3
Implementando Agentes Inteligentes

Neste capítulo considera-se como construir um agente BDI (Woo00), que é o utilizado no presente trabalho, de maneira mais completa. Isso será apresentado introduzindo-se progressivamente arquiteturas de agentes mais complexas e, para cada uma dessas arquiteturas, investiga-se o tipo de comportamento que tal agente exibiria. Com isso motiva-se a introdução de uma arquitetura mais refinada para um agente.

3.1
Introdução

Assim como acontece com os agentes lógicos, o tempo também pode ser problemático. Os processos de deliberação e de raciocínio *means-ends* não são instantâneos. Eles possuem um custo de tempo que lhes são associados. Supondo que o agente comece a deliberação em t_0, comece o raciocínio *means-ends* em t_1 e, posteriormente, comece a executar a intenção escolhida em t_2, o tempo decorrido na deliberação é, portanto, $t_d = t_1 - t_0$, e o tempo decorrido no raciocínio *means-ends* é de $t_m = t_2 - t_1$.

Adicionalmente, supondo que o processo de deliberação seja ótimo, isto é, que se o agente selecionar uma intenção para ser atingida, a conclusão dessa intenção é ótima no tempo t_0, no tempo t_1, o agente selecionou uma intenção a ser atingida que teria sido ótima se tivesse sido atingida em t_0. Porém, a não ser que t_d seja muito pequeno, o agente corre o risco de que a intenção selecionada não seja mais ótima em t_1.

Obviamente, a seleção de uma intenção a ser atingida é apenas uma parte do todo no problema do raciocínio prático; o agente ainda tem que determinar como atingir a intenção, através do raciocínio *means-ends*.

Assim como a deliberação, assume-se que o processo de raciocínio *means-ends* seja ótimo no sentido de que, se ao agente for dada uma intenção a ser atingida, e ele começar a executá-la no tempo t_1, ele irá selecionar a ação para atingir essa intenção de forma que essa ação seria ótima se tivesse sido executada em t_1. Novamente, o raciocínio *means-ends* não é instantâneo. Leva-se tempo para encontrar a melhor ação para atingir a intenção. Deste ponto

PUC-Rio - Certificação Digital Nº 0321119/CB

em diante, uma ação será representada por um plano, que é composto por uma seqüência de ações. Logo, se o agente tiver n possíveis ações disponíveis em um dado tempo, haverá $n!$ planos possíveis que são seqüências dessas ações. Como $n!$ cresce exponencialmente, a técnica mais óbvia para se encontrar um plano (buscar sistematicamente o espaço de seqüências de ações possíveis) será impossível na prática a não ser para valores de n bem pequenos. O problema é pior pelo fato de que o plano é construído com base em uma observação feita do ambiente em t_0. O ambiente pode ser muito diferente em t_2.

Assumindo que o agente tem os componentes de deliberação e o raciocínio *means-ends* ótimos da maneira descrita acima, deve ficar claro que esse agente terá o comportamento global ótimo nas seguintes circunstâncias:

- Quando a deliberação e o raciocínio *means-ends* levarem uma quantidade de tempo bem pequena;
- Quando houver a garantia de que o ambiente permanecerá estático enquanto o agente estiver deliberando e realizando o raciocínio *means-ends*, de forma que as hipóteses nas quais se baseiam a escolha da intenção a ser atingida e o plano para atingir a intenção até que o agente tenha terminado a deliberação e o raciocínio *means-ends* continuem válidas, ou
- Quando houver a garantia de que uma intenção que seria ótima se fosse atingida em t_0 (quando o ambiente foi observado) continuará ótima até o tempo t_2 (o tempo no qual o agente escolheu um plano para atingir a intenção).

As duas últimas condições não valerão na maioria dos ambientes reais. Afinal, os tipos de ambientes nos quais os agentes tendem a trabalhar tendem a ser altamente dinâmicos. Portanto, tais ambientes não permanecerão estáticos por tempo suficiente para que um agente determine uma intenção ótima ou, dada uma intenção, um plano ótimo para atingi-la. De maneira semelhante, a maioria das tarefas que serão dadas aos agentes serão altamente dependentes de tempo.

