



3

Lec 3

: فهرس الشابتر دا هنتكلم عن **

1. الايجينت اركتيكشر
2. نتكلم شويه عن السيمبولييك ايجينت • وهو ابسط انوع الايجينتس واللي بيعتمد على الرموز في شغله
3. هنتكلم عن الديداكتيف ريزونينج • ومالقصد بيه ان الايجينت بعد شويه تحليل بيستنتج افضل اكتشن ممكن ياخده لحل المشكله اللي قدامه

DEDUCTIVE REASONING AGENTS

3-1

-
- 3.1 Agents as Theorem Provers
 - 3.2 Agent-Oriented Programming
 - 3.3 Concurrent MetateM
-

3-2

- تحت هنا بيديك تعريف للايجينت ويقول انه :

Agent Architectures

- We want to build agents, that enjoy the properties of **autonomy**, **reactiveness**, **pro-activeness**, and **social** ability that we talked about earlier
- This is the area of *agent architectures*
- **Agent architecture can be defined as:**
 '[A] particular methodology for building [agents].
 It specifies how... the agent can be decomposed into the construction of a set of component modules.
 How these modules should be made to interact.
 The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions and future internal state of the agent.
 An architecture encompasses techniques and algorithms that support this methodology.'

3-4

- لما بنجي نتكلم عن الايجينت اركتيكشر .. احنا محتاجين اننا نخلص الايجينت بتاعنا دا
 1. يعني نخلصه بيشتغل ويعامل لوحده بصوره اوتوماتيكيه مع اي تغير ممكن : يحصل
 2. reactive .. بيقدر يتفاعل مع الايجينتس و البيئه اللي حوليه عشان يقدر ينفذ : التاسكات بافضل صوره ممكنه
 3. pro-active .. والمقصود فيها ان الايجينت عنده قدره علي توقع المشاكل اللي : ممكت تحصل في المستقبل واخذ خطوات استباقيه للتعامل معها
- وهنأ بيديك تعريف تاني و بيقول ان الأجينت اركتيكشر ويبيقولك انه :

Agent Architectures

- Another definition of Agent architecture to be:

'[A] specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules.

A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks.'

3-5

Agent Architectures

- An *agent* is a computer system capable of *flexible autonomous action*...
- Issues one needs to address in order to build agent-based systems...
- Three types of agent *architecture*:
 - symbolic/logical
 - reactive
 - hybrid

3-3

Agent Architectures

- Originally (1956-1985), most all agents designed within AI were **symbolic reasoning** agents
- Its purest expression proposes that agents use **explicit logical reasoning** in order to decide what to do
- Problems with symbolic reasoning led to a reaction against this — the so-called *reactive agents* movement, 1985–present
- From 1990-present, a number of alternatives proposed: *hybrid* architectures, which attempt to combine the **best of reasoning** and reactive architectures

3-6

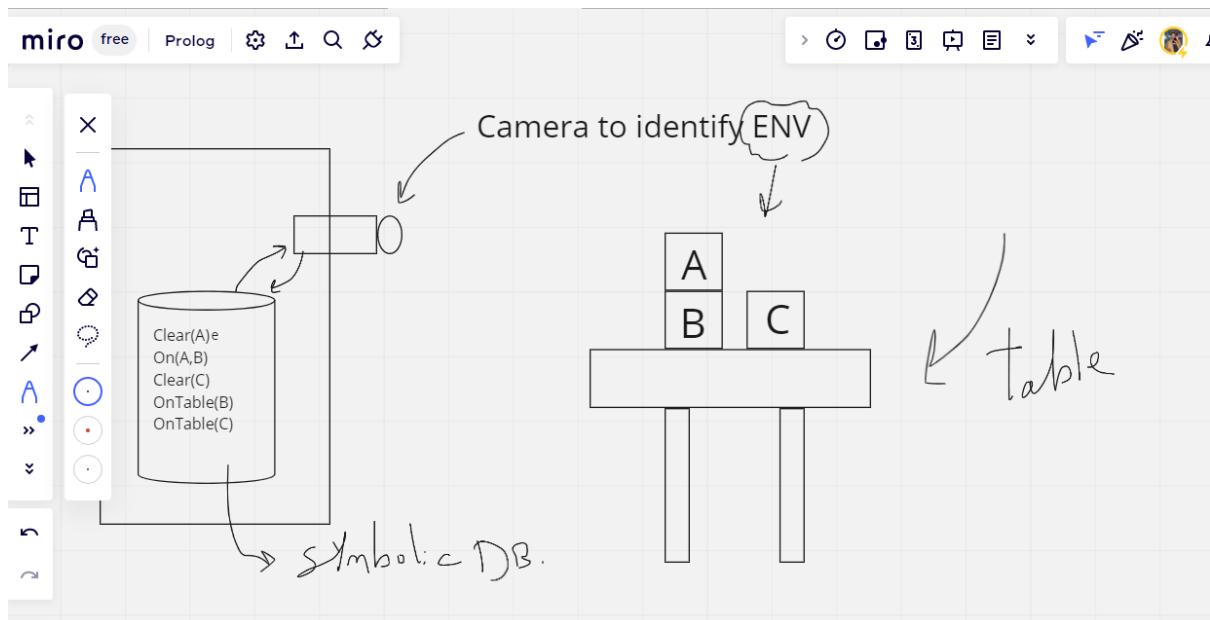
- ومن الكام تعريف اللي كانو فوق دول .. هتلaci لـ الـaiـجـيـنـتـ ليـهـاـ اـكـتـرـ منـ تـعـرـيفـ وـاـكـتـبـ :
- من طريـقـهـ لـاـنـشـائـهـمـ .. تـعـالـاـ نـشـوـفـ بـقاـ اـيـهـ اـنـوـاعـ الـaiـجـيـنـتـ الليـ مـمـكـنـ نـقـومـ بـيهـاـ

