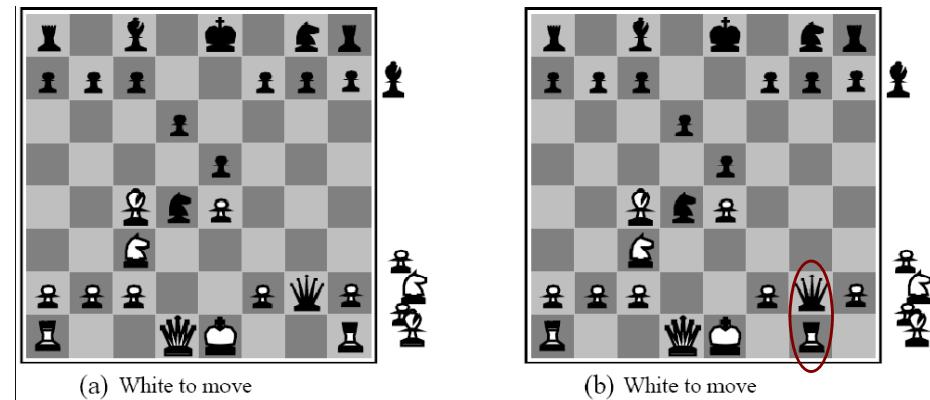
- ۸ لایه = کامپیوترهای معمولی و انسان های حرفه ای

- ۱۲ لایه = Deep Blue و کاسپاروف

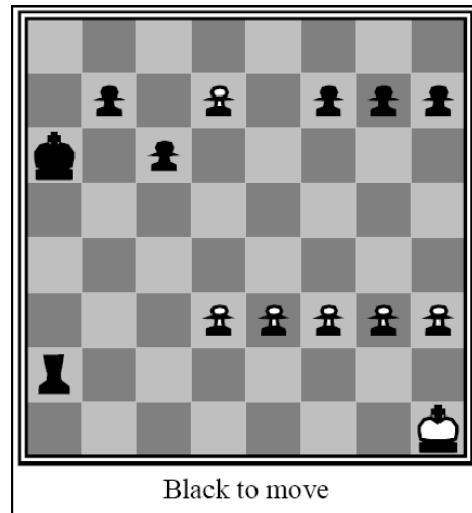
## پیاده سازی

- تعیین یک محدوده عمقی مانند  $d$
- باید طوری انتخاب شود که زمان لازم از قوانین بازی تجاوز نکند
- روش بهتر: استفاده از IDS تا وقتی که زمان داریم
- مشکلات:
- خطابه دلیل تقریبی بودن تابع ارزیابی (شکل 6.8b)
- به تابع تست Cutoff پیچیده تری نیاز داریم
- مشکل حالت های نا آرام ← جستجوی انتظار برای آرامش
- مشکل اثر افق (شکل 6.9) ←
- بهبود سخت افزار برای انجام جستجوهای عمیق تر
- گسترش های انفرادی

## شکل ۶-۶



## اثر افق



N. Razavi - AI course - 2005

25

## بازی های قطعی در عمل

- Deep Blue :Chess را شکست داد. Deep blue در یک ثانیه ۲۰۰ میلیون موقعیت را جستجو می کند و از توابع ارزیابی بسیار پیچیده ای استفاده می کند.

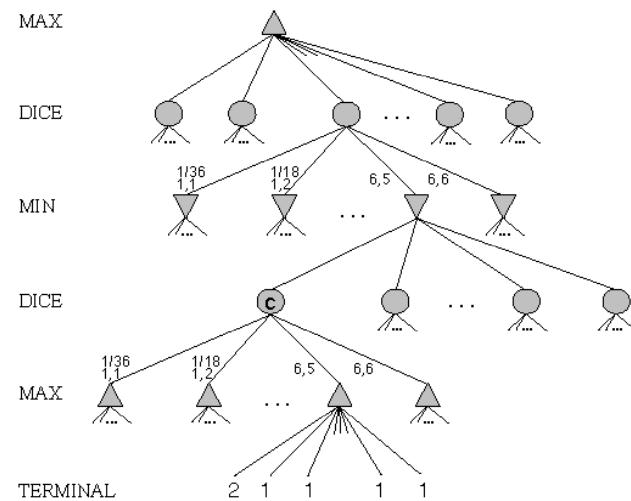
- Othello هستند) امتناع می ورزند.

- Go: قهرمان های انسانی از رقابت با کامپیوترها (که خیلی خوب می ورزند. در بازی Go فاکتور انشعاب بیشتر از ۳۰۰ می باشد، بنابراین اکثر برنامه ها برای ارائه حرکت های معقول از پایگاه های دانش الگویی استفاده می کنند.

N. Razavi - AI course - 2005

26

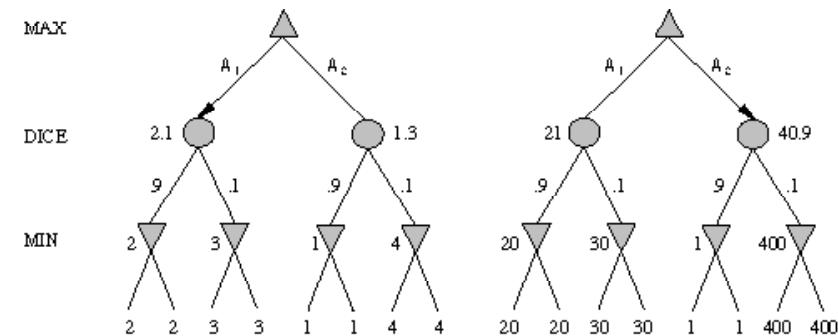
## بازی هایی که دارای عناصر شанс هستند



N. Razavi - AI course - 2005

27

## معنای تابع ارزیابی



- تغییری که ترتیب نسبی مقادیر را حفظ می کند، در تصمیم minimax تأثیری ندارد، اما می تواند تصمیم در مورد گره های شанс را تغییر دهد
- راه حل: تبدیلات خطی

N. Razavi - AI course - 2005

28

## خلاصه

- کار کردن بر روی بازی ها لذت بخش است!
- بازی ها نکات مهم متعددی را در AI به نمایش می گذارند
- عالی بودن ممکن نیست ← باید تقریب زد
- فکر کردن در مورد اینکه به چه چیزی فکر کنی ایده خوبی می باشد

## مقدمه

- عامل های مبتنی بر دانش
- محیط Wumpus
- منطق - مدل ها و استلزمات
- منطق گزاره ای (بولین)
- هم ارزی، اعتبار و صدق پذیری
- قوانین استنتاج و اثبات تئوری
  - زنجیره استنتاج رو به جلو (forward chaining)
  - زنجیره استنتاج رو به عقب (backward chaining)
  - رزولوشن

N. Razavi - AI course - 2005

2

## فصل هفتم

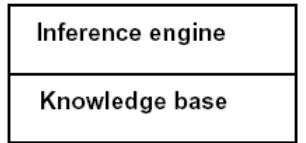
فصل هفتم

سید ناصر رضوی

E-mail:[razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۴

## پایگاه دانش



domain-independent algorithms

domain-specific content

- پایگاه دانش = مجموعه ای از **جملات** در یک زبان **رسمی**
- رهیافت **توصیفی** برای ایجاد یک عامل (یا سیستم های دیگر):
  - به عامل آنچه را که نیاز دارد بداند، بگو (TELL)
  - سپس عامل می تواند از خود بپرسد (ASK) که چه عملی انجام دهد - پاسخ ها باید از KB پیروی کنند.
- می توان عامل ها را در **سطح دانش** در نظر گرفت:
  - یعنی، چه می دانند، بدون توجه به چگونگی پیاده سازی
  - یا در سطح **پیاده سازی**:
    - یعنی، ساختارهای داده ای در KB و الگوریتم هایی که بر روی آنها کار می کنند.

N. Razavi - AI course - 2005

3

## یک عامل ساده مبتنی بر دانش

**function** KB-AGENT(*percept*) **returns** an *action*

**static:** *KB*, a knowledge base

*t*, a counter, initially 0, indicating time

```
TELL( KB, MAKE-PERCEPT-SENTENCE( percept, t) )
action  $\leftarrow$  ASK( KB, MAKE-ACTION-QUERY( t) )
TELL( KB, MAKE-ACTION-SENTENCE( action, t) )
t  $\leftarrow$  t + 1
return action
```

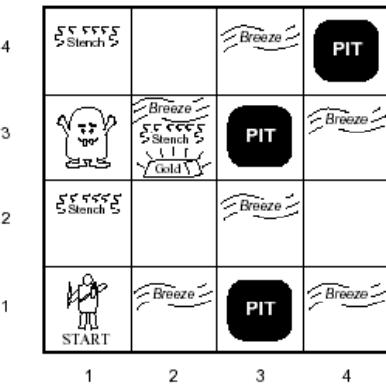
- عامل باید قادر باشد:

- حالات و اعمال و ... را بازنمایی کند.
- ادراک جدید دریافت کند.
- بازنمایی داخلی دنیا را بهنگام سازد.
- خواص پنهان دنیا و اعمال مناسب را نتیجه گیری کند.

N. Razavi - AI course - 2005

4

# Wumpus دنیای



N. Razavi - AI course - 2005

5

- **معیار کارآبی**
  - طلا +10000، مرگ 1000-
  - هر عمل 1، شلیک 10-

## • محیط

- خانه های مجاور وامپوس دارای بو هستند
- خانه های مجاور چاله ها دارای نسمیم هستند
- در خانه حاوی طلا، درخشش وجود دارد
- شلیک وامپوس را می کشد، اگر عامل رو به وامپوس باشد
- تنها یک شلیک موثر است
- اگر در خانه عامل طلا باشد، می تواند آنرا بردارد
- عامل می تواند طلا را رها کند

## • حسگرها

- نسم، درخشش، بو، ضربه و جیغ
- **محرك ها**
  - چرخش به چپ و راست
  - حرکت به جلو
  - برداشتن و رها کردن طلا و شلیک تیر

# مشخصات محیط وامپوس

- دسترس پذیر؟؟ خیر-- تنها ادراک محلی میسر می باشد
- قطعی؟؟ بله-- نتیجه اعمال کاملا مشخص است
- اپیزودیک؟؟ خیر
- ایستا؟؟ بله-- وامپوس و چاله ها حرکت نمی کنند
- گستته؟؟ بله
- **تک-عاملی؟؟** بله -- وامپوس اساساً یک ویژگی طبیعی است

N. Razavi - AI course - 2005

6

# کاوش دنیای وامپوس

OK			
OK	OK		

N. Razavi - AI course - 2005

7

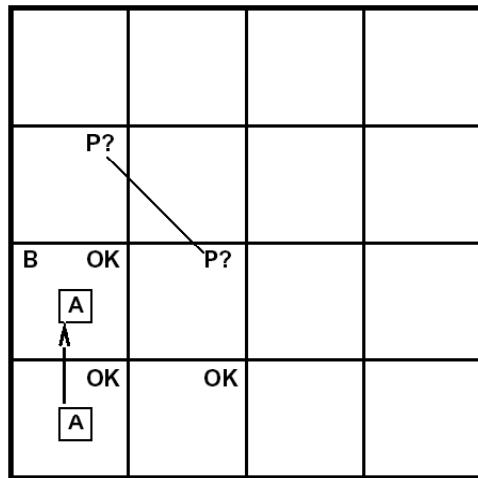
# کاوش دنیای وامپوس

B	OK		
A	OK	OK	

N. Razavi - AI course - 2005

8

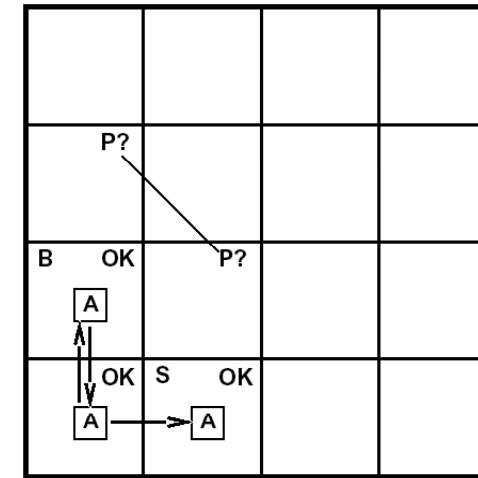
## کاوش دنیای وامپوس



N. Razavi - AI course - 2005

9

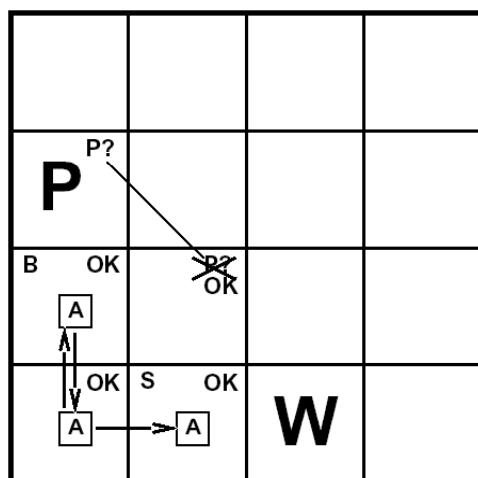
## کاوش دنیای وامپوس



N. Razavi - AI course - 2005

10

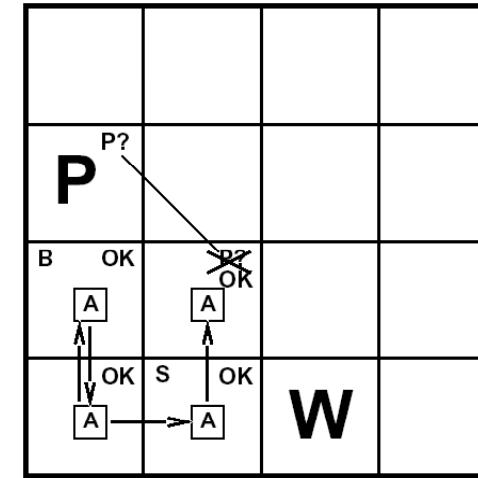
## کاوش دنیای وامپوس



N. Razavi - AI course - 2005

11

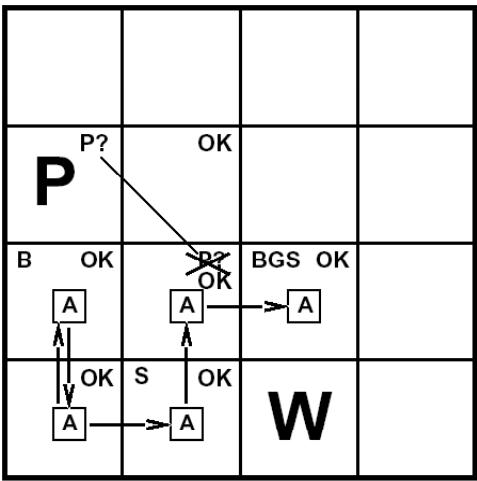
## کاوش دنیای وامپوس



N. Razavi - AI course - 2005

12

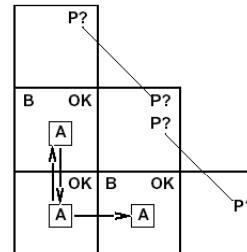
## کاوش دنیای وامپوس



N. Razavi - AI course - 2005

13

## موقعیت های تصمیم گیری دیگر



- در خانه های (1, 2) و (2, 1) نسیم احساس می شود  $\leftarrow$  عمل مطمئنی وجود ندارد
- با فرض توزیع یکنواخت چاله ها، احتمال چاله در (2, 2) بیشتر