Na seção 2.4.2, formalizou-se as notações de crenças, desejos e intenções. Também é necessário uma maneira para representar a percepção do agente. Serão usados $p, p', p_1, ...$ para representar as percepções que o agente recebe. P, como em 2.4.2, representa o conjunto de todas as percepções.

PUC-Rio - Certificação Digital Nº 0321119/CB

3.2 Planos

É necessário agora definir o que é plano. Os planos e as intenções estão intimamente ligados. Freqüentemente se usa os dois termos no dia a dia como se tivessem o mesmo significado. Porém, neste trabalho, planos se referem a *receitas* para se atingir intenções. Um plano é visto como uma tupla consistindo de:

- uma pré-condição, que define as circunstâncias nas quais um plano é aplicável;
- uma pós-condição, que define quais as conseqüências da execução do plano, e
- um corpo, que é a parte do plano que é a sua "receita" – para este texto, o corpo é simplesmente uma seqüência de ações.

π (com decorações $\pi', \pi_1, ...$) será usado para denotar planos, e $Plan$ será o conjunto de todos os planos (sobre algum conjunto de ações A). Serão feitas algumas definições auxiliares para manipular os planos:

- Se π for um plano, $pre(\pi)$ denota a pré-condição de π; $pos(\pi)$, a pós-condição de π, e $corpo(\pi)$, o corpo de π;
- Se π for um plano, $vazio(\pi)$ significa que o plano π é a seqüência vazia (portanto, a função $vazio(\,)$ é uma função booleana);
- $executar(\,)$ é um procedimento que tem como entrada um único plano e executa-o até o fim sem parar – executar um plano significa simplesmente executar cada ação do corpo do plano na devida ordem;
- Se π for um plano, $cab(\pi)$ significa o plano feito da primeira ação do corpo do plano de π (por exemplo se o corpo de π for $\alpha_1, \alpha_2, ..., \alpha_n$, o corpo de $cab(\pi)$ contém apenas a ação α_1);
- Se π for um plano, $resto(\pi)$ significa o plano feito de todas as ações do corpo do plano π menos a primeira (por exemplo, se o corpo de π for $\alpha_1, \alpha_2, ..., \alpha_n$, o corpo de $resto(\pi)$ contém as ações $\alpha_2, ..., \alpha_n$);
- Se π for um plano; $I \subseteq Int$, um conjunto de intenções, e $B \subseteq Bel$, um conjunto de crenças, $consis(\pi, I, B)$ significa que π é um plano consistente para se atingir I com as crenças B.

Na seção 2.4.2 foram definidas as funções $frcr()$, $opcoes()$ e $filtro()$. Também é necessário definir a função

$$plano : \mathcal{P}(Bel) \times \mathcal{P}(Int) \rightarrow Plan,$$

PUC-Rio - Certificação Digital Nº 0321119/CB

onde $Plan$ é o conjunto de planos. Essa função determina o plano para atingir a intenção escolhida baseado nas crenças e nas intenções atuais. Deve-se notar que não há nada na definição da função $plano(\,)$ que requeira que o agente se engaje na geração do plano (construir o plano do nada (All90)). Na maioria dos sistemas BDI, a função $plano(\,)$ é implementada fornecendo ao agente uma biblioteca de planos (Geo87). Uma biblioteca de planos é uma coleção de planos pré-compilada, que um construtor de agentes fornece a um agente. Achar um plano para atingir uma intenção simplesmente envolve uma única passada pela biblioteca de planos para achar um plano que, ao ser executado, terá a intenção como uma pós-condição e será consistente com as crenças do agente. Em implementações de agentes BDI, as pré-condições e as pós-condições são geralmente representadas como (listas de) átomos de lógica de primeira ordem, e crenças e intenções como átomos já instanciados de lógica de primeira ordem. Portanto, achar um plano para atingir uma intenção se reduz a achar um plano cuja pré-condição se unifique com as crenças do agente e cuja pós-condição, com a intenção.