1. symbolic agents :

- من اقدم انواع الـaiـجـيـنـتـسـ .. وهو الـaiـجـيـنـتـ لـلـبـيـئـهـ حـولـيـهـ
- الليـ بـيـقـاـ عـنـدـهـ

**واطن باين او اي من اسمه يعني سيمبوليـكـ .. يعني بيحتاج **
سيمبولـزـ عـشـانـ يـقـدـرـ يـتـعـاـمـلـ**

- كلمـهـ سـيـمـبـولـزـ هـنـاـ المـقصـودـ بـيهـ الرـمـوزـ .. انـ الـaiـجـيـنـتـ دـاـ بـيـشـتـغـلـ وـيـتـعـرـفـ عـلـيـ الـبـيـئـهـ الليـ
- حـولـيـهـ منـ خـلـالـ مـجـمـوعـهـ سـيـمـبـولـزـ بـتـبـقـيـ مـتـخـزـنـهـ عـنـدـهـ
- مـثـالـ عـشـانـ نـفـهـمـ الـحـوارـ دـاـ اـكـتـرـ



- بص في المثال دا للبيئه بتاعت الايجينت و طريقه فهمه ليها .. هتللاقيها عباره عن مجموعه : سيمبولز من خلالها ايجينت قدر يفهم البيئه اللي حوليه

- معناها ان ال مفيش فوقها حاجه → (A)

- معناها ان على بي → (A, B)

- معناها ان السى علي التربيعه → (C)

- اخدت بالك .. هي جموعه رموز "سيمبولز" .. قدر ايجينت من خلالها انه يفهم البيئه ويبيقا "عنه سيمبوليک ريبريزنطيشن ليها"

- مثال علي النوع دا هو الريزونينج ايجينت بتاع المحاضره الجايه اللي هي شابتير 4

- والنوع دا كان فيه مشاكل كتيره او لحد اما طلع النوع اللي بعده واللي هو

وهو ايجينت رسپونسيف اكتر مع البيئه اللي حوليه "اشطر في التعامل" :
"مع البيئه اللي حوليه"

- الميزة في الرياكتيف عن السيمبوليک هو انه اكتر اكتر مع البيئه اللي حوليه .. مرن . وبيتعامل بصوره افضل مع البيئه المحيطه

- وبعد كدا قررو انهم ياخدو hybrid : Symbolic - reactive علشان بعد كدا يطلعو ب افضل حاجه في ال

- وبعد كدا المطورين قالك هناخد الميزة في كل واحد من الاثنين اللي فوق ونطلع النوع : الثالث اللي هو الهايدر

- تعالا نكتب تعريف لكل واحد فيهم بقا

1. Symbolic Architecture : agents use explicit logical reasoning in order to decide what to do .
2. Reactive : Problems with symbolic reasoning led to a reaction against this - led to reactive agents movement .
3. Hybrid Agents : Contain the best of Symbolic and reactive .

Symbolic Reasoning Agents

- The classical approach to building agents is to view them as a particular type of knowledge-based system, and bring all the associated (discredited?!) methodologies of such systems to bear
- This paradigm is known as **symbolic AI**
- We define a deliberative agent or agent architecture to be one that:
 - contains an explicitly represented, symbolic model of the world
 - makes decisions (for example about what actions to perform) via symbolic reasoning

3-7

** Symbolic Arch تعاالا بقا نتكلم شويه عن ال

- من السهل او ي انك تميز ال symbolic agent لو لقيت فيه :
 - لو انه لازم يحتوي علي ريبريزنتيشن صريح للبيئه اللي حوليه .. محتاج موديل عشان يفهم البيئه اللي حوليه
 - لما بيجي ياخد قرار بيأخذه من خلال لوجيکال ريزونينج
1. Symbolic explicit logical reasoning لو علشان يقرر هيعمل ايه .. بيقا على طول دا لقيت انه بيستخدم
 - في الاسلaid اللي فوق بيتكلم عن ازاي تميز السيمبوليک ايجينت
 - الاسلaid اللي جاي بيكلمك عن عيوب السيمبوليک ايجينت

Symbolic Reasoning Agents

- If we aim to build an agent in this way, there are two key problems to be solved:
 1. *The transduction problem*:
that of translating the **real world** into an accurate, adequate symbolic description, in time for that description to be useful...vision, speech understanding, learning
 2. *The representation/reasoning problem*:
that of how to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful...knowledge representation, automated reasoning, automatic planning

3-8

- .. وانت بتبني الايجينت دا هتقابلك مشكلتين
- 1. الترانسدكشن .. والمقصود بيها هي تحويل العالم الحقيقي ل شويه سيمبوليز يقدر من . خلالها الايجينت انه يتعرف على البيئه دي
- 2. الريبريزينتيشن نفسه .. في المثال اللي كان فوق كان الحوار سهل اوي لأنها بيئه بسيطة اوي .. لكن لو انت عامل ايجينت عباره عن عربيه تسوق نفسها مثلا .. انت متخيل شكل البيئه وكميه المتغيرات والتعقيدات الموجوده فيها !! .. مش هينفع خالص يا باشا جو الرموز بتاعك دا .. مش هتلافق تعبير عن ايه ولا ايه