- احساس بو در (1, 1)  $\leftarrow$  قادر به حرکت نیست
- استفاده از استراتژی اجبار:
- (1) به خانه روبرو شلیک کن
- (2) اگر وامپوس آنجا بوده  $\leftarrow$  مرده  $\leftarrow$  امن
- (3) اگر وامپوس آنجا نبوده  $\leftarrow$  امن

N. Razavi - AI course - 2005

14

## منطق

- منطق یک زبان رسمی برای بازنمایی دانش بطوری که بتوان از آن نتیجه گیری نمود.
- دستور ساختاری (syntax): ساختار جملات زبان را تعریف می کند
- معنا (semantic): معنای جملات را تعریف می کند
  - یعنی، تعریف درستی یک جمله در یک دنیا
- مثال: زبان ریاضی
- $x + 2 \geq y$  جمله
- $x^2 + y \geq$  جمله نیست
- $x + 2 \geq y$  در دنیایی با  $x=7, y=1$  درست و در دنیایی با  $x=0$  و  $y=6$  نادرست می باشد.

N. Razavi - AI course - 2005

15

## استلزم (entailment)

- استلزم بدین معناست که چیزی از چیز دیگری پیروی می کند:  $KB \models \alpha$
- پایگاه دانش  $KB$  مستلزم جمله  $\alpha$  است، اگر و فقط اگر  $\alpha$  در تمام دنیاهایی که در آن  $KB$  درست است، درست باشد.

- مثال:  $x + y = 4$  مستلزم  $x + y = 4$  می باشد.
- استلزم رابطه ایست که بین ساختار جملات (syntax) و بر مبنای معنای جملات تعریف می شود.

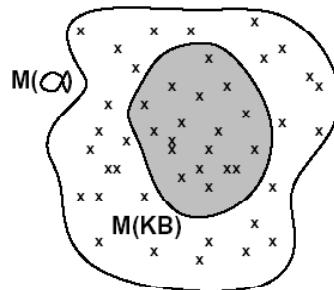
N. Razavi - AI course - 2005

16

## مدل ها

- منطق دانان عموماً بر حسب **مدل ها** فکر می کنند، که بطور رسمی دنیاهای ساخت یافته ای می باشد که درستی را می توان نسبت به آنها ارزیابی کرد.

- می گوییم  $m$  **مدلی** از جمله  $\alpha$  می باشد اگر  $\alpha$  در  $m$  درست باشد
- $M(\alpha)$  مجموعه تمام مدل های  $\alpha$  می باشد
- بنابراین  $\alpha \models KB$  اگر و فقط اگر  $\models \alpha$

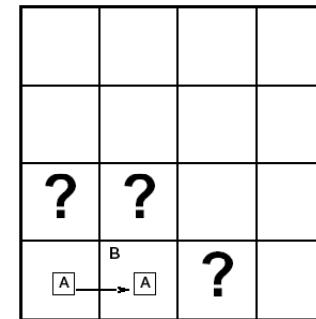


N. Razavi - AI course - 2005

17

## استلزاھ در دنیای وامپوس

- موقعیت پس از  $[1, 1]$ ، رفتن به راست، دریافت نسیم در  $[2, 1]$



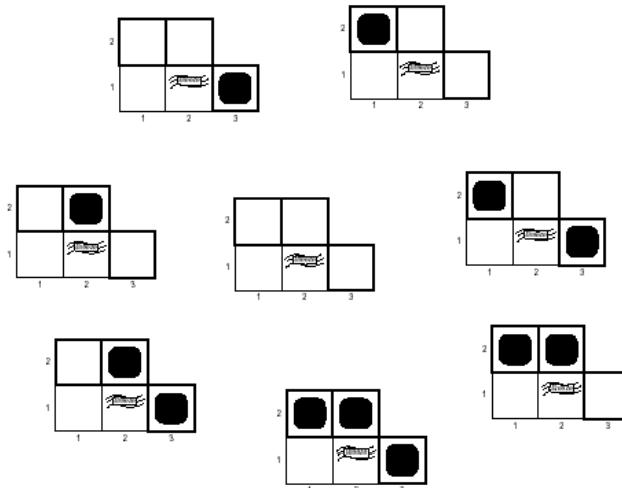
- مدلهای ممکن برای  $?$  ها را تنها با فرض چاله ها در نظر بگیرید

- سه انتخاب بولین  $\leftarrow$  هشت مدل مختلف

N. Razavi - AI course - 2005

18

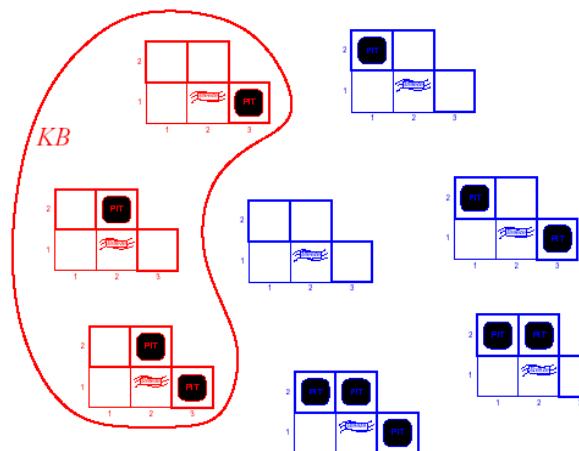
## مدل های وامپوس



N. Razavi - AI course - 2005

19

## مدل های وامپوس

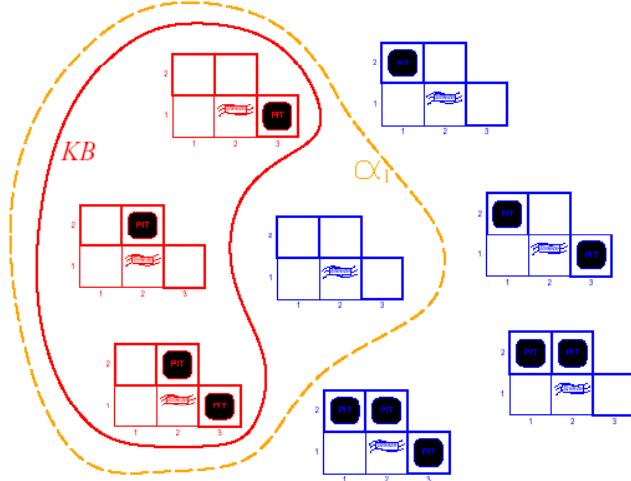


KB = Wumpus-world rules + observations

N. Razavi - AI course - 2005

20

## مدل های وامپوس



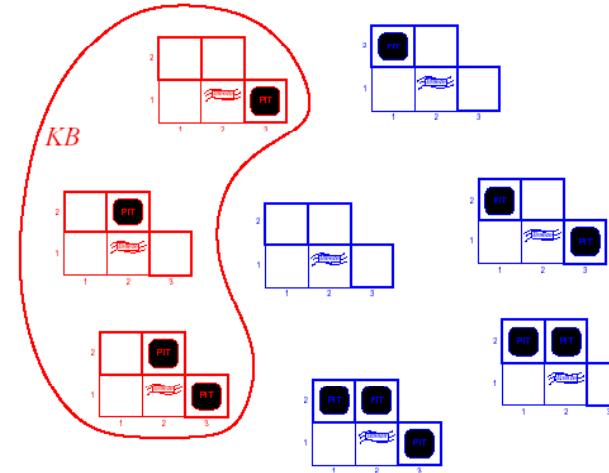
$KB = \text{Wumpus-world rules} + \text{observations}$

$\alpha_1 = "[1, 2] \text{ is safe}", KB \models \alpha_1$ , proved by model checking

N. Razavi - AI course - 2005

21

## مدل های وامپوس

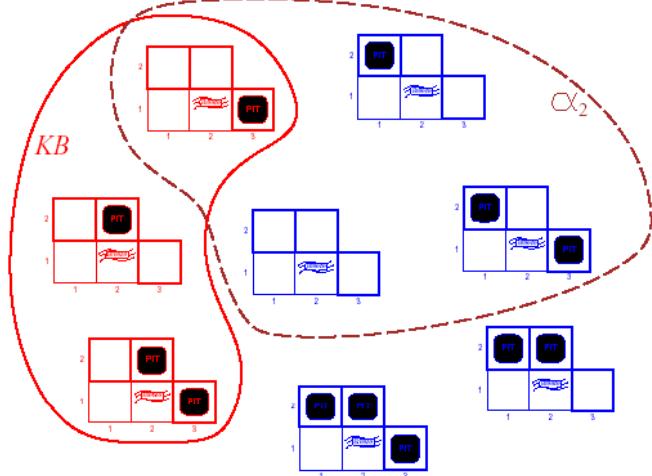


$KB = \text{Wumpus-world rules} + \text{observations}$

N. Razavi - AI course - 2005

22

## مدلهای وامپوس



$KB = \text{Wumpus-world rules} + \text{observations}$

$\alpha_2 = "[2, 2] \text{ is safe}", KB \not\models \alpha_2$

N. Razavi - AI course - 2005

23

## استنتاج (Inference)

- $KB \models_i \alpha = KB \models \alpha$  جمله  $\alpha$  بوسیله رویه  $i$  از  $KB$  قابل اشتقاق می باشد.
- نتایج  $KB$  مانند یک انبار کاه می باشد، و  $\alpha$  مانند یک سوزن استلزم = سوزن در انبار کاه؛ استنتاج = یافتن سوزن
- صحت (soundness): رویه  $i$  صحیح است اگر

$$KB \models_i \alpha \Rightarrow KB \models \alpha$$

- کامل بودن (completeness): رویه استنتاج  $i$  کامل است اگر

$$KB \not\models \alpha \Rightarrow KB \not\models_i \alpha$$

- مثال: در منطق مرتبه اول (First Order Logic) یک رویه استنتاج کامل و صحیح وجود دارد.