Pode-se, agora, reescrever o algoritmo para a função $acao()$ descrito na seção 2.4.2 usando a definição de plano. Note-se que a função $execucao()$ agora executa um plano e não mais uma intenção:

1. $funcao\ acao(p : P) : A$
2. $inicio$
3. $\quad B := frcr(B, P)$
4. $\quad D := opcoes(B, I)$
5. $\quad I := filtro(B, D, I)$
6. $\quad \pi := plano(B, I)$
7. $\quad retorne\ execucao(\pi)$
8. fim_acao

Figura 3.1: Função $acao()$ de um agente BDI com planos

Para os próximos algoritmos, a função $acao()$ não aparecerá explicitamente e sim o *loop* de controle do agente. Então, o algoritmo acima é equivalente ao da figura 3.2, onde B_0 e I_0 são as crenças e as intenções iniciais, respectivamente.

Esse algoritmo destaca algumas limitações dessa abordagem simples do controle do agente. Em particular, os passos de (5) a (9) inclusive assumem que o ambiente não se modificou desde que foi observado. Assumindo que o tempo levado para de fato executar o plano domine o tempo levado para revisar as crenças, os desejos e as intenções, e construir o plano, a preocupação principal é a de que o ambiente pode se modificar enquanto o plano está sendo executado. O problema é que o agente permanece comprometido com a intenção que ele

PUC-Rio - Certificação Digital Nº 0321119/CB

1. $B := B_0$
2. $I := I_0$
3. $enquanto\ verdadeiro\ faca$
4. $\quad pegue\ a\ proxima\ percepcao\ p$
5. $\quad B := frcr(B, P)$
6. $\quad D := opcoes(B, I)$
7. $\quad I := filtro(B, D, I)$
8. $\quad \pi := plano(B, I)$
9. $\quad execucao(\pi)$
10. $fim_enquanto$

Figura 3.2: *Loop* de controle de um agente BDI com planos

forma no passo (7) até que ele tenha executado o plano no passo (9). Se o ambiente mudar depois do início da função *acao*(), as hipóteses nas quais esse plano depende podem muito bem se tornarem inválidas antes que o plano seja de fato executado.

3.3 Estratégias de Comprometimento

Quando uma opção passa com sucesso pela função de filtro e é, portanto, escolhida pelo agente como uma intenção, diz-se que o agente se comprometeu com essa opção. Comprometimento significa persistência temporal (uma intenção, uma vez adotada, não deve ser abandonada imediatamente). Uma questão crítica é o quanto um agente deve se comprometer às suas intenções. Ou seja, por quanto tempo um agente deve persistir? Em quais circunstâncias uma intenção deve ser abandonada?

O mecanismo que um agente usa para determinar quando e como abandonar intenções é conhecido como *estratégia de comprometimento*. As três estratégias de comprometimento a seguir são discutidas freqüentemente na literatura de agentes inteligentes (Rao91):

- Comprometimento cego
  Um agente comprometido cegamente continuará a manter uma intenção até que ele acredite que a intenção tenha sido de fato atingida. O comprometimento cego também é chamado de comprometimento fanático.
- Comprometimento cabeça-dura
  Um agente cabeça-dura continuará a manter uma intenção até que ele acredite que a intenção tenha sido atingida ou que não seja mais possível atingi-la.
- Comprometimento cabeça-aberta
  Um agente cabeça-aberta tem as mesmas características do cabeça-dura,

PUC-Rio - Certificação Digital Nº 0321119/CB

com a diferença de que pode abandonar uma intenção se a motivação que levou à sua adoção não valer mais (Bor06).

Pode-se notar que um agente tem comprometimento com os fins (o que ele deseja fazer) e com os meios (como ele deseja fazer). Até agora, o *loop* de controle do agente é excessivamente comprometido tanto com os meios quanto com os fins. Ele nunca pára para reconsiderar as suas intenções e permanece comprometido aos planos até que ele tenha sido executado completamente. Mostra-se agora como esse *loop* de controle básico pode ser refinado de várias maneiras de forma a se obter diferentes tipos de comprometimento.