Symbolic Reasoning Agents

- Most researchers accept that neither problem is anywhere near solved
- Underlying problem lies with the complexity of symbol manipulation algorithms in general: many (most) search-based symbol manipulation algorithms of interest are *highly intractable*
- Because of these problems, some researchers have looked to alternative techniques for building agents; we look at these later

3-9

- ملخص الاسلaid دا .. بيقولك ان معظم الباحثين م كانواش مرتاحين للنوع دا وشافين انه بيسكب مشاكل تانية
- ان الايجينت عشان يتعرف علي البيئه اللي حوليه بجو الرموز دا .. الموضوع معقد او يوي ويحتاج وقت ومعادلات وليله كبيره او يوي حضرتك
- لذلك كانوا بيعاولو يلجأو ل اي نظام تاني غير السيمبولييك الشلل دا

Deductive Reasoning Agents

- How can an agent decide what to do using theorem proving?
- Basic idea is to use **logic** to encode a theory stating the *best* action to perform in any given situation
- Let:
 - ρ be this theory (typically a set of rules)
 - Δ be a logical database that describes the current state of the world
 - Ac be the set of actions the agent can perform
 - $\Delta \models_{\rho} \phi$ mean that ϕ can be proved from Δ using ρ

3-10

- ندخل بقا في النوع الثاني وهو الديداكتيف ريزونينج
- كلمه ديداكتيف معناها ان الايجينت دا بيستنتج حلول المشاكل اللي قدامه من خلال تنفيذ شويه تحليلات واستنتاج افضل حل ممكن
- الفكرة البسيطة فيه هو ان الايجينت بيشوف الاكتشنز اللي هو ممكن يعملها .. وبعدين .. يحاول يثبت ان الاكتشن الفلاني هو افضل حل مبينهم كلهم .. ف يتعامل بيها
- بكل بساطه يحاول يثبت ل نفسه نظرية ان افضل اكتشن ممكن يعمله هو الاكتشن الفلاني عشان يطبقه فعليا

تعالا بقا نتكلم عن شويه النوتيشنر اللي الايجينت دا * * : بيستخدمهم

1. $P \rightarrow$ logical theory for optimal action to perform .

- الشوري اللي من خلالها الايجينت بيقدر يحدد الاوتيمايل اكتشن بناءه .. شويه قواعد اذا انطبقت على الاكتشن اللي الايجينت هي عمله بيقا الاكتشن دا اوتيمايل
- لثوري دي هي عباره عن مجموعه رولز منطقيه من خلالها الايجينت يقدر يحدد هل الاكتشن دا اوتيمايل والمناسب ولا لا

- اي ايجينت واحنا بنبنيه لازم بيبقا عنده الشيوري اللي بتوضحله ايه هو افضل اكشن ممكن ياخده
- الدلتا وهي اللوجيکال داتا بيز واللي بيبقا الايجينت مخزن فيها كل اللي يعرفه وعرفه عن . الانفiroمنت اللي هو فيها
- مش احنا كنا قابلين عن السيمبوليک انه بيعرف على البيئه اللي حوليه من خلال شويه سيمبولز ،، تمام دا نفس الفكره هو بيبقا مخزن عنده شويه سيمبولز بتعبر عن كل حاجه في الانفiroمنت اللي حوليه .
3. كل الاكشنز اللي ممكن الايجينت يعملها → AC
- والشكل الرابع دا معناه انك ممكن تثبت ان الفاي هي جزء من اللوجيال داتا بيز بتاعت الايجينت من خلال الشيوري الفلاينيه
- عارف انها مش واضحه او اي لكن هتووضح اكتر في المثال اللي جاي :

Deductive Reasoning Agents

```
/* try to find an action explicitly prescribed */
for each  $a \in Ac$  do
    if  $\Delta \models_p Do(a)$  then
        return  $a$ 
    end-if
end-for
/* try to find an action not excluded */
for each  $a \in Ac$  do
    if  $\Delta \not\models_p \neg Do(a)$  then
        return  $a$ 
    end-if
end-for
return null /* no action found */
```

3-11

- دلوقتي عايزين نتكلم عن اللوب اللي من خلالها الايجينت بيأخذ الاوتيمال اكشن بتاعه
- في الجزء الاول من الاسلайд ترجمه السيدوكود البسيط اللي قدامك دا هو

لكل اكشن محتمل ينتمي لمجموعه الاكتشنز اللي ممكن (Do(alpha) الايجينت يقوم بيها .. لو قدرت ثبت ال

انها من البيليفز بتاعت الايجينت "اللوجيکال داتا بيز بتاعته" .. لا وكمان انها بتحقق الثوري يعني عدت الرولز بتاعت الايجينت عشان تقدر تقول ان دا اوبيتيمال اكشن " .. ببقا ساعتها رجع الالفا دي لانها اوبيتيمال اكشن