N. Razavi - AI course - 2005

24

## منطق گزاره ای : ساختار

- منطق گزاره ای ساده ترین نوع منطق است - برای بیان ایده های ساده و مبنایی
- سیمboleای گزاره ای  $P_1, P_2, \dots$  هر کدام یک جمله می باشند
- ثابت های گزاره ای می باشند و هر کدام به تنها یک جمله اند  $True$  و  $False$
- اگر  $S$  جمله باشد، آنگاه  $\neg S$  نیز یک جمله است (نفیض)
- اگر  $S_1$  و  $S_2$  جمله باشند،  $S_1 \wedge S_2$  نیز یک جمله است (ترکیب عطفی)
- اگر  $S_1$  و  $S_2$  جمله باشند،  $S_1 \vee S_2$  نیز یک جمله است (ترکیب فصلی)
- اگر  $S_1$  و  $S_2$  جمله باشند،  $S_1 \Rightarrow S_2$  نیز یک جمله است (ترکیب شرطی)
- اگر  $S_1$  و  $S_2$  جمله باشند،  $S_1 \Leftrightarrow S_2$  نیز یک جمله است (ترکیب دوشرطی)

N. Razavi - AI course - 2005

25

## منطق گزاره ای: محذا

- هر مدل درست بودن/غلط بودن سیمboleای گزاره ای را مشخص می کند
- مثلا  $P_{1,2}$  (درست)،  $P_{2,2}$  (درست)،  $P_{3,1}$  (نادرست)
- قوانین ارزیابی درستی نسبت به یک مدل  $m$

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

N. Razavi - AI course - 2005

26

## جملات دنیای وامپوس

- اجازه دهد  $i, j$  درست باشد، اگر و فقط اگر در خانه  $[i, j]$  چاله باشد.
- اجازه دهد  $i, j$  درست باشد، اگر و فقط اگر در خانه  $[i, j]$  نسیم باشد.
- $\neg P_{1,1}$
- $\neg B_{1,1}$
- $B_{2,1}$
- ” چاله ها باعث وزش نسیم در خانه های مجاور می شوند.”
- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- ” در یک خانه نسیم می ورد اگر و فقط اگر چاله ای مجاور آن باشد“

N. Razavi - AI course - 2005

27

## استفاده از جدول درستی برای استنتاج

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$KB$	$\alpha_1$
false	false	true						
false	false	false	false	false	false	true	false	true
:	:	:	:	:	:	:	:	:
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	true	true
false	true	false	false	false	false	true	true	true
false	true	false	false	false	false	true	true	true
false	true	false	false	true	false	false	false	true
:	:	:	:	:	:	:	:	:
true	false	false						

N. Razavi - AI course - 2005

28

# استنتاج بوسیله شما (ش)

- شمارش تمام مدل ها به روش اول - عمق صحیح و کامل است

**function** TT-ENTAILS?(*KB*,  $\alpha$ ) **returns** true or false

*symbols*  $\leftarrow$  a list of the proposition symbols in *KB* and  $\alpha$

**return** TT-CHECK-ALL(*KB*,  $\alpha$ , *symbols*, [])

**function** TT-CHECK-ALL(*KB*,  $\alpha$ , *symbols*, *model*) **returns** true or false

**if** EMPTY?(*symbols*) **then**

**if** PL-TRUE(*KB*, *model*) **then return** PL-TRUE( $\alpha$ , *model*)

**else return** true

**else do**

*P*  $\leftarrow$  FIRST(*symbols*); *rest*  $\leftarrow$  REST(*symbols*)

**return** TT-CHECK-ALL(*KB*,  $\alpha$ , *rest*, EXTEND(*P*, true, *model*) **and**

TT-CHECK-ALL(*KB*,  $\alpha$ , *rest*, EXTEND(*P*, false, *model*))

برای  $n$  سیمول  $O(2^n)$

N. Razavi - AI course - 2005

29

## اعتبار و صدق پذیری

- یک جمله **معتبر** (*valid*) است اگر در **تمام** مدل ها درست باشد
  - مثال:  $A \Rightarrow A$ ,  $A \vee \neg A$ , True,  $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- ارتباط معتبر بودن با استنتاج:

$KB \models \alpha$  iff ( $KB \Rightarrow \alpha$ ) is **valid**

- یک جمله **صدق پذیر** (*satisfiable*) اگر در **بعضی** از مدل ها درست باشد
  - مثال:  $A \vee B$
- یک جمله **صدق ناپذیر** است اگر در **هیچ** مدلی درست نباشد
  - مثال:  $A \wedge \neg A$
- ارتباط صدق پذیری با استنتاج:

$KB \models \alpha$  iff ( $KB \wedge \neg \alpha$ ) is **unsatisfiable**

N. Razavi - AI course - 2005

## هم ارز منطقی

- دو جمله **هم ارز منطقی** می باشند، اگر و فقط اگر هر دو در مدل های یکسانی درست باشند.

- $\alpha \equiv \beta$  iff  $\alpha \models \beta$  and  $\beta \models \alpha$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$  commutativity of  $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$  commutativity of  $\vee$

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$  associativity of  $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$  associativity of  $\vee$

$\neg(\neg \alpha) \equiv \alpha$  double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$  contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$  implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$  biconditional elimination

$\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$  de Morgan

$\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$  de Morgan

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$  distributivity of  $\wedge$  over  $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$  distributivity of  $\vee$  over  $\wedge$

30

## روش های اثبات

- روش های اثبات به دو نوع تقسیم می شوند:

- **اعمال قوانین استنتاج:**

- تولید صحیح جملات جدید از جملات قدیمی
- **اثبات** = دنباله ای از اعمال قوانین استنتاج می توان از قوانین استنتاج به عنوان عملگرها در الگوریتم استاندارد جستجو استفاده کرد.
- اغلب نیاز به تبدیل جملات به یک **شکل نرم ام** دارند.

- **بررسی مدل:**

- شمارش جدول درستی (بر حسب  $n$  نمایی)
- عقبگرد بهبود یافته (DPLL)
- جستجوی هیوریستیک در فضای مدل (صحیح اما نا کامل)

N. Razavi - AI course - 2005

32

31

## استنتاج (و به جلو و و به عقب

- شکل نرمال (HNF) Horn : عبارت های Horn = ترکیب عطفی عبارت های Horn = عبارت Horn = سیمبول گزاره ای - سیمبول گزاره ای  $\Rightarrow$  (ترکیب عطفی سیمبول های گزاره ای)
- $P_1 \wedge P_2 \wedge \dots \wedge P_n \Rightarrow Q$

مثال : -

- قانون استنتاج Modes Ponens برای شکل Horn

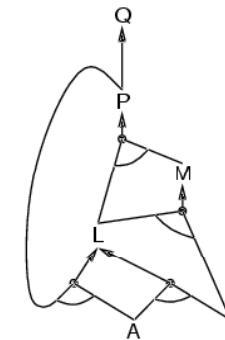
$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n, \quad \alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- می تواند در هر دو روش رویه جلو و رویه عقب بکار رود.
- این روشها بسیار طبیعی هستند و در زمان خطی (بر حسب اندازه KB) اجرا می شوند.

## استنتاج (و به جلو

- ایده : هر قانونی که بخش شرایط آن در KB ارضاء شده را اعمال کن (fire) و نتیجه قانون را به KB اضافه کن، تا زمانیکه پاسخ پیدا شود و یا استنتاج دیگری ممکن نباشد.

$$\begin{aligned} P &\Rightarrow Q \\ L \wedge M &\Rightarrow P \\ B \wedge L &\Rightarrow M \\ A \wedge P &\Rightarrow L \\ A \wedge B &\Rightarrow L \\ A \\ B \end{aligned}$$



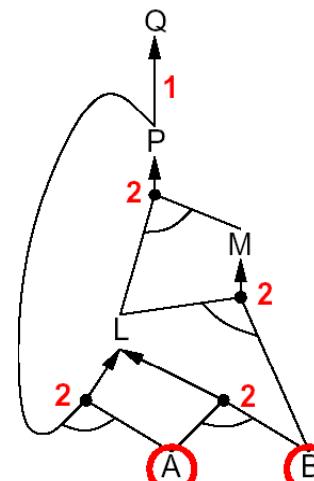
## الgoritم استنتاج (و به جلو

```
function PL-FC-ENTAILS?(KB, q) returns true or false
local variables: count, a table, indexed by clause, initially the number of premises
inferred, a table, indexed by symbol, each entry initially false
agenda, a list of symbols, initially the symbols known to be true

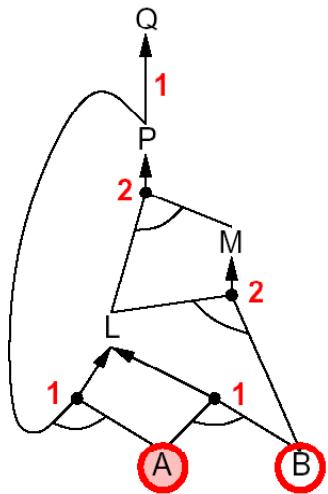
while agenda is not empty do
    p ← POP(agenda)
    unless inferred[p] do
        inferred[p] ← true
        for each Horn clause c in whose premise p appears do
            decrement count[c]
            if count[c] = 0 then do
                if HEAD[c] = q then return true
                PUSH(HEAD[c], agenda)
    return false
```

- استنتاج رویه جلو برای پایگاه دانش در شکل Horn کامل و صحیح است.

## مثال استنتاج (و به جلو



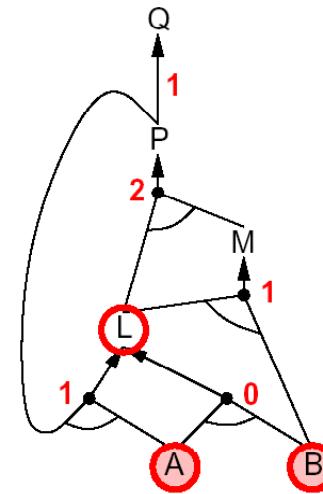
## مثال استنتاج (و به جلو)



N. Razavi - AI course - 2005

37

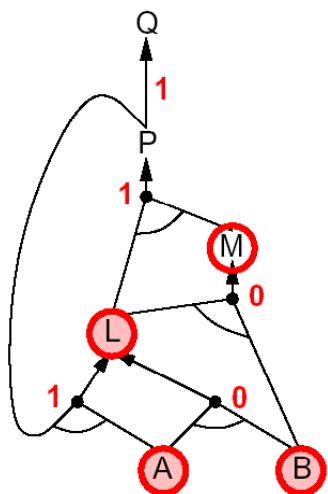
## مثال استنتاج (و به جلو)



N. Razavi - AI course - 2005

38

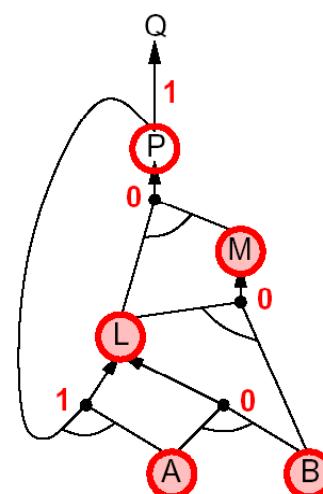
## مثال استنتاج (و به جلو)



N. Razavi - AI course - 2005

39

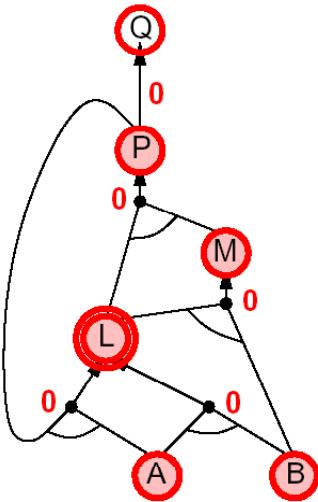
## مثال استنتاج (و به جلو)



N. Razavi - AI course - 2005

40

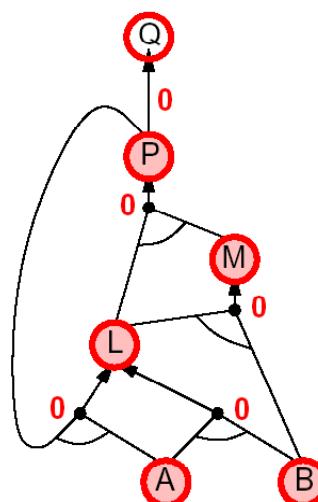
## مثال استنتاج (و به جلو)



N. Razavi - AI course - 2005

41

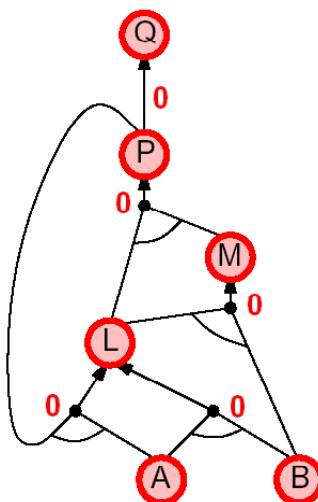
## مثال استنتاج (و به جلو)



N. Razavi - AI course - 2005

42

## مثال استنتاج (و به عقب)



N. Razavi - AI course - 2005

43

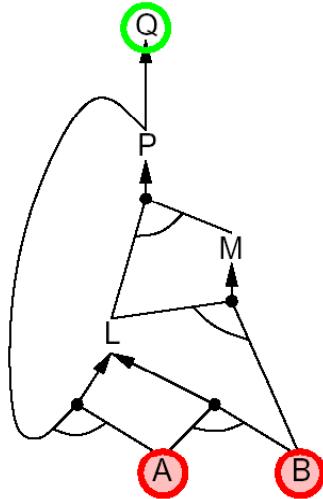
## استنتاج (و به عقب)

- **ایده:** برای اثبات  $q$  به سمت عقب حرکت کن
- برای اثبات  $q$  بوسیله  $BC$
- بررسی کن که آیا  $q$  اکنون ثابت شده می باشد، یا
- بوسیله  $BC$  تمام شرایط برخی از قوانین را که نتیجه آنها  $q$  است اثبات کن
- **اجتناب از حلقه:** بررسی قرار داشتن زیرهدف جدید روی پشته هدف
- **اجتناب از اعمال تکراری:** بررسی اینکه آیا زیرهدف جدید
- (1) قبل درستی اش اثبات شده، یا
- (2) قبل شکست خورده است (fail)

N. Razavi - AI course - 2005

44

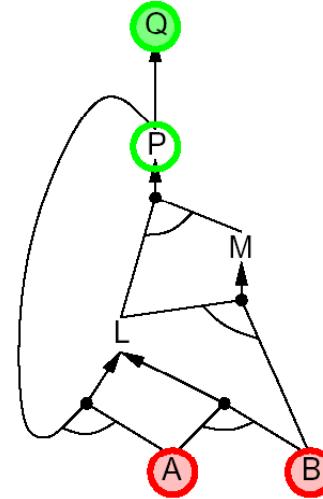
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