A primeira modificação a ser feita é permitir ao agente o replanejamento se um plano der errado. Isso reduz o comprometimento que um agente tem com os meios para atingir às suas intenções. O *loop* de controle revisado é mostrado na figura 3.3

1. $B := B_0$
2. $I := I_0$
3. $enquanto\ verdadeiro\ faca$
4. $\quad pegue\ a\ proxima\ percepcao\ p$
5. $\quad B\ := frcr(B, P)$
6. $\quad D\ := opcoes(B, I)$
7. $\quad I\ := filtro(B, D, I)$
8. $\quad \pi\ := plano(B, I)$
9. $\quad enquanto\ \neg vazio(\pi)\ faca$
10. $\quad\quad \alpha = cab(\pi)$
11. $\quad\quad execucao(\alpha)$
12. $\quad\quad \pi := resto(\pi)$
13. $\quad\quad pegue\ a\ proxima\ percepcao\ p$
14. $\quad\quad B\ := frcr(B, P)$
15. $\quad\quad se\ \neg consis(\pi, I, B)\ entao\ \pi := plano(B, I)\ fim_se$
16. $\quad fim_enquanto$
17. $fim_enquanto$

Figura 3.3: Introduzindo reatividade: o agente procurará um novo plano quando ele acreditar que um plano falhou por qualquer razão

Usando esse *loop*, o agente irá se comprometer a um plano para atingir suas intenções somente enquanto ele acreditar que o plano é consistente com suas crenças sobre o estado atual do seu ambiente. Ou seja, enquanto a pré-condição do plano for consistente com as crenças do agente. Se ele em algum momento determinar que seu plano não é mais apropriado para atingir a intenção atual, ele procurará um plano alternativo. Sabendo-se que suas crenças são atualizadas toda a vez em que ele executa uma ação, isso implica pelo menos algum grau de reatividade.

PUC-Rio - Certificação Digital Nº 0321119/CB

Deve-se notar que, em lógica clássica de primeira ordem, a função $consis()$ é semi-decidível. Portanto, se as crenças, as intenções e a pré-condição e a pós-condição dos planos forem descritas em sentenças de primeira ordem, é possível que a função $concis()$ entre em *loop* infinito se o plano não for consistente com o que o agente acredita e pretende atingir (Mac78, Pap93, End02, Car00). Mais ainda, mesmo para lógicas decidíveis, ela é em geral ineficiente, pois o processo de decisão para essas lógicas costuma ser NP-completo ou PSPACE-completo (Mac78, Pap93, Cor90, Aho95, Cla99). Logo, para descrever o ambiente e os estados mentais de um agente, deve-se procurar uma lógica ou um fragmento lógico que diminua a complexidade de resolução do problema para tornar essa abordagem de agentes viável.

Essa versão do *loop* de controle claramente dá mais atenção aos estados do ambiente do que a anterior, porém ainda é excessivamente comprometida às intenções. Isso acontece porque, apesar de o agente replanejar se em algum momento acreditar que o plano não é mais apropriado, ele nunca pára para considerar se as suas intenções são apropriadas ou não. Essa consideração motiva a nova versão do *loop* de controle, na qual o agente explicitamente pára para determinar se as suas intenções já tiveram sucesso ou se são impossíveis. Diz-se $sucesso(I, B)$ para indicar que dadas as crenças B, as intenções I podem ser consideradas como satisfeitas. De maneira similar, diz-se $impossivel(I, B)$ para indicar que as intenções I são impossíveis de serem atingidas com as crenças B. O *loop* de controle é ilustrado na figura 3.4.