- تخلص لوجع الدماغ دا .. لو قدرت ثبت ان الالفا انها واحد من الاكتشنز اللي الايجينت ممكن يقوم بيها .. لا وكمان بينطبق عليه الثوري "انه يعني الاكتشن دا اوبيتيمال " .. ببقا ساعتها خلاص قشطه الالفا دي اوبيتيمال والايجينت هينفذه بالفعل
- كل الفكره انك بتحاول ثبت هل الاكتشن دا اوبيتيمال ولا .. وللي بيحدد طبعا هي المعايير بتاعت الثوري

**طيب افرض بقا يا نجم اتنا ملقيناش الاوبيتيمال اكشن .. خلاص **
الايجينت هيتشل ويفضل مكانه .. لا طبعا لذلك بتدخل في اللوب الثاني وهو انت بتحاول تشووف ايه افضل حل موجود من كل حلول الممكنه .. احسن الوحشين يعني :**

- ترجمة الرموز :

لكل اكشن ممكن الايجينت يقوم بيها .. لو الالفا بتاعك يتعارض مع كل الرولز بتاعت الايجينت .. "دا معناه ان الاكتشن بتاعك دا مش نافع ولا مع حاله " .. ساعتها سيبك منه وشوف اللي بعده . لكن لو انطبقت عليه بعض الرولز والبعض الآخر لا .. ساعتها حلو رجعه لان الايجينت هيتعامل بيها .

- الموضوع وما فيه في اللوب الثاني هو انك بتحاول ثبت ان الاكتشن دا مش محظوظ علي الايجينت استخدامه
- في اللوب الاول انت كنت بتحاول ثبت ان الاكتشن دا هو الاوبيتيمال
- في اللوب الثاني انت بتحاول ثبت ان الاكتشن دا مش محظوظ
- وامتي الاكتشن يتحظر .. لما مينطبقش عليه ولا واحده من الرولز بتاعت الثوري للإيجينت

Deductive Reasoning Agents

```
Function: Action Selection as Theorem Proving
1. function action( $\Delta:D$ ) returns an action  $Ac$ 
2. begin
3.   for each  $\alpha \in Ac$  do
4.     if  $\Delta \vdash_p Do(\alpha)$  then
5.       return  $\alpha$ 
6.     end-if
7.   end-for
8.   for each  $\alpha \in Ac$  do
9.     if  $\Delta \not\vdash_p \neg Do(\alpha)$  then
10.      return  $\alpha$ 
11.    end-if
12.  end-for
13.  return null
14. end function action
```

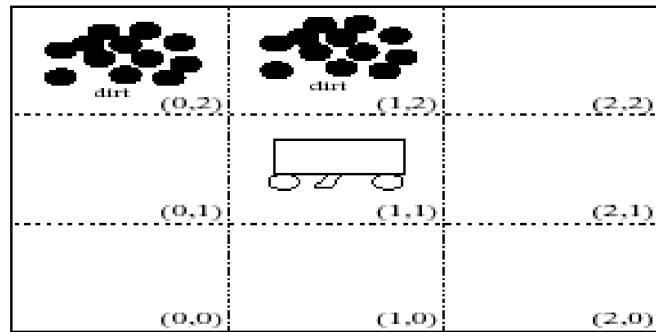
Figure 3.2 Action selection as theorem-proving.

3-12

- دا البرنامج كله علي بعضه وانت خلاص فهمت النقط اللي فيه

Deductive Reasoning Agents

- An example: The Vacuum World
- Goal is for the robot to clear up all dirt



3-13

- عندك مثل هنا وهي المكنسه .. زي مانت شايف هي بتشوف الاحداثيات بتاعتتها (X , Y) على اساس
- (0 , 0) , (2 , 2) لوكيشنر كتير بقا من

Deductive Reasoning Agents

- Use 3 *domain predicates* to solve problem:

$In(x, y)$ agent is at (x, y)

$Dirt(x, y)$ there is dirt at (x, y)

$Facing(d)$ the agent is facing direction d

- Possible actions:

$$Ac = \{turn, forward, suck\}$$

P.S. *turn* means “turn right”

3-14

- عندك السيمبوز اللي متخزنه في اللوجيال داتا بيز بتاعت الايجينت مثلاً
 - 1. هو ان الايجينت في الموضع الاحداثي $(x, y) : In(x, y)$
 - 2. الزباله كلها في الموضع الاحداثي $da : Dirt(x, y)$
 - 3. انت اتجاه وش الايجينت كذا $: Facing(d)$
- الاتجاهات عندك North , south , east , west
- الاكتشنز بتاعت الايجينت هما
 - 1. انه يلف يمين
 - 2. انه يتحرك واحده ل قدام
 - 3. انه يشفط

Deductive Reasoning Agents

- Rules ρ for determining what to do:

$$\begin{aligned} In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) &\longrightarrow Do(forward) \\ In(0,1) \wedge Facing(north) \wedge \neg Dirt(0,1) &\longrightarrow Do(forward) \\ In(0,2) \wedge Facing(north) \wedge \neg Dirt(0,2) &\longrightarrow Do(turn) \\ In(0,2) \wedge Facing(east) &\longrightarrow Do(forward) \end{aligned}$$

- ...and so on!
- Using these rules (+ other obvious ones), starting at (0, 0) the robot will clear up dirt

3-15

- هنا بيديك مثال على شويه الرولز بتاعت الايجينت ف مثلاً
 1. الايجينت في المكان 0 و 0 وتجاهه للشمال والزباله مش في المكان 0 و 0 دا .. ساعتها بالعقل كدا الايجينت هيفضل واقف في المكان دا يعمل ايه ؟ .. ف يتحرك ل قدام احسن وهكذا

Deductive Reasoning Agents

■ Problems:

- How to convert video camera input to $Dirt(0, 1)$?
- decision making assumes a *static* environment: *calculative* rationality
- decision making using first-order logic is *undecidable*!