45

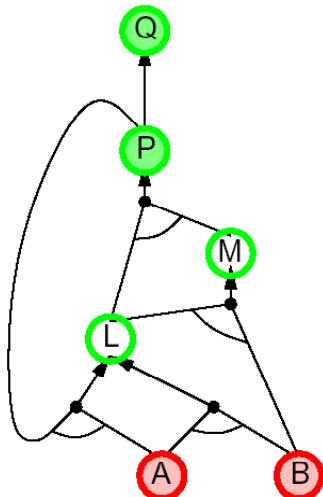
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

46

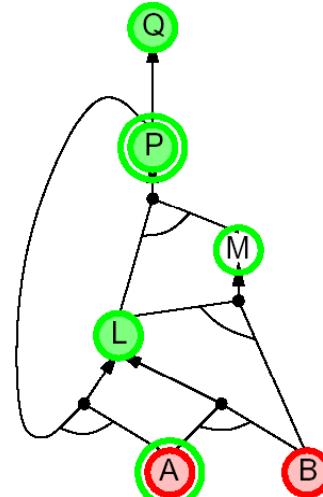
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

47

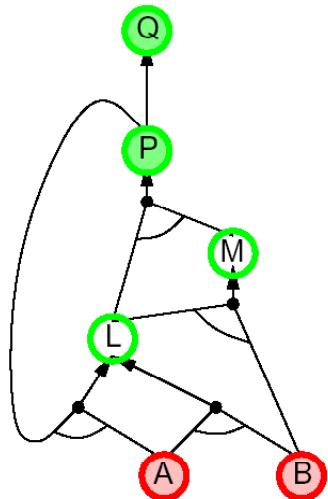
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

48

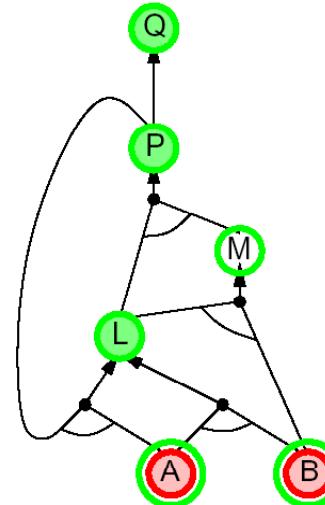
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

49

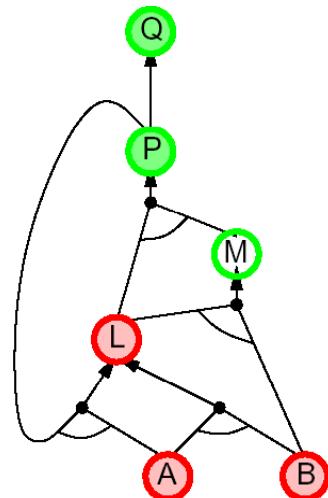
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

50

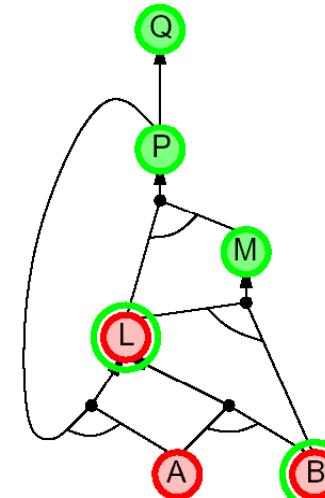
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

51

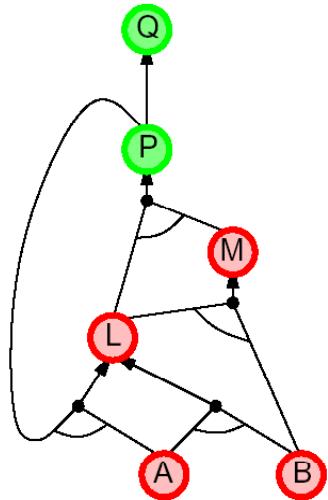
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

52

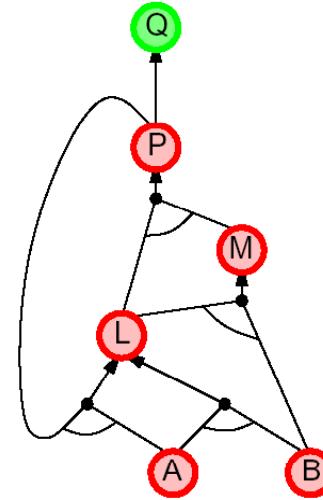
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

53

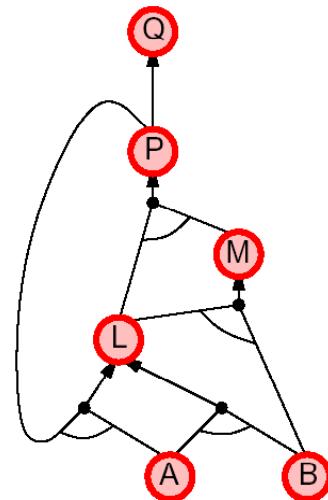
## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

54

## مثال از استنتاج و به عقب



N. Razavi - AI course - 2005

55

## مقایسه دو روش

### • FC

- بر مبنای داده (data driven)
- ممکن است کارهای بسیاری انجام دهد که به هدف مربوط نمی شوند

### • BC

- بر مبنای هدف (goal driven)
- پیچیدگی BC می تواند بسیار بهتر از خطی نسبت به اندازه KB باشد.

N. Razavi - AI course - 2005

56

## Resolution

- شکل نرمال عطفی (CNF)
- conjunctions of disjunctions of literals*
- clauses*

$$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$$

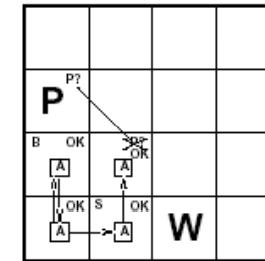
- قانون استنتاج رزولوشن (برای CNF): صحیح و کامل برای منطق گزاره ای

$$\frac{l_1 \vee \dots \vee l_i \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_j \vee \dots \vee m_n}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

$m_j$  و  $l_i$  تقیص یکدیگرند.

N. Razavi - AI course - 2005

57



$$\frac{P_{1,3} \vee P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}$$

- مثال:

- رزولوشن برای منطق گزاره ای صحیح و کامل می باشد.

N. Razavi - AI course - 2005

58

## Resolution

- صحیح قانون استنتاج رزولوشن

$$\neg(l_i \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k) \Rightarrow l_i$$

$$\neg m_j \Rightarrow (m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)$$

$$\neg(l_i \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k) \Rightarrow (m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)$$

N. Razavi - AI course - 2005

59

## CNF به تبدیل

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

1. Eliminate  $\Leftrightarrow$ , replacing  $\alpha \Leftrightarrow \beta$  with  $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ .  
 $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate  $\Rightarrow$ , replacing  $\alpha \Rightarrow \beta$  with  $\neg \alpha \vee \beta$ .  
 $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. Move  $\neg$  inwards using de Morgan's rules and double-negation:  
 $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Apply distributivity law ( $\wedge$  over  $\vee$ ) and flatten:  
 $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

N. Razavi - AI course - 2005

60

## الگوریتم Resolution

- اثبات بواسیله تناقض، یعنی نشان بده  $\alpha \sim \neg\alpha$  صدق پذیر است

```

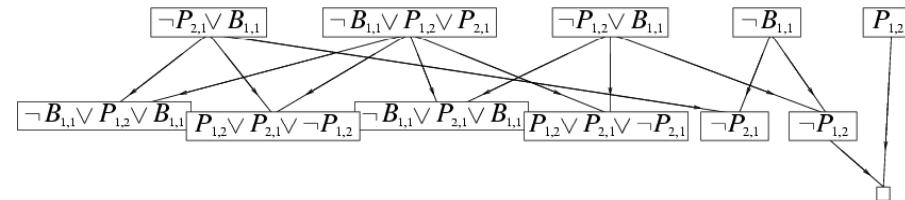
function PL-RESOLUTION( $KB, \alpha$ ) returns true or false
  clauses  $\leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
  new  $\leftarrow \{\}$ 
  loop do
    for each  $C_i, C_j$  in clauses do
      resolvents  $\leftarrow$  PL-RESOLVE( $C_i, C_j$ )
      if resolvents contains the empty clause then return true
      new  $\leftarrow$  new  $\cup$  resolvents
    if new  $\subseteq$  clauses then return false
    clauses  $\leftarrow$  clauses  $\cup$  new
  
```

N. Razavi - AI course - 2005

61

## مثال برای (زولوشن)

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



N. Razavi - AI course - 2005

62

## الگوریتم های استنتاج کارآ در منطق گزاره ای

- دو خانواده از الگوریتم های استنتاج کارآ برای منطق گزاره ای
- الگوریتم های کامل جستجوی عقبگرد:

  - الگوریتم (Davis, Putnam, Logemann, Loveland) DPLL
  - الگوریتم ناکامل جستجوی محلی WalkSAT

N. Razavi - AI course - 2005

63

## DPLL الگوریتم

- تعیین کن که آیا یک جمله ورودی در زبان منطق گزاره ای (در شکل نرمال CNF) صدق پذیر است یا خیر.
- بهبودها نسبت به روش شمارش جدول درستی:

  - خاتمه زود هنگام
    - یک فراکرد (Clause) درست است اگر هر یک از لیترال ها درست باشد.
    - یک جمله نادرست است اگر هر یک از فراکردهای آن نادرست باشد.
  - هیوریستیک سیمبول محض
    - **سیمبول محض:** سیمبولی که در تمام فراکردها با یک علامت ظاهر شود.
    - مثال: در سه فراکرد ( $A \vee \neg B$ ), ( $\neg B \vee \neg C$ ), ( $C \vee A$ ) سیمboleای  $A$  و  $B$  سیمبول محض می باشند، اما  $C$  یک سیمبول محض نیست.
    - لیترال یک سیمبول محض را درست تلقی کن.
  - هیوریستیک فراکرد واحد
    - **فراکرد واحد:** تنها شامل یک لیترال می باشد. یا فراکردی که تمام لیترال های آن غیر از یک لیترال، نادرست می باشند.
    - تنها لیترال موجود در یک فراکرد واحد باید درست باشد.

N. Razavi - AI course - 2005

64

## DPLL الگوریتم

```

function DPLL-SATISFIABLE?(s) returns true or false
  inputs: s, a sentence in propositional logic
  clauses  $\leftarrow$  the set of clauses in the CNF representation of s
  symbols  $\leftarrow$  a list of the proposition symbols in s
  return DPLL(clauses, symbols, [])


---


function DPLL(clauses, symbols, model) returns true or false
  if every clause in clauses is true in model then return true
  if some clause in clauses is false in model then return false
  P, value  $\leftarrow$  FIND-PURE-SYMBOL(symbols, clauses, model)
  if P is non-null then return DPLL(clauses, symbols-P, [P = value|model])
  P, value  $\leftarrow$  FIND-UNIT-CLAUSE(clauses, model)
  if P is non-null then return DPLL(clauses, symbols-P, [P = value|model])
  P  $\leftarrow$  FIRST(symbols); rest  $\leftarrow$  REST(symbols)
  return DPLL(clauses, rest, [P = true|model]) or
         DPLL(clauses, rest, [P = false|model])

```

N. Razavi - AI course - 2005

65

## WalkSAT الگوریتم

- الگوریتم جستجوی محلی و ناکامل
- تابع ارزیابی: هیوریستیک حداقل درگیری برای کمینه کردن تعداد فراکردهای ارضاء نشده
- تعادل میان میزان حریصانه بودن و تصادفی بودن

N. Razavi - AI course - 2005

66

## WalkSAT الگوریتم

```

function WALKSAT(clauses, p, max-flips) returns a satisfying model or failure
  inputs: clauses, a set of clauses in propositional logic
          p, the probability of choosing to do a "random walk" move
          max-flips, number of flips allowed before giving up
  model  $\leftarrow$  a random assignment of true/false to the symbols in clauses
  for i = 1 to max-flips do
    if model satisfies clauses then return model
    clause  $\leftarrow$  a randomly selected clause from clauses that is false in model
    with probability p flip the value in model of a randomly selected symbol
      from clause
    else flip whichever symbol in clause maximizes the number of satisfied clauses
  return failure

```

N. Razavi - AI course - 2005

67

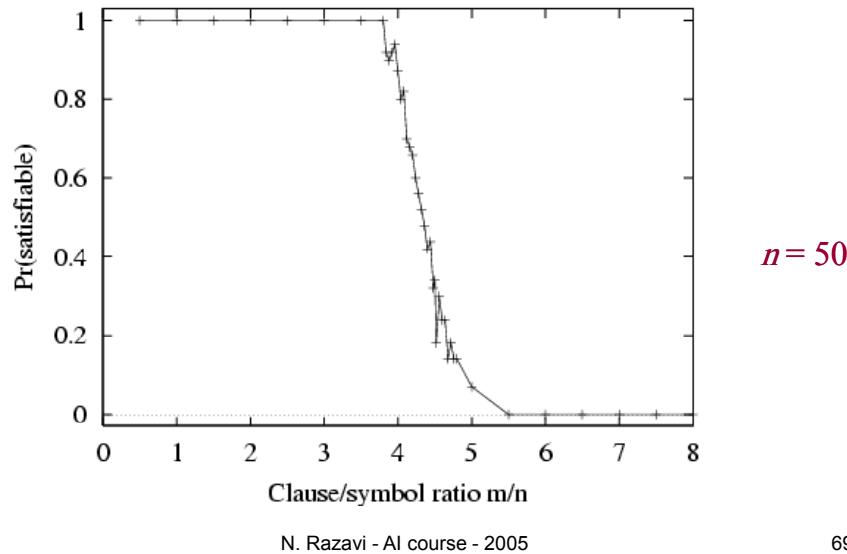
## مسائل ارضاء‌پذیری سخت

- جملات 3-CNF تصادفی را در نظر بگیرید، مثلا:
- $$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$
- ۵ سیمبول و ۵ فراکرد - ۳۲ انتساب و ۱۶ مدل - به طور متوسط دو حدس کافی می باشد (برای یافتن مدل)
- $m =$  تعداد فراکردها
- $n =$  تعداد سیمبول ها
- به نظر می رسد مسائل سخت نزدیک  $m/n = 4.3$  باشند (نسبت بحرانی)

N. Razavi - AI course - 2005

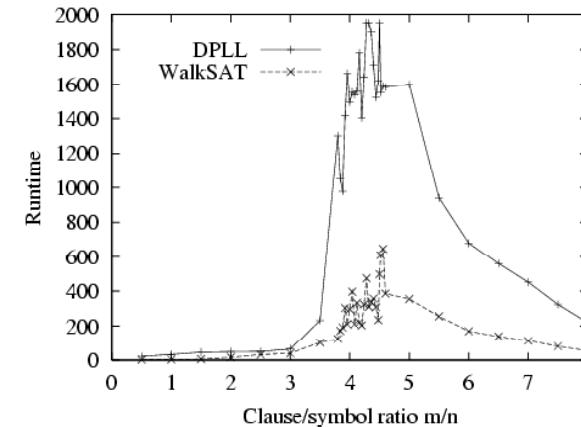
68

## مسائل ارضاء پذیری سخت



69

## مسائل ارضاء پذیری سخت



70

## مسائل ارضاء پذیری سخت

- سه نکته واضح در شکل قبل:

- مسائل نزدیک نقطه بحرانی بسیار سخت از دیگر مسائل تصادفی هستند.