1. $B := B_0$
2. $I := I_0$
3. $enquanto\ verdadeiro\ faca$
4. $\quad pegue\ a\ proxima\ percepcao\ p$
5. $\quad B := frcr(B, P)$
6. $\quad D := opcoes(B, I)$
7. $\quad I := filtro(B, D, I)$
8. $\quad \pi := plano(B, I)$
9. $\quad enquanto\ \neg vazio(\pi) \wedge \neg sucesso(I, B) \wedge \neg impossivel(I, B)\ faca$
10. $\quad\quad \alpha = cab(\pi)$
11. $\quad\quad execucao(\alpha)$
12. $\quad\quad \pi := resto(\pi)$
13. $\quad\quad pegue\ a\ proxima\ percepcao\ p$
14. $\quad\quad B := frcr(B, P)$
15. $\quad\quad se\ \neg consis(\pi, I, B)\ entao\ \pi := plano(B, I)\ fim_se$
16. $\quad fim_enquanto$
17. $fim_enquanto$

Figura 3.4: Descartando intenções quando elas forem impossíveis ou já tiverem sido atingidas

PUC-Rio - Certificação Digital Nº 0321119/CB

É fácil ver que esse *loop* de controle revisado implementa o comprometimento cabeça-dura. A modificação necessária para implementar o comprometimento cabeça-aberta é trivial. É necessário apenas acrescentar o predicado $motivado(I, B)$, que indica que o agente ainda está motivado para alcançar a intenção I dadas as crenças B. Isso é ilustrado na figura 3.5.

1. $B := B_0$
2. $I := I_0$
3. $enquanto\ verdadeiro\ faca$
4. $\quad pegue\ a\ proxima\ percepcao\ p$
5. $\quad B := frcr(B, P)$
6. $\quad D := opcoes(B, I)$
7. $\quad I := filtro(B, D, I)$
8. $\quad \pi := plano(B, I)$
9. $\quad enquanto\ \neg vazio(\pi) \wedge \neg sucesso(I, B) \wedge \neg impossivel(I, B)$
10. $\qquad \wedge motivado(I, B)\ faca$
11. $\qquad \alpha = cab(\pi)$
12. $\qquad execucao(\alpha)$
13. $\qquad \pi := resto(\pi)$
14. $\qquad pegue\ a\ proxima\ percepcao\ p$
15. $\qquad B := frcr(B, P)$
16. $\qquad se\ \neg consis(\pi, I, B)\ entao\ \pi := plano(B, I)\ fim_se$
17. $\quad fim_enquanto$
18. $fim_enquanto$

Figura 3.5: Descartando intenções quando o agente não estiver mais motivado para atingi-las

É preciso enfatizar que não há uma estratégia de comprometimento ideal. Diferentes circunstâncias necessitam de diferentes estratégias de comprometimento.

3.4 Reconsiderando a Intenção

No último algoritmo apresentado na seção anterior, um agente irá reconsiderar as suas intenções uma vez a cada vez que executar o *loop* externo. Isso implica que ele irá reconsiderar as suas intenções quando uma das quatro condições a seguir for verdadeira:

- Ele executou completamente um plano para atingir as suas intenções atuais;
- Ele acredita que atingiu as suas intenções atuais;
- Ele acredita que as suas intenções atuais não são mais possíveis, ou
- Ele não está mais motivado para atingir as intenções atuais.

PUC-Rio - Certificação Digital Nº 0321119/CB

Embora isso garanta que o agente não seja comprometido nem demais nem de menos às suas intenções, há um limite na maneira pela qual permite que um agente reconsidere a sua intenção. O problema principal é que ele não permite a um agente explorar uma outra opção inesperada.

Uma primeira tentativa de modificar o *loop* de controle do agente envolveria reconsiderar as intenções toda a vez que o agente executar o *loop* interno da figura 3.5, como é possível ver na figura 3.6. Tal agente é cauteloso, pois sempre pára para reconsiderar suas intenções antes de realizar uma ação.