■ Even where we use *propositional* logic, decision making in the worst case means solving co-NP-complete problems (PS: co-NP-complete = bad news!)

■ Typical solutions:

- weaken the logic
- use symbolic, non-logical representations
- shift the emphasis of reasoning from *run time* to *design time*

3-16

- المشاكل اللي مقبلاك في الحوار دا
- 1. ازاي هيقدر الايجينت يحول مكان الزبالة واللي هو اتعرف عليه من خلال الكاميرا ل احداثيات
- 2. لو اخذ القرار كان معتمد علي بيئه مفيش حاجه فيها بتتغير وفجاه في حاجه اتغيرت
- بعض الحلول بقا للحوار
- 1. انك تضعف اللوجيك بتاعت الايجينت .. بس دا طبعا ممكن يادي ل مصايب .. يعني الايجينت ممكن يعمل حاجات مينفعش اساسا انها تتعمل .. بس انت السبب انت اللي اضعف اللوجيك بتاعه
- 2. انك تستخدم السيمبوليک ريبريزنتشن
- 3. انك ترجع تعدل في طريقة تفكير الايجينت والالجوريزم بتاعه

More Problems...

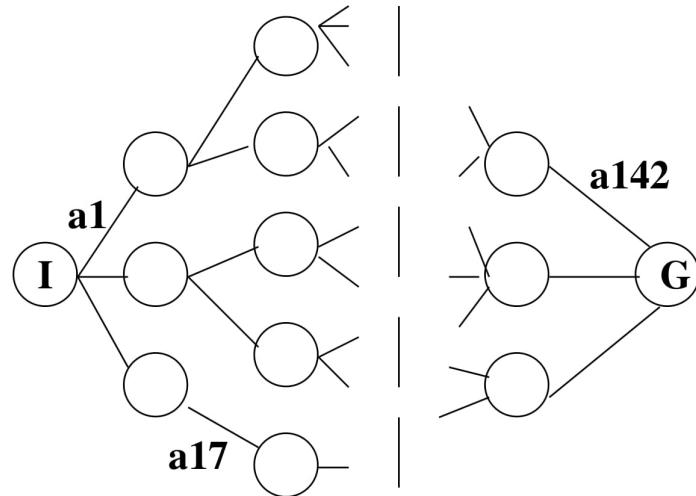
- The “logical approach” that was presented implies adding and removing things from a database
- That’s not pure logic
- Early attempts at creating a “planning agent” tried to use true logical deduction to solve the problem

3-17

- اللوجيکال ابروتش اللي كنا لسه بنتكلم فيه اوضحلنا احتياجاتنا ل اننا نشيل حاجات ونصيف حاجات علي الداتا بيز ودا طبعا مشكله .. انت كل اما تحط الايجينت في حته هتضطر انك تقعد تعدل وتعمل
- دا طبعا مش لوجيك بيور
- وهنا احتاجنا اننا نعمل بلانينج ايجينت يحاول انه يتسترج ايه افضل حل للمشكله اللي قدامه دي

Planning Systems (in general)

- Planning systems find a sequence of actions that transforms an initial state into a goal state



3-18

- الـplanning سـystem هو السـystem اللي بـيحاـول انه يـلاقـي تـرتـيب مـجمـوعـه الاـكـشنـز اللي وـراـ . بعضـها والـلـي من خـلـال يـقدـر يـوصل لـلـجـول بـتـاعـه .

Planning

- Planning involves issues of both Search and Knowledge Representation
- Sample planning systems:
 - Robot Planning (STRIPS)
 - Planning of biological experiments (MOLGEN)
 - Planning of speech acts
- For purposes of exposition, we use a simple domain – The Blocks World

3-19

- البلاينج هتلaci فيه مشكلتين 2 .. هما البحث و المعرفه
- : مثال افضل للتوضيح .. هنسخدم جو المكعبات وكدا عشان نفهم

The Blocks World

- The Blocks World (today) consists of equal sized blocks on a table
- A robot arm can manipulate the blocks using the actions:
 - UNSTACK(a, b)
 - STACK(a, b)
 - PICKUP(a)
 - PUTDOWN(a)

3-20

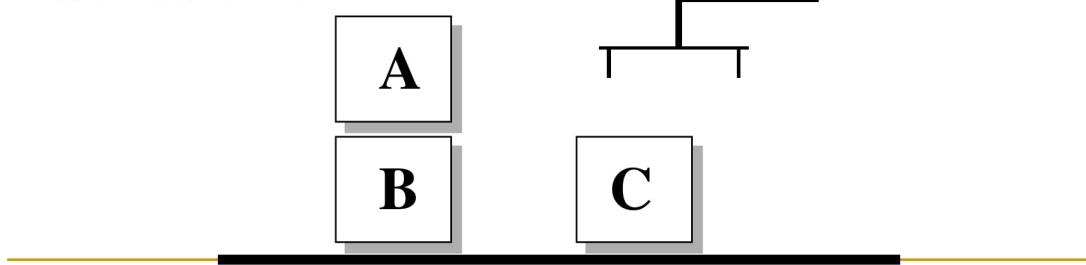
- شوبيه المكعبات اللي قدامك دول متساويين في الحجم
- : انت عندك ارم روبوت و معاك شوبيه اكشنز ممكن تقوم بيهم
 1. انهم تنزل || جنب بي
 2. انك تحط || فوق بي
 3. انك تخلي الدراع يشيل || ويس
 4. انك تخلي الدراع ينزل || ويبقا فاضي