- حتی در مسائل سخت، الگوریتم DPLL نسبتاً کارآمد است - چند هزار مرحله به طور میانگین در مقایسه با  $10^{15} \approx 2^{50}$  برای شمارش جدول درستی.

- در کل محدوده، الگوریتم WalkSAT بسیار سریعتر از DPLL می باشد.

## عامل های مبتنی بر استنتاج در دنیای وامپوس

- یک عامل دنیای وامپوس با استفاده از منطق گزاره ای:

$$\begin{aligned}
 & \neg P_{1,1} \\
 & \neg W_{1,1} \\
 B_{x,y} & \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y}) \\
 S_{x,y} & \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y}) \\
 W_{1,1} & \vee W_{1,2} \vee \dots \vee W_{4,4} \\
 \neg W_{1,1} & \vee \neg W_{1,2} \\
 \neg W_{1,1} & \vee \neg W_{1,3} \\
 & \dots
 \end{aligned}$$

- ۶۴ سیمبول گزاره ای متفاوت
- ۱۵۵ جمله

## محدودیت های منطق گزارهای

- به طور کلی منطق گزاره ای از قدرت بیان کافی برخوردار نیست:

– برای هر مربع شامل جملات "فیزیکی" آن مربع

– برای هر زمان  $t$  و هر مکان  $[x, y]$

$$L_{x,y}^t \wedge FacingRight^t \wedge Forward^t \Rightarrow L_{x+1,y}^{t+1}$$

```

function PL-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench,breeze,glitter]
  static: KB, initially containing the "physics" of the wumpus world
    x, y, orientation, the agent's position (init. [1,1]) and orient. (init. right)
    visited, an array indicating which squares have been visited, initially false
    action, the agent's most recent action, initially null
    plan, an action sequence, initially empty

  update x,y,orientation, visited based on action
  if stench then TELL(KB, Sx,y) else TELL(KB,  $\neg S_{x,y}$ )
  if breeze then TELL(KB, Bx,y) else TELL(KB,  $\neg B_{x,y}$ )
  if glitter then action  $\leftarrow$  grab
  else if plan is nonempty then action  $\leftarrow$  POP(plan)
  else if for some fringe square  $[i,j]$ , ASK(KB,  $(\neg P_{i,j} \wedge \neg W_{i,j})$ ) is true or
    for some fringe square  $[i,j]$ , ASK(KB,  $(P_{i,j} \vee W_{i,j})$ ) is false then do
      plan  $\leftarrow$  A*-GRAPH-SEARCH(ROUTE-PB([x,y], orientation, [i,j], visited))
      action  $\leftarrow$  POP(plan)
  else action  $\leftarrow$  a randomly chosen move
  return action

```

N. Razavi - AI course - 2005

73

N. Razavi - AI course - 2005

74

## خلاصه

- عامل های منطقی از استنتاج بر روی یک پایگاه دانش برای استتفاق دانش جدید و تصمیم گیری استفاده می کنند

• مفاهیم مبنایی منطق:

– ساختار (syntax)

– معنا (semantics)

– استلزم (entailments)

– استنتاج (inference)

– صحت (soundness)

– کامل بودن (completeness)

- استنتاج رو به جلو و رو به عقب برای عبارت های Horn دارای زمان خطی هستند و کامل می باشند.

• رزولوشن برای منطق گزاره ای کامل است

• منطق گزاره ای در بازنمایی دانش ضعیف می باشد

N. Razavi - AI course - 2005

75

## مقدمه

# منطق مرتبه اول

فصل هشتم

سید ناصر رضوی

E-mail: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۴

N. Razavi- AI course - 2005

1

- چرا FOL
- بررسی ساختار و معانی جملات در FOL
- استفاده از FOL
- دنیای وامپوس در FOL
- مهندسی دانش در FOL

N. Razavi- AI course - 2005

2

## مزایا و معایب منطق گزاره ای

- (+) منطق گزاره ای **توصیفی** است --- دانش و استنتاج مستقل مرتبه اول (مانند زبان طبیعی) فرض می کند دنیا از **حقایق** تشکیل شده است، منطق
- (+) منطق گزاره ای **ترکیی** است --- در غیر اینصورت کار سیستم استدلال دشوار است یعنی، معنای یک جمله مرکب با توجه به جملات سازنده آن تعیین می شود
- (+) معنای جملات در منطق گزاره ای **مستقل از متن** می باشد (برخلاف زبان طبیعی)

- (-) منطق گزاره ای قدرت بیان بسیار محدودی دارد (برخلاف زبان طبیعی)  
مثال، در منطق گزاره ای نمی توان گفت " چاله ها باعث وزش نسیم در خانه های مجاور می شوند"  
- مگر آنکه برای هر خانه یک جمله نوشته شود.

N. Razavi- AI course - 2005

3

## FOL

- در حالیکه منطق گزاره ای فرض می کند دنیا از **حقایق** تشکیل شده است، منطق
- اشیاء (**Objects**): دنیا از اشیایی تشکیل شده که بواسطه خصوصیاتان (Properties) از یکدیگر قابل تمایز می باشند
- روابط (**Relations**): در بین اشیاء روابط مختلفی می تواند وجود داشته باشد که بعضی از این روابط به صورت تابع (Function) می باشند.
- توابع (**functions**): پدر، برادر، یکی بیشتر از و ...

- حقایق به صورت ارجاع به اشیاء، خصوصیات و روابط بین اشیاء بیان می شوند.
- مثال :

- Sum(1, 2, 3), Even(2), Odd(3), ...
- Parent (Bob, Jim), Male(Bob), ...
- Add(1, 2), LeftLegof(John), ...

N. Razavi- AI course - 2005

4

# منطق ها به طور کلی

Language	Ontology	Epistemology
Propositional Logic	facts	true/false/unknown
First Order Logic	facts, objects, relations	true/false/unknown
Temporal Logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	degree of truth $\in [0, 1]$	known interval value

N. Razavi- AI course - 2005

5

# ساختار *FOL*: عناصر ابتدایی

- ثابت ها (King, John, 2, ...)
- مسندها (Brother, >, ...)
- توابع (Sqrt, LeftLegOf, ...)
- متغیرها (x, y, a, b, ...)
- رابطهای منطقی ( $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$ )
- تساوي (=)
- سورها ( $\forall, \exists$ )

N. Razavi- AI course - 2005

6

# ساختار جملات در *FOL*

Sentence  $\rightarrow$  AtomicSentence

- | Sentence Connective Sentence
- | Quantifier Variable, ... Sentence
- |  $\sim$  Sentence
- | ( Sentence )

AtomicSentence  $\rightarrow$  Predicate(Term, ...) | Term = Term

Term  $\rightarrow$  Function(Term, ...)

- | Constant
- | Variable

N. Razavi- AI course - 2005

7

# ساختار جملات در *FOL*

Connective  $\rightarrow$   $\Rightarrow$  |  $\wedge$  |  $\vee$  |  $\Leftrightarrow$

Quantifier  $\rightarrow$   $\forall$  |  $\exists$

Constant  $\rightarrow$  A | X<sub>1</sub> | John | ...

Variable  $\rightarrow$  a | x | ...

Predicate  $\rightarrow$  Before | HasColor | ...

Function  $\rightarrow$  MotherOf | LeftLegOf

N. Razavi- AI course - 2005

8

## ثابت ها، مسندها و توابع

- **سیمبولهای ثابت**: به یک شی خاص در مدل رجوع می کنند.  
– A, B, John, ...
- **مسندها** (Predicates): به یک رابطه ویژه در مدل رجوع می کنند.  
– Brother, Mother, ...
- **توابع**: بعضی از روابط تابع هستند، یعنی هر شی تو سط رابطه دقیقاً به یک شیء دیگر رجوع می کند.  
– MotherOf, Cos, LeftLegOf, ...

N. Razavi- AI course - 2005

9

## Term (Term)

- **Term**: یک عبارت منطقی که به یک شیء رجوع می کند.
  - ثابت ها (A, B, C, John, ...)
  - متغیرها (a, b, x, ...)
  - توابع

MotherOf(Richard),  
Length(LeftLegOf(John)),  
Cos(30), ...

N. Razavi- AI course - 2005

10

## جملات ساده (Atomic Sentences)

- یک جمله اتمی به صورت زیر می باشد:

### Predicate(Term<sub>1</sub>, ..., Term<sub>n</sub>)

- Brother(KingJohn, RichardTheLionheart)
- >(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))
- Married(FatherOf(Richard), MotherOF(John))

- یک جمله ساده زمانی درست است که رابطه Predicate بین اشیایی که Term<sub>1</sub> و ... و Term<sub>n</sub> به آنها اشاره می کنند برقرار باشد.

N. Razavi- AI course - 2005

11

## جملات مرکب (Complex Sentences)

- جملات مرکب از ترکیب جملات ساده بوسیله رابطه های منطقی بدست می آیند.

$\neg S_1, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$

- Brother(John, Richard)  $\wedge$  Brother(Richard, John)
- Older(John, 30)  $\vee$  Younger(John, 30)
- Older(John, 30)  $\Rightarrow$  ~Younger(John, 30)
- ~Brother(Robin, John)

N. Razavi- AI course - 2005

12

## درستی در منطق مرتبه اول

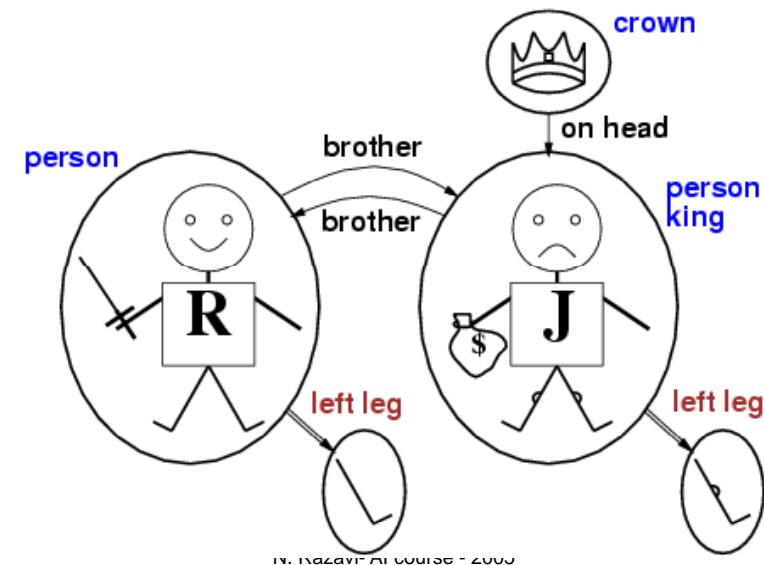
- جملات نسبت به یک **مدل** و یک **تفسیر** درست می باشند.
- مدل ها شامل اشیا (**عناصر دامنه**) و روابط میان آنها می باشند.
- تفسیر، مورد مراجعه را برای موارد زیر مشخص می کند:

اشیاء	$\leftarrow$	سیمبول های ثابت
روابط	$\leftarrow$	سیمبول های مستندی
روابط تابعی	$\leftarrow$	سیمبول های تابع

N. Razavi- AI course - 2005

13

## مدل ها در منطق مرتبه اول: مثال



14

## سرو(عمومی)

$$\forall \langle \text{Variables} \rangle \langle \text{Sentence} \rangle$$

Everyone at the class is smart

$$\forall x \text{at}(x, \text{Class}) \Rightarrow \text{Smart}(x)$$

- سور عمومی برابر است با توکیب عطفی تمام جملاتی که با جایگذاری متغیرها در Sentence بدست می آیند.

$$\text{at}(\text{KingJohn}, \text{Class}) \Rightarrow \text{Smart}(\text{KingJohn})$$

$$\wedge \text{at}(\text{Richard}, \text{Class}) \Rightarrow \text{Smart}(\text{Richard})$$

$$\wedge \text{at}(\text{Class}, \text{Class}) \Rightarrow \text{Smart}(\text{Class})$$

$$\wedge \dots$$

N. Razavi- AI course - 2005

15

## سرو(وجودی)

$$\exists \langle \text{Variables} \rangle \langle \text{Sentence} \rangle$$

Someone at class is smart

$$\exists x \text{at}(x, \text{class}) \wedge \text{Smart}(x)$$

- سور وجودی برابر است با توکیب فصلی تمام جملاتی که با جایگذاری متغیرها در Sentence بدست می آیند.

$$\text{at}(\text{KingJohn}, \text{Class}) \wedge \text{Smart}(\text{KingJohn})$$

$$\vee \text{at}(\text{Richard}, \text{Class}) \wedge \text{Smart}(\text{Richard})$$

$$\vee \text{at}(\text{Class}, \text{Class}) \wedge \text{Smart}(\text{Class})$$

$$\vee \dots$$

N. Razavi- AI course - 2005

16

## یک اشتباه متداول

- سور عمومی اغلب با ترکیب شرطی و سور وجودی اغلب با ترکیب عطفی  
بکار می رود

$$\forall x \text{Human}(x) \Rightarrow \text{Mortal}(x)$$

$$\exists x \text{Sister}(x, \text{Spot}) \wedge \text{Cat}(x)$$

- مثال: جمله زیر به معنای "هر کسی در کلاس است و هر کسی باهوش است" می باشد

$$\forall x \text{at}(x, \text{Class}) \wedge \text{Smart}(x)$$

و یا جمله زیر:

$$\exists x \text{at}(x, \text{Class}) \Rightarrow \text{Smart}(x)$$

N. Razavi- AI course - 2005

17

## سورهای تودرتو (Nested Quantifiers)

$$\forall x \forall y \text{Parent}(x, y) \Rightarrow \text{Child}(y, x)$$

$$\exists x \exists y \text{Owner}(x, y)$$

- در جملات بالا می توان جای سورها را با هم تعویض نمود  
اما  $\exists x \forall y \exists x \forall y$  معادل  $\exists x \forall y \exists x \forall y$  نمی باشد.

$$\forall x \exists y \text{mother}(y, x)$$

$$\exists y \forall x \text{mother}(y, x)$$

N. Razavi- AI course - 2005

18

## ارتباط بین سورها

- سور عمومی و وجودی از طریق تناقض با یکدیگر در ارتباط هستند:

$$\neg(\forall x P) \equiv \exists x \neg P$$

$$\neg(\exists x P) \equiv \forall x \neg P$$

$$\forall x P \equiv \neg \exists x \neg P$$

$$\exists x P \equiv \neg \forall x \neg P$$

$$\forall x \text{Likes}(x, \text{IceCream})$$

$$\neg \exists x \neg \text{Likes}(x, \text{IceCream})$$

$$\exists x \text{Likes}(x, \text{Broccoli})$$

$$\neg \forall x \neg \text{Likes}(x, \text{Broccoli})$$

- مثال: دو گانی سور

N. Razavi- AI course - 2005

19

## تساوی

- $Term_1 = Term_2$  تحت یک تفسیر مشخص درست است اگر و فقط اگر  $Term_1$  و  $Term_2$  هر دو به شیء یکسانی مراجعه کنند.

- مثال: تعریف  $Sibling$  بر حسب  $Parent$

$$\forall x, y \text{Sibling}(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg(m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$

N. Razavi- AI course - 2005

20

## استفاده از FOL

دامنه مجموعه:

- $\forall s \text{ Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x|s_2\})$
- 
- $\neg \exists x, s \{x|s\} = \{\}$
- 
- $\forall x, s \ x \in s \Leftrightarrow s = \{x|s\}$
- 
- $\forall x, s \ x \in s \Leftrightarrow [\exists y, s_2 (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
- 
- $\forall s_1, s_2 \ s_1 \subseteq s_2 \Leftrightarrow (\forall x \ x \in s_1 \Rightarrow x \in s_2)$   
N. Razavi- AI course - 2005
- $\forall s_1, s_2 (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$

22

## استفاده از FOL

دامنه روابط خانوادگی:

- برادرها با یکدیگر Sibling هستند

$$\forall x, y \ Brother(x, y) \Rightarrow Sibling(x, y)$$

- مادر هر شخص والد مونث آن شخص می باشد

$$\forall m, c \ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m, c))$$

- رابطه "Sibling" دارای خاصیت تقارنی می باشد

$$\forall x, y \ Sibling(x, y) \Leftrightarrow Sibling(y, x)$$

N. Razavi- AI course - 2005

21

## محاوره با پایگاه دانش در FOL

- فرض کنید عاملی در دنیای وامپوس از پایگاه دانش FOL استفاده می کند و در زمان  $t = 5$  می باشد؟

Tell(KB, Percept([Smell, Breeze, None], 5))

Ask(KB,  $\exists a \text{ BestAction}(a, 5)$ )

- یعنی، آیا KB مستلزم بهترین عمل در  $t = 5$  می باشد؟
- پاسخ: بله ،  $\{a/\text{shoot}\} \rightarrow$  لیست جانشینی
- با داشتن جمله  $S$  و لیست جانشینی  $\sigma$  :
- $S\sigma$  به نتیجه حاصل از جایگذاری  $\sigma$  در  $S$  اشاره دارد

$S = \text{Smarter}(x, y)$

$\sigma = \{x/\text{Hillary}, y/\text{Bill}\}$

$S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})$

$$KB \models \sigma \text{ همه یا برخی از } \sigma \text{ ها را بر می گرداند به طوری که } ASK(KB, \sigma) -$$

23

## پایگاه دانش برای دنیای وامپوس

- ادراک

$$- \forall t, s, b \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{Glitter}(t)$$

–

- واکنش

$$- \forall t \ \text{Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab}, t)$$

## استنتاج خواص پنهانی

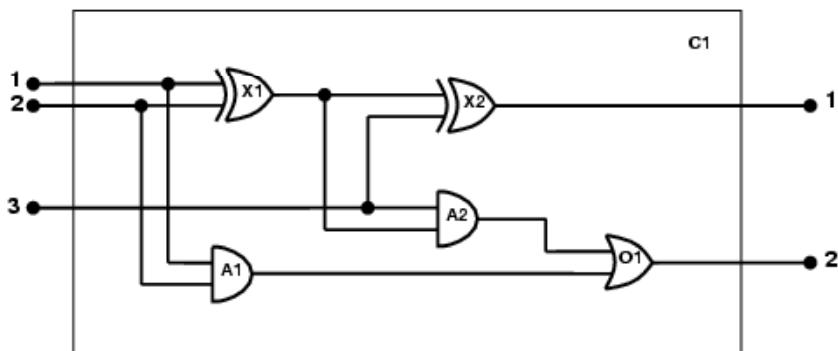
- $\forall x,y,a,b \text{ Adjacent}([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y], [x-1,y], [x,y+1], [x,y-1]\}$
- خصوصیات مکان ها:
- $\forall s,t At(\text{Agent},s,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$
- خانه های مجاور با چاه ها دارای نسیم می باشند:
  - قانون تشخیصی --- علت را از روی اثر آن نتیجه می گیرد
$$\forall s \text{ Breezy}(s) \Rightarrow \exists r \text{ Adjacent}(r,s) \wedge \text{Pit}(r)$$

N. Razavi- AI course - 2005

25

## دامنه مدارهای الکتریکی

- جمع کننده یک بیتی



N. Razavi- AI course - 2005

27

## مهندسى دانش در FOL

- مشخص نمودن وظیفه
- جمع آوری دانش مربوطه
- تصمیم گیری در مورد مسندها، توابع و ثابت ها
- کد نمودن دانش عمومی دامنه
- کد نمودن توصیف نمونه مساله خاص
- اعمال پرس و جو به رویه استنتاج و در یافت پاسخ
- اشکال زدایی پایگاه دانش

N. Razavi- AI course - 2005

26

## دامنه مدارهای الکتریکی

- مشخص نمودن وظیفه
  - آیا مدار واقعا به درستی جمع می کند؟ (وارسی مدار)
- جمع آوری دانش مربوطه
  - ترکیب سیم ها و گیت ها؛ انواع گیت ها (AND, OR, XOR, NOT)
  - دانش نامربوط: اندازه، شکل، رنگ، قیمت گیت ها
- تصمیم گیری در مورد لغات
  - راه های مختلف:

Type(X<sub>1</sub>) = XOR

Type(X<sub>1</sub>, XOR)  
XOR(X<sub>1</sub>)

N. Razavi- AI course - 2005

28

## دامنه مدارهای الکتریکی

۴. کد نمودن دانش عمومی دامنه

$$\neg \forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$$

$$\neg \forall t \text{ Signal}(t) = 1 \vee \text{Signal}(t) = 0$$

$$-1 \neq 0$$

$$\neg \forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$$

$$\begin{aligned} \neg \forall g \text{ Type}(g) = \text{OR} &\Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \\ \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 1 & \end{aligned}$$

$$\begin{aligned} \neg \forall g \text{ Type}(g) = \text{AND} &\Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \\ \Leftrightarrow \exists n \text{ Signal}(\text{In}(n,g)) = 0 & \end{aligned}$$

$$\begin{aligned} \neg \forall g \text{ Type}(g) = \text{N. Razavi- Al course - 2005 XOR} &\Rightarrow \text{Signal}(\text{Out}(1,g)) = 2^b \\ \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g)) & \end{aligned}$$

## دامنه مدارهای الکتریکی

۶. اعمال پرس و جو بر رویه استنتاج

چه ترکیبی از ورودی ها باعث می شوک که اولین خروجی مدار  $C_1$  (بیت جمع) به صفر و خروجی دوم مدار  $C_1$  (بیت نقلی) به یک تبدیل شود؟

$$\exists i_1, i_2, i_3 \text{ Signal}(\text{In}(1,C_1)) = i_1 \wedge \text{Signal}(\text{In}(2,C_1)) = i_2 \wedge \text{Signal}(\text{In}(3,C_1)) = i_3 \wedge \text{Signal}(\text{Out}(1,C_1)) = 0 \wedge \text{Signal}(\text{Out}(2,C_1)) = 1$$

پاسخ های ممکن:

$$\{i_1/1, i_2/1, i_3/0\},$$

$$\{i_1/1, i_2/0, i_3/1\},$$

$$\{i_1/0, i_2/1, i_3/1\}$$

## دامنه مدارهای الکتریکی

۵. کد نمودن دانش مربوط به یک نمونه مساله خاص

$$\text{Type}(X_1) = \text{XOR}$$

$$\text{Type}(X_2) = \text{XOR}$$

$$\text{Type}(A_1) = \text{AND}$$

$$\text{Type}(A_2) = \text{AND}$$

$$\text{Type}(O_1) = \text{OR}$$

$$\text{Connected}(\text{Out}(1,X_1), \text{In}(1,X_2))$$

$$\text{Connected}(\text{In}(1,C_1), \text{In}(1,X_1))$$

$$\text{Connected}(\text{Out}(1,X_1), \text{In}(2,A_2))$$

$$\text{Connected}(\text{In}(1,C_1), \text{In}(1,A_1))$$

$$\text{Connected}(\text{Out}(1,A_2), \text{In}(1,O_1))$$

$$\text{Connected}(\text{In}(2,C_1), \text{In}(2,X_1))$$

$$\text{Connected}(\text{Out}(1,A_1), \text{In}(2,O_1))$$

$$\text{Connected}(\text{In}(2,C_1), \text{In}(2,A_1))$$

$$\text{Connected}(\text{Out}(1,X_2), \text{Out}(1,C_1))$$

$$\text{Connected}(\text{In}(3,C_1), \text{In}(2,X_2))$$

$$\text{Connected}(\text{Out}(1,O_1), \text{Out}(2,C_1))$$

$$\text{Connected}(\text{In}(3,C_1), \text{In}(1,A_2))$$

N. Razavi- Al course - 2005

30

## دامنه مدارهای الکتریکی

۶. اعمال پرس و جو بر رویه استنتاج

$$\begin{aligned} \exists i_1, i_2, i_3, o_1, o_2 \text{ Signal}(\text{In}(1,C_1)) = i_1 \wedge \text{Signal}(\text{In}(2,C_1)) = i_2 \wedge \\ \text{Signal}(\text{In}(3,C_1)) = i_3 \wedge \text{Signal}(\text{Out}(1,C_1)) = o_1 \wedge \text{Signal}(\text{Out}(2,C_1)) = o_2 \end{aligned}$$

**پاسخ:** جدول ورودی/خروجی کامل که می تواند برای بررسی مدار استفاده شود.

## دامنه مدارهای الکتریکی

### ۷. اشکال زدایی پایگاه دانش

اگر حقیقت  $0 \neq 1$  را در پایگاه دانش نداشته باشیم:

$$\exists i_1, i_2, o \quad Signal(In(1, C_1)) = i_1 \wedge Signal(In(2, C_1)) = i_2 \wedge \\ Signal(Out(1, X_1)) = o$$

این پرسش مشخص می کند که برای ورودی های ۱۰ و ۱۱ هیچ خروجی ای در  $X_1$  مشخص نمی باشد. با توجه به اصل موضوعی در مورد XOR

$$\forall g \text{ Type}(g) = \text{XOR} \Rightarrow Signal(Out(1, g)) = 1 \Leftrightarrow Signal(In(1, g)) \neq Signal(In(2, g))$$

اگر ورودی ها صفر و یک بشوند:

$$Signal(Out(1, X_1)) = 1$$

$$\Leftrightarrow 1 \neq 0$$

## خلاصه

- منطق مرتبه اول

- اشیاء و روابط عناصر اولیه معنایی هستند

- ساختار: ثابت ها، توابع، مستندها، تساوی، سورها

- قدرت بیان بیشتر: کافی برای تعریف دنیای وامپوس

## مقدمه

- تقلیل استنتاج مرتبه-اول به استنتاج گزاره ای
- یکسان سازی (Unification)
- MP تعمیم یافته (GMP)
- زنجیره استنتاج روبه جلو
- زنجیره استنتاج رو به عقب
- رزولوشن

## استنتاج در منطق مرتبه اول

فصل نهم

سید ناصر رضوی

E-mail: [razavi@Comp.iust.ac.ir](mailto:razavi@Comp.iust.ac.ir)

۱۳۸۴

N. Razavi- AI course - 2006

2

## نمونه سازی عمومی (UI)

- هر نمونه از یک جمله دارای سور عمومی، توسط آن جمله قابل استلزم است.

$$\frac{\forall v \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

- مثال:  $\forall x \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
- $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
- $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
- $\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$

## نمونه سازی وجودی (EI)

- برای هر جمله  $\alpha$ ، متغیر  $V$  و سیمبول ثابت  $k$  که در جای دیگری از پایگاه دانش ظاهر نشده باشد:

$$\frac{\exists v \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- مثال:  $\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$
- $\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$

به شرطی که  $C_1$  یک سیمبول جدید باشد، که به آن ثابت اسکولم می گویند.