The Blocks World

- We also use predicates to describe the world:

- ON(A,B)
- ONTABLE(B)
- ONTABLE(C)
- CLEAR(A)
- CLEAR(C)
- ARMEMPTY

In general:
ON(a,b)
HOLDING(a)
ONTABLE(a)
ARMEMPTY
CLEAR(a)



3-21

- وصف الاستيت الحالي هو :

ان المكعب || على المكعب :
On (A , B)
ان بي علي التربيزه علي طول :
OnTable(B)
ان سي علي التربيزه علي طول :
OnTable(C)
ان || مفيش فوقها حاجه :
Clear (A)
ان سي مفيش فوقها حاجه :
Clear (C)
الدراع فاضي مش ماسك ولا مكعب :
ARMEMPTY

- احنا الهدف بتاعنا اتنا نخلی المكعبات كلها جنب بعض .. يعني هنفدا (A , B)
- بعض في الكود اللي علي اليمين وانت تفهم ازاي البروسيس دي تمت

Agent Oriented Programming

3-22

AGENT0 and PLACA

- Much of the interest in agents from the AI community has arisen from Shoham's notion of *agent oriented programming* (AOP)
- AOP a 'new programming paradigm, based on a societal view of computation'
- The key idea that informs AOP is that of directly programming agents in terms of intentional notions like belief, commitment, and intention
- The motivation behind such a proposal is that, as we humans use the intentional stance as an *abstraction* mechanism for representing the properties of complex systems.
In the same way that we use the intentional stance to describe humans, it might be useful to use the intentional stance to program machines.

3-23

- جماعه الذكاء الصناعي قررو انهم يعملو ايجينت اوريانتد بروجراميينج طلعاو لغه 0 اسمها
 - في النقطه الثانيه بيقولك انه استايل جديد في البرمجه معتمد علي فكره المجتمعات وكدا
 - احنا ك بشر اللي بيحركنا هي معتقدتنا وكمان رغباتنا .. الحاجات اللي احنا حابيين نحققها .. ف احنا برضو عايزين نعمل للايجينتس دول
1. ”بيليف“ حاجات هو معتقد فيها
 2. انتشن .. وهي الحاجات اللي الايجينت بيرغب في انه يتحققها
 3. الكومتمتنس هي الالتزامات .. الحاجات اللي ضوريه علي الايجينت انه ينفذها

AGENT0

- Shoham suggested that a complete AOP system will have 3 components:
 - a logic for specifying agents and describing their mental states
 - an interpreted programming language for programming agents
 - an ‘agentification’ process, for converting ‘neutral applications’ (e.g., databases) into agents
- Results only reported on first two components.
- Relationship between logic and programming language is *semantics*
- We will skip over the logic(!), and consider the first AOP language, AGENT0

3-24

AGENT0

- AGENT0 is implemented as an extension to LISP
- Each agent in AGENT0 has 4 components:
 - a set of capabilities (things the agent can do)
 - a set of initial beliefs
 - a set of initial commitments (things the agent will do)
 - a set of *commitment rules*
- The key component, which determines how the agent acts, is the commitment rule set

3-25

- لغه برمجه الايجينت زورو دي بتكون من 4 حاجات بمعني عشان تقدر تكود في الايجينت 0 :
زي انت تحتاج
1. مجموعه الحاجات اللي الايجينت يقدر يعملها
2. اعتقادات اوليه ,، بتدي من خلالها للايجينت وصف اولي للانفيرومنت اللي حوليه
3. الكومتمننس وهي الاولويات اللي علي الايجينت انه ينفذها
4. الكومتمننس رولز ,، ودي هي الشيوري اللي بتوصف الاوتيمايل اكتشن واللي كنا بنتكلم عليها في الديداكتف ريزونينج ووظيفتها هي انها تقول للايجينت ازاي يقدر ينتج كومتمننس جديده بناء علي اللي بيحصل في الانفيرومنت اللي حوليه .

AGENTO

- Each commitment rule contains
 - a *message condition*
 - a *mental condition*
 - an action
- On each ‘agent cycle’...
 - The message condition is matched against the messages the agent has received
 - The mental condition is matched against the beliefs of the agent
 - If the rule fires, then the agent becomes committed to the action (the action gets added to the agent’s commitment set)

3-26

- الكوممنت رول واللي وظيفتها انها توصل الايجينت للاوبيتمال اكشن + انها تعمله : كوممنتس جديده .. بت تكون من 3 حاجات
 1. Message Condition : بحيث انها بتوافق مع باقي رسائل الايجينت
 2. Metal Condition : بحيث انها تتوافق مع البيليفز بتاعت الايجينت دا
 3. action : وهو الاكشن اللي متواافق مع رولز الكوممنتس دي

طيب حلو اوي .. احنا فهمنا اجزائه .. بيشتغل ازاي بقا ؟ **

1. انه بيتص على الماسج وبيشوف هل هي متوافقه مع باقي الماسجز ولا لا
2. بيشوف هل المينتال كوندشن بتاعه مقبول ولا مرفوض
3. اذا كان الماسج كوندشن مطابق والمتنال كوندشن مقبول او مطابق .. ساعتها بيدا يعمل اكشن