N. Razavi- AI course - 2006

4

N. Razavi- AI course - 2006

3

## تقلیل به استنتاج گزاره ای

- فرض کنید که KB فقط شامل جملات زیر باشد:  
 $\forall x \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$   
 King(John)  
 Greedy(John)  
 Brother(Richard, John)
- با نمونه سازی جمله دارای سور عمومی به تمام طرق ممکن، داریم:  
 $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$   
 $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$   
 King(John)  
 Greedy(John)  
 Brother(Richard, John)
- KB جدید گزاره ای سازی شده است. سیمبول های گزاره ای عبارتند از:  
 King(John), Greedy(John), Evil(John), King(Richard), etc

N. Razavi- AI course - 2006

5

## تقلیل به استنتاج گزاره ای (ادامه)

- هر پایگاه دانش در FOL می تواند به گونه ای گزاره ای سازی شود که استلزم را حفظ کند
- یک جمله توسط KB جدید قابل استلزم است اگر و فقط اگر توسط KB اصلی قابل استلزم باشد
- ایده: پایگاه دانش و کوئری را گزاره ای سازی کن، رزولوشن را اعمال و نتیجه را برگردان
- مشکل: در مورد سیمبول های تابعی تعداد نامحدودی ترم زمینی وجود دارد، مثلا:

*Father(Father(Father(John)))*

N. Razavi- AI course - 2006

6

## تقلیل به استنتاج گزاره ای (ادامه)

- تئوری Herbrand (۱۹۳۰): اگر جمله  $\alpha$  توسط پایگاه دانش FOL قابل استلزم باشد، آنگاه این جمله توسط زیرمجموعه محدودی از پایگاه دانش گزاره ای سازی شده قابل استلزم است
- ایده:

For  $n = 0$  to  $\infty$  do

create a propositional KB by instantiating with depth-\$n\$ terms  
see if  $\alpha$  is entailed by this KB

- مشکل: اگر KB مستلزم  $\alpha$  باشد کار می کند، در غیر اینصورت در حلقه بی نهایت می افتد

- تئوری تورینگ و چرج (۱۹۳۶): استلزم برای FOL نیمه تصمیم پذیر است
- الگوریتم هایی وجود دارند که برای هر جمله قابل استلزم پاسخ بله را تولید می کنند، ولی الگوریتمی وجود ندارد که برای هر جمله ای که قابل استلزم نباشد پاسخ خیر تولید کند.

N. Razavi- AI course - 2006

7

## مشکلات گزاره ای سازی نمودن

- به نظر می رسد که گزاره ای سازی کردن جملات نامربوط زیادی تولید می کند. برای مثال از جملات زیر:

$\forall x \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$   
 King(John)  
 $\forall y \text{Greedy}(y)$   
 Brother(Richard, John)

- بدیهی است که Evil(John). اما گزاره ای سازی کردن حقایق زیادی مانند Greedy(Richard) تولید می کند که نامربوط می باشند

N. Razavi- AI course - 2006

8

## یکسان سازی (Unification)

- اگر بتوانیم یک جانشینی مانند  $\theta$  بیاییم به طوری که  $(x)$  و  $King(John)$  و  $Greedy(x)$  تطیق یابند، آنگاه استنتاج بلافصله بدست می‌آید.  
 $\theta = \{x/John, y/John\}$

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

p	q	$\theta$
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

N. Razavi- AI course - 2006

9

## یکسان سازی (Unification)

- اگر بتوانیم یک جانشینی مانند  $\theta$  بیاییم به طوری که  $(x)$  و  $King(John)$  و  $Greedy(x)$  تطیق یابند، آنگاه استنتاج بلافصله بدست می‌آید.  
 $\theta = \{x/John, y/John\}$

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

p	q	$\theta$
Knows(John,x)	Knows(John,Jane)	$\{x/Jane\}$
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

N. Razavi- AI course - 2006

10

## یکسان سازی (Unification)

- اگر بتوانیم یک جانشینی مانند  $\theta$  بیاییم به طوری که  $(x)$  و  $King(John)$  و  $Greedy(x)$  تطیق یابند، آنگاه استنتاج بلافصله بدست می‌آید.  
 $\theta = \{x/John, y/John\}$

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

p	q	$\theta$
Knows(John,x)	Knows(John,Jane)	$\{x/Jane\}$
Knows(John,x)	Knows(y,OJ)	$\{x/OJ, y/John\}$
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

N. Razavi- AI course - 2006

11

## یکسان سازی (Unification)

- اگر بتوانیم یک جانشینی مانند  $\theta$  بیاییم به طوری که  $(x)$  و  $King(John)$  و  $Greedy(x)$  تطیق یابند، آنگاه استنتاج بلافصله بدست می‌آید.  
 $\theta = \{x/John, y/John\}$

- $Unify(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

p	q	$\theta$
Knows(John,x)	Knows(John,Jane)	$\{x/Jane\}$
Knows(John,x)	Knows(y,OJ)	$\{x/OJ, y/John\}$
Knows(John,x)	Knows(y,Mother(y))	$\{y/John, x/Mother(John)\}$
Knows(John,x)	Knows(x,OJ)	

N. Razavi- AI course - 2006

12

# یکسان سازی (Unification)

- اگر بتوانیم یک جانشینی مانند  $\theta$  بیاییم به طوری که  $\text{King}(x) \theta \text{Greedy}(x)$  و  $\text{King}(y) \theta \text{Greedy}(y)$  تطیق یابند، آنگاه استنتاج بلا فاصله بدست می آید.  
 $\theta = \{x/\text{John}, y/\text{John}\}$

- $\text{Unify}(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

p	q	$\theta$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(\text{John}, \text{Jane})$	$\{x/\text{Jane}\}$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(y, \text{OJ})$	$\{x/\text{OJ}, y/\text{John}\}$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(y, \text{Mother}(y))$	$\{y/\text{John}, x/\text{Mother}(\text{John})\}$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(x, \text{OJ})$	$\{\text{fail}\}$
استاند ردد سازی متغیرها: مثلاً N. Razavi- AI course - 2006		

13

# یکسان سازی

- برای یکسان سازی سازی  $\text{Knows}(y, z)$  و  $\text{Knows}(\text{John}, x)$  دو جانشینی وجود دارد یا

- $\theta = \{y/\text{John}, x/z\}$
- $\theta = \{y/\text{John}, x/\text{John}, z/\text{John}\}$

- اولین یکسان ساز عمومی تر از دومی می باشد
- فقط یک عمومی ترین یکسان ساز (MGU) وجود دارد که منحصر به فرد می باشد.

$$\text{MGU} = \{y/\text{John}, x/z\}$$

N. Razavi- AI course - 2006

14

## الگوریتم یکسان سازی

```

function UNIFY( $x, y, \theta$ ) returns a substitution to make  $x$  and  $y$  identical
  inputs:  $x$ , a variable, constant, list, or compound
           $y$ , a variable, constant, list, or compound
           $\theta$ , the substitution built up so far

  if  $\theta = \text{failure}$  then return failure
  else if  $x = y$  then return  $\theta$ 
  else if VARIABLE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )
  else if VARIABLE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )
  else if COMPOUND?( $x$ ) and COMPOUND?( $y$ ) then
    return UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))
  else if LIST?( $x$ ) and LIST?( $y$ ) then
    return UNIFY(REST[ $x$ ], REST[ $y$ ], UNIFY(FIRST[ $x$ ], FIRST[ $y$ ],  $\theta$ ))
  else return failure
  
```

N. Razavi- AI course - 2006

15

## الگوریتم یکسان سازی

```

function UNIFY-VAR( $var, x, \theta$ ) returns a substitution
  inputs:  $var$ , a variable
           $x$ , any expression
           $\theta$ , the substitution built up so far

  if  $\{var/val\} \in \theta$  then return UNIFY( $val, x, \theta$ )
  else if  $\{x/val\} \in \theta$  then return UNIFY( $var, val, \theta$ )
  else if OCCUR-CHECK?( $var, x$ ) then return failure
  else return add  $\{var/x\}$  to  $\theta$ 
  
```

N. Razavi- AI course - 2006

16

# Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\theta} \quad \text{where } p_i'\theta = p_i \theta \text{ for all } i$$

$p_1$ is <i>King</i> (John)	$p_1$ is <i>King</i> (x)
$p_2$ is <i>Greedy</i> (y)	$p_2$ is <i>Greedy</i> (x)
$\theta$ is {x/John,y/John}	q is <i>Evil</i> (x)
q θ is <i>Evil</i> (John)	

- GMP با پایگاه دانشی از فراکردهای معین (دقیقاً یک لیترال مثبت) کار می کند
  - فرض می شود که تمام متغیرها دارای سور عمومی هستند.

N. Razavi- AI course - 2006

17

GMP ملک

بايد نشان دهيم:

$$p_1', \dots, p_n', (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models q\theta$$

به شرطی که برای هر  $i$  داشته باشیم:

للم: برای هر جمله  $p$  توسط UI داریم:

1.  $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$
  - 2.
  2.  $p_1', \backslash; \dots, \backslash; p_n' \models p_1' \wedge \dots \wedge p_n' \models p_1'\theta \wedge \dots \wedge p_n'\theta$
  3. From 1 and 2,  $q\theta$  follows by ordinary Modus Ponens

N. Razavi- AI course - 2006

18

یاپگاه دانش نمونه

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
  - 
  - Prove that Colonel West is a criminal

N Razavi- AI course - 2006

19

بیانیه دانش نمونه (ادامه)

... it is a crime for an American to sell weapons to hostile nations:

$$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

Nono ... has some missiles, i.e.,  $\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x)$ :

*Owns(Nono,M1) and Missile(M1)*

... all of its missiles were sold to it by Colonel West

*Missile(x)  $\wedge$  Owns(Nono, x)  $\Rightarrow$  Sells(West, x, Nono)*

Missiles are weapons:

*Missile(x)  $\Rightarrow$  Weapon(x)*

An enemy of America counts as "hostile":

*Enemy(x,America)  $\Rightarrow$  Hostile(x)*

West, who is American ...

American (West)

The country Nono, an enemy of America

20

## الگوریتم استنتاج و ب جلو

```

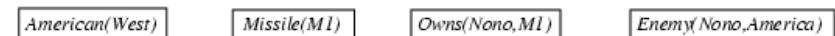
function FOL-FC-ASK( $KB, \alpha$ ) returns a substitution or false
repeat until  $new$  is empty
   $new \leftarrow \{ \}$ 
  for each sentence  $r$  in  $KB$  do
    ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ )  $\leftarrow$  STANDARDIZE-APART( $r$ )
    for each  $\theta$  such that  $(p_1 \wedge \dots \wedge p_n)\theta = (p'_1 \wedge \dots \wedge p'_n)\theta$ 
      for some  $p'_1, \dots, p'_n$  in  $KB$ 
         $q' \leftarrow$  SUBST( $\theta, q$ )
        if  $q'$  is not a renaming of a sentence already in  $KB$  or  $new$  then do
          add  $q'$  to  $new$ 
           $\phi \leftarrow$  UNIFY( $q', \alpha$ )
          if  $\phi$  is not fail then return  $\phi$ 
    add  $new$  to  $KB$ 
  return false

```

N. Razavi- AI course - 2006

21

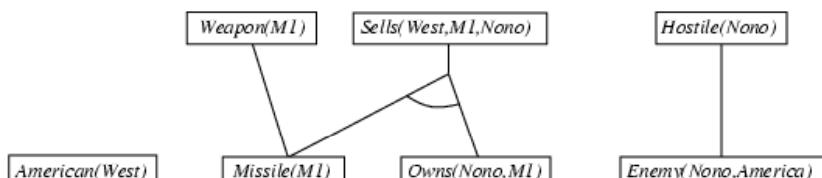
## اُبَاتَ تَوْسِطَ اسْتِنْتَاجَ و ب جلو



N. Razavi- AI course - 2006

22

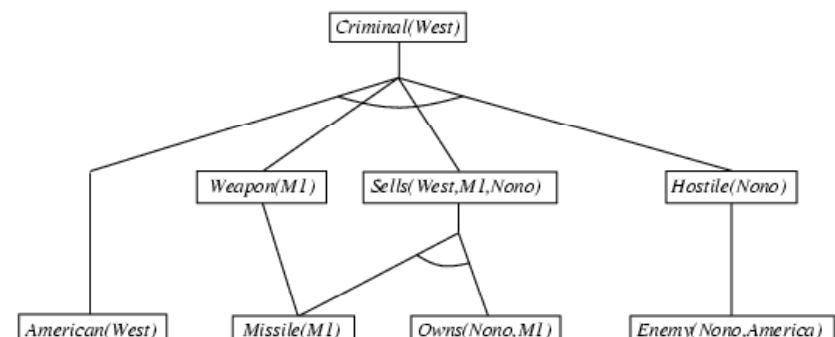
## اُبَاتَ تَوْسِطَ اسْتِنْتَاجَ و ب جلو



N. Razavi- AI course - 2006

23

## اُبَاتَ تَوْسِطَ اسْتِنْتَاجَ و ب جلو



N. Razavi- AI course - 2006

24

## خصوصیات استنتاج و به جلو

- برای فراکردهای معین FOL، کامل و صحیح است
- = فراکردهای معین FOL بدون سیمبول تابعی Datalog
- FC برای Datalog با تعداد تکرارهای محدودی متوقف می شود.
- در حالت کلی، اگر پایگاه دانش مستلزم  $\alpha$  نباشد، ممکن است متوقف نشود
- این اجتناب ناپذیر است: استلزم با فراکردهای معین نیمه تصمیم پذیر است

N. Razavi- AI course - 2006

25

## کارآی استنتاج و به جلو

- استنتاج رو به جلوی افزایشی: در تکرار  $k^{\text{th}}$  نیاز به تطبیق قانونی که هیچ کدام از شرایطش در تکرار  $k-1^{\text{th}}$  اضافه نشده اند، نیست
- خود انطباق می تواند پرهزینه باشد:
  - شاخص بندی پایگاه داده اجازه می دهد که حقایق شناخته شده در زمان  $O(1)$  بازیابی شوند.
- استنتاج رو به جلو به طور گسترده ای در پایگاه های داده استنتاجی به کار می رود

N. Razavi- AI course - 2006

26

## الگوریتم استنتاج و به عقب

```
function FOL-BC-ASK( $KB$ ,  $goals$ ,  $\theta$ ) returns a set of substitutions
  inputs:  $KB$ , a knowledge base
           $goals$ , a list of conjuncts forming a query
           $\theta$ , the current substitution, initially the empty substitution { }
  local variables:  $ans$ , a set of substitutions, initially empty
  if  $goals$  is empty then return { $\theta$ }
   $q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals))$ 
  for each  $r$  in  $KB$  where  $\text{STANDARDIZE-APART}(r) = (p_1 \wedge \dots \wedge p_n \Rightarrow q)$ 
    and  $\theta' \leftarrow \text{UNIFY}(q, q')$  succeeds
     $ans \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \dots, p_n | REST(goals)], \text{COMPOSE}(\theta, \theta')) \cup ans$ 
  return  $ans$ 
```

$$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$$

N. Razavi- AI course - 2006

27

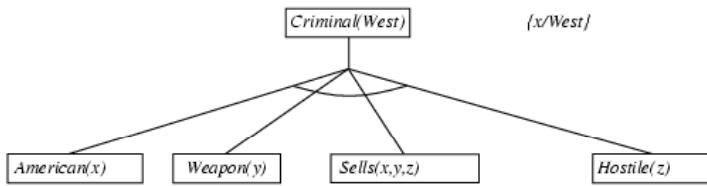
## مثال استنتاج و به عقب

Criminal(West)

N. Razavi- AI course - 2006

28

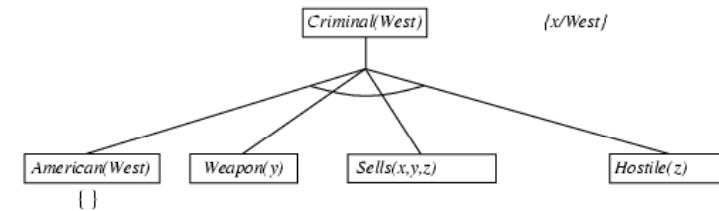
## مثال استنتاج و به عقب



N. Razavi- AI course - 2006

29

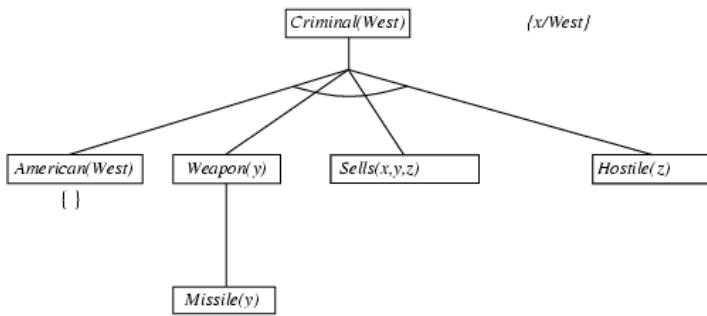
## مثال استنتاج و به عقب



N. Razavi- AI course - 2006

30

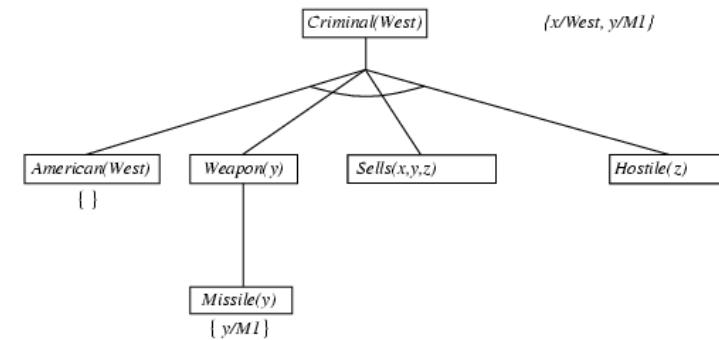
## مثال استنتاج و به عقب



N. Razavi- AI course - 2006

31

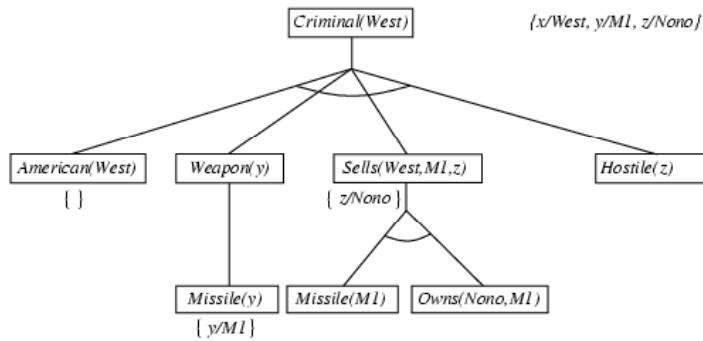
## مثال استنتاج و به عقب



N. Razavi- AI course - 2006

32

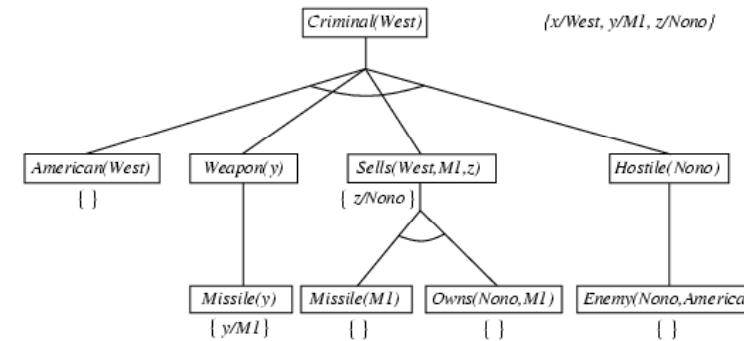
## مثال استنتاج و به عقب



N. Razavi- AI course - 2006

33

## مثال استنتاج و به عقب



N. Razavi- AI course - 2006

34

## خصوصیات استنتاج و به عقب

- اثبات: جستجوی اول-عمق بازگشته - فضای حالت بر حسب اثبات خطی
- ناکامل به دلیل وجود حلقه ها
  - راه حل: مقایسه هدف فعلی با تمام اهداف موجود در پشته
- ناکارآ به دلیل زیرهدف های تکرای (چه موفقیت و چه شکست)
  - راه حل: ذخیره نتایج قبلی (با مصرف حافظه بیشتر) - نوعی memoization
- به طور گسترده ای در **برنامه نویسی منطقی** استفاده می شود

N. Razavi- AI course - 2006

35

## برنامه نویسی منطقی: پرولوگ

- Algorithm = Logic + Control
- اصول: عقبگرد با فراکردهای هورن - به طور گسترده ای در اروپا و ژاپن استفاده شده است (در پروژه نسل پنجم کامپیوترها)
- برنامه = مجموعه ای از فراکردها
- فراکرد =

head :- literal<sub>1</sub>, ... literal<sub>n</sub>.

criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).

N. Razavi- AI course - 2006

36

## خصوصیات پرولوگ

- استنتاج روبره عقب، اول-عمق و از چپ به راست
- $X \text{ is } Y * Z + 3$
- عملگرهایی برای محاسبات ریاضی مثلاً:
- مسندهایی با عوارض جانبی (مثلاً ورودی و خروجی)
- فرض دنیای بسته (نقیض به عنوان شکست)
  - مثال: با داشتن

`alive(X) :- not dead(X).`

موفق می شود اگر `alive(Joe)` بخورد `not dead(Joe)`

N. Razavi- AI course - 2006

37

## پرولوگ

- اضافه کردن یک لیست به انتهای لیست دیگر برای تولید یک لیست جدید (عمل Append)

`append([ ], Y, Y).`

`append([X|L], Y, [X|Z]) :- append(L, Y, Z).`

- کوئیری:

`append(A, B, [1, 2]) ?`

- پاسخ ها:

`A = [ ] B = [ 1 , 2 ]`

`A = [ 1 ] B = [ 2 ]`

N. Razavi- AI course - 2006

38

`A = [ 1 . 2 ] B = [ 1 ]`

## (زولوشن: خلاصه)

- رزولوشن در منطق مرتبه اول

$$l_1 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n$$

$$(l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n) \theta$$

$$\text{Unify}(l_i, \neg m_j) = \theta$$

- فرض می شود که دو فراکرد استاندارد شده اند به گونه ای که شامل متغیر یکسانی نباشند
- مثال:

$$\neg Rich(x) \vee Unhappy(x)$$

$$Rich(Ken)$$

$$Unhappy(Ken)$$

with  $\theta = \{x/Ken\}$

N. Razavi- AI course - 2006

39

## CNF به تبدیل

- Everyone who loves all animals is loved by someone:  
 $\forall x [\forall y Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y Loves(y,x)]$

- 1. Eliminate biconditionals and implications

- $\forall x [\neg \forall y \neg Animal(y) \vee Loves(x,y)] \vee [\exists y Loves(y,x)]$

- 2. Move  $\neg$  inwards:  $\neg \forall x p \equiv \exists x \neg p$ ,  $\neg \exists x p \equiv \forall x \neg p$

- $\forall x [\exists y \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y Loves(y,x)]$

40

$$\forall x [\exists y \neg \neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$$

# تبدیل به CNF (ادامه)

3. Standardize variables: each quantifier should use a different one
4.  $\forall x [\exists y \text{Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists z \text{Loves}(z,x)]$
4. Skolemize: a more general form of existential instantiation.  
Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

$$\forall x [\text{Animal}(F(x)) \wedge \neg \text{Loves}(x,F(x))] \vee \text{Loves}(G(x),x)$$

5. Drop universal quantifiers:

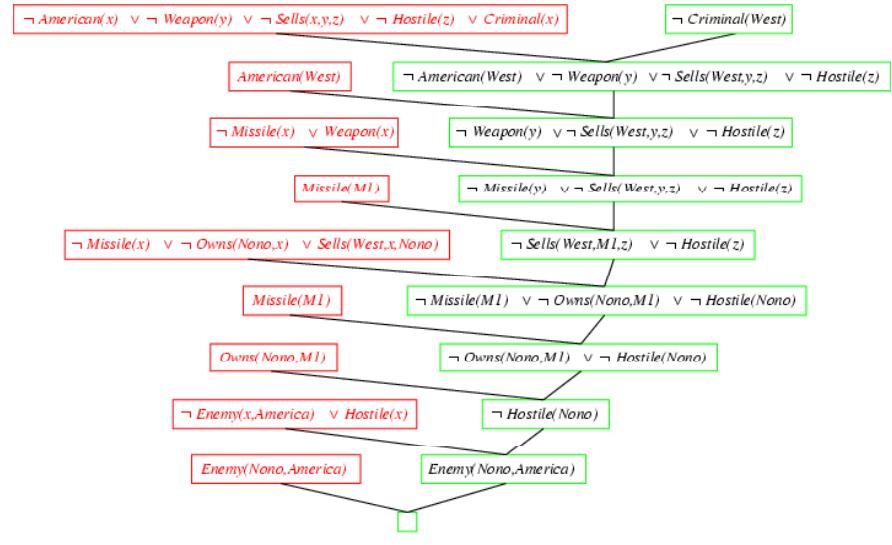
- 6.
6.  $[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x,F(x))] \vee \text{Loves}(G(x),x)$

N. Razavi- AI course - 2006

41

6. Distribute  $\vee$  over  $\wedge$ :

# اُبَات بوسیله (زولوشن: مثال



N. Razavi- AI course - 2006

42