AGENTO

- Actions may be
 - *private*:
an internally executed computation, or
 - *communicative*:
sending messages
- Messages are constrained to be one of three types:
 - “requests” to commit to action
 - “unrequests” to refrain from actions
 - “informs” which pass on information

3-27

- انت عندك نوعين من الاكتشنز
- 1. البرايفت اكتشن .. ودا عامل زي default constructor in OOP
- حاجه بيحصلها اكسكيوشن في الخبائه كدا من غير مانت تعرف
- 2. كوميونيتييف : وهو ارسال الرسائل ل باقي الايجينتس
- وبما اننا قولنا ان الايجينت بيبعث رسائل .. يبقا لازم نعرف ايه هي انواع الرسائل دي
- 1. ريكويست : ان الايجينت بيطلب من الثاني انه ينفذ حاجه معينه
- 2. ان ريكويست : وهو انه بيسليغ الايجينت الثاني انه خلاص مبقاش عايزةها
- 3. وهو ان الايجينت بيسليغ الثاني بحاجه هو عملها عشان يضيفها للبيليفز بتاعتة

AGENT0

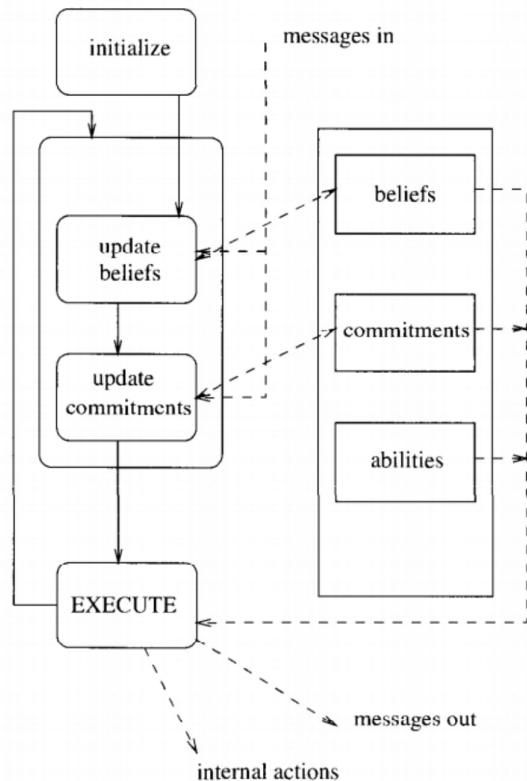


Figure 3.4 The flow of control in Agent0.

3-28

AGENT0

- A commitment rule:

```
COMMIT(  
    ( agent, REQUEST, DO(time, action)  
    ), ;;; msg condition  
    ( B,  
        [now, Friend agent] AND  
        CAN(self, action) AND  
        NOT [time, CMT(self, anyaction)]  
    ), ;;; mental condition  
    self,  
    DO(time, action)  
)
```

- هنا دا مثال على الكومتمنت رول
- كل اللي فوق كلمه سيلف هو كوندشن بارت .. واللي تحتها هو ال اكشن بارت
- تعالا نشوف بقا ترجمت الكلام دا ايه
- الماسج كوندشن من الرساله

الايجينت الفلاني بعاتلي ماسج من النوع ريكويست بيطلب مني اني انفذ اكشن مين في وقت كذا

- المنتال كوندشن من الرساله
 - نفهم تفاصيل الاول وبعدين نجمع الطلب
1. **البيليفر بتاعت ايجينت :**
- ان ايجينت اللي طلب دا صديق لي : [Now , Friend agent]
 - اقدر اقوم ب الاكشن دا : Can (self , action)
 - اني في الوقت المطلوب في الكومتمنت اللي : [(time , CMT) self , anyaction]
جايه .. انا فاضي مش بعمل حاجه

من خلال البيليفر بتاعتني "B" ان ايجينت اللي طلب دا هو صديق لي
وانني اقدر بالفعل اني اقوم بي
والوقت اللي هو بيطلب فيه اني اعمل كذا .. انا فاضي مش مشغول
في حاجه
يبقا بديهي جدا اني هنفذ الاكشن دا في الوقت المطلوب

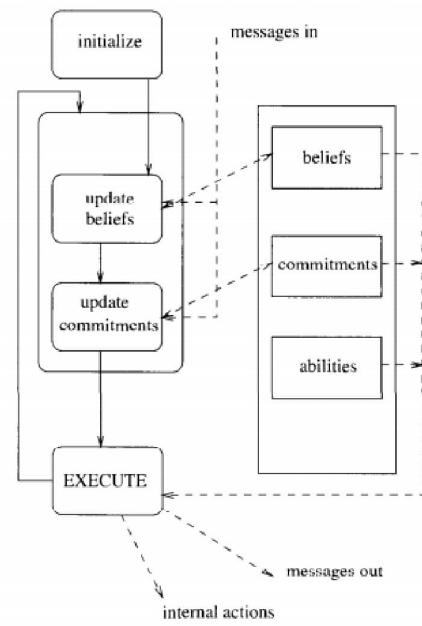
AGENT0

- This rule may be paraphrased as follows:
if I receive a message from *agent* which
requests me to do *action* at *time*, and I believe
that:
 - *agent* is currently a friend
 - I can do the action
 - At *time*, I am not committed to doing any other
actionthen commit to doing *action* at *time*

3-30

The operation of an agent can be described by the following loop (see Figure 3.4).

- (1) Read all current messages, updating beliefs - and hence commitments - where necessary.
- (2) Execute all commitments for the current cycle where the capability condition of the associated action is satisfied.
- (3) Goto (1).



3-31

AGENT0 and PLACA

- AGENT0 provides support for multiple agents to cooperate and communicate, and provides basic provision for debugging...
- ...it is, however, a *prototype*, that was designed to illustrate some principles, rather than be a production language
- A more refined implementation was developed by Thomas, for her 1993 doctoral thesis
- Her Planning Communicating Agents (PLACA) language was intended to address one severe drawback to AGENT0: the **inability of agents to plan, and communicate requests** for action via high-level goals
- Agents in PLACA are programmed in much the same way as in AGENT0, in terms of *mental change* rules

3-32

Concurrent METATEM

3-33

Concurrent METATEM

- Concurrent METATEM is a multi-agent language in which each agent is programmed by giving it a ***temporal logic*** specification of the behavior it should exhibit
- These specifications are executed directly in order to generate the behavior of the agent
- Temporal logic is classical logic augmented by *modal operators* for describing how the truth of propositions **changes over time**

3-34

Concurrent METATEM

- For example. . .

Table 3.1 Temporal connectives for Concurrent MetateM rules.

Operator	Meaning
$\circ \varphi$	φ is true 'tomorrow'
$\bullet \varphi$	φ was true 'yesterday'
$\diamond \varphi$	at some time in the future, φ
$\Box \varphi$	always in the future, φ
$\blacklozenge \varphi$	at some time in the past, φ
$\blacksquare \varphi$	always in the past, φ
$\varphi U \psi$	φ will be true until ψ
$\varphi S \psi$	φ has been true since ψ
$\varphi W \psi$	φ is true unless ψ
$\varphi Z \psi$	φ is true zince ψ

3-35

Concurrent METATEM

- MetateM is a framework for *directly executing* temporal logic specifications
- The root of the MetateM concept is Gabbay's *separation theorem*:
Any arbitrary temporal logic formula can be rewritten in a logically equivalent $\text{past} \Rightarrow \text{future}$ form.
- This $\text{past} \Rightarrow \text{future}$ form can be used as *execution rules*
- A MetateM program is a set of such rules
- Execution proceeds by a process of continually matching rules against a "history", and *firing* those rules whose antecedents are satisfied
- The instantiated future-time consequents become *commitments* which must subsequently be satisfied

3-36

Concurrent METATEM

- Execution is thus **a process of iteratively** generating a model for the formula made up of the program rules
- The future-time parts of instantiated rules represent *constraints* on this model
- An example MetateM program: the resource controller...
$$\begin{array}{c} \forall x \quad \text{ask}(x) \Rightarrow \diamond \text{give}(x) \\ \forall x,y \quad \text{give}(x) \wedge \text{give}(y) \Rightarrow (x=y) \end{array}$$
- First rule ensure that an 'ask' is eventually followed by a 'give'
- Second rule ensures that only one 'give' is ever performed at any one time
- There are **algorithms** for executing MetateM programs that appear to give reasonable performance
- There is also *separated normal form*

3-37

Concurrent METATEM

- ConcurrentMetateM provides an operational framework through which societies of MetateM processes can operate and communicate
- It is based on a new model for concurrency in executable logics: the notion of executing a logical specification to generate individual agent behavior
- A ConcurrentMetateM system contains a number of agents (objects), each object has 3 attributes:
 - a name
 - an interface
 - a MetateM program

3-38

Concurrent METATEM

- An object's interface contains two sets:
 - environment predicates — these correspond to messages the object will accept
 - component predicates — correspond to messages the object may send
- For example, a 'stack' object's interface:
$$\text{stack}(\text{pop}, \text{push})[\text{popped}, \text{stackfull}]$$

{pop, push} = environment preds
{popped, stackfull} = component preds
- If an agent receives a message headed by an environment predicate, it accepts it
- If an object satisfies a commitment corresponding to a component predicate, it broadcasts it

3-39

The actual execution of an agent in Concurrent MetateM is, superficially at least, very simple to understand. Each agent obeys a cycle of trying to match the past-time antecedents of its rules against a *history*, and executing the consequents of those rules that ‘fire’. More precisely, the computational engine for an agent continually executes the following cycle.

- (1) Update the *history* of the agent by receiving messages (i.e. environment propositions) from other agents and adding them to its history.
- (2) Check which rules *fire*, by comparing past-time antecedents of each rule against the current history to see which are satisfied.
- (3) *Jointly execute* the fired rules together with any commitments carried over from previous cycles.
This involves first collecting together consequents of newly fired rules with old commitments – these become the *current constraints*. Now attempt to create the next state while satisfying these constraints. As the current constraints are represented by a disjunctive formula, the agent will have to choose between a number of execution possibilities.
Note that it may not be possible to satisfy *all* the relevant commitments on the current cycle, in which case unsatisfied commitments are carried over to the next cycle.
- (4) Goto (1).

3-40

Concurrent METATEM

- Summary:
 - an(other) experimental language
 - very nice underlying theory...
 - ...but unfortunately, lacks many desirable features
 - could not be used in current state to implement ‘full’ system
 - **currently** prototype only, full version on the way!

3-41