1. $B := B_0$
2. $I := I_0$
3. $enquanto\ verdadeiro\ faca$
4. $\quad pegue\ a\ proxima\ percepcao\ p$
5. $\quad B := frcr(B, P)$
6. $\quad D := opcoes(B, I)$
7. $\quad I := filtro(B, D, I)$
8. $\quad \pi := plano(B, I)$
9. $\quad enquanto\ \neg vazio(\pi) \wedge \neg sucesso(I, B) \wedge \neg impossivel(I, B)$
10. $\qquad \wedge\ motivado(I, B)\ faca$
11. $\qquad \alpha = cab(\pi)$
12. $\qquad execucao(\alpha)$
13. $\qquad \pi := resto(\pi)$
14. $\qquad pegue\ a\ proxima\ percepcao\ p$
15. $\qquad B := frcr(B, P)$
16. $\qquad D := opcoes(B, I)$
17. $\qquad I := filtro(B, D, I)$
18. $\qquad se\ \neg consis(\pi, I, B)\ entao\ \pi := plano(B, I)\ fim_se$
19. $\quad fim_enquanto$
20. $fim_enquanto$

Figura 3.6: Um agente cauteloso, que pára para reconsiderar as intenções antes de realizar qualquer ação

Se a geração de opções e o processo de filtragem fossem computacionalmente baratos, essa seria uma estratégia aceitável. Infelizmente, sabe-se que isso leva uma grande quantidade de tempo. Enquanto o agente está gerando opções e filtrando-as, o ambiente no qual o agente está trabalhando está mudando, possivelmente fazendo com que as novas intenções formadas sejam irrelevantes.

Para que ele não reconsidere suas intenções excessivamente, pode-se modificá-lo para incorporar um componente de controle em um meta-nível explicitamente. A idéia é ter uma função booleana $reconsiderar()$, tal que $reconsiderar(I, B)$ seja verdadeira somente no caso em que for apropriado para um agente com crenças B e intenções I reconsiderar suas intenções. O

PUC-Rio - Certificação Digital Nº 0321119/CB

loop de controle do agente incorporando a função *reconsiderar* está exibido na figura 3.7.

```
1.    B := B_0
2.    I := I_0
3.    enquanto verdadeiro faca
4.        pegue a proxima percepcao p
5.        B := frcr(B, P)
6.        D := opcoes(B, I)
7.        I := filtro(B, D, I)
8.        π := plano(B, I)
9.        enquanto ¬vazio(π) ∧ ¬sucesso(I, B) ∧ ¬impossivel(I, B)
10.                ∧ motivado(I, B) faca
11.            α = cab(π)
12.            execucao(α)
13.            π := resto(π)
14.            pegue a proxima percepcao p
15.            B := frcr(B, P)
16.            se reconsiderar(I, B) entao
17.                D := opcoes(B, I)
18.                I := filtro(B, D, I)
19.            fim_se
20.            se ¬consis(π, I, B) entao π := plano(B, I) fim_se
21.        fim_enquanto
22.    fim_enquanto
```

Figura 3.7: Um agente que tenta atingir um balanço entre teimosia e cautela

Nessa versão do *loop* de controle do agente, a tarefa de decidir quando gastar tempo atualizando os desejos e as intenções cabe à função *reconsiderar*( ). É interessante considerar as circunstâncias sob as quais pode-se dizer que essa função se comporta de maneira ótima. Suponha-se que as funções geradoras de planos e intenções sejam perfeitas: a de intenções sempre escolhe as melhores intenções (o que é definido para a aplicação em questão), e a de planos sempre produz um plano apropriado. Adicionalmente, suponha-se que o tempo gasto sempre tenha um custo (o agente não se beneficia quando não faz nada). Logo, não é difícil ver que a função *reconsiderar*( ) vai se comportar de maneira ótima se, e somente se, quando ele mudar suas opções, também mudar suas intenções (Woo99a). Se o agente atualizar seus desejos mas não as suas intenções, o agente terá perdido tempo na atualização. De maneira similar, se um agente devesse ter mudado de intenções, mas não o fez, então o esforço empreendido na tentativa de atingir suas intenções terá sido em vão.

As possíveis interações entre o controle em meta-nível e a geração de opções está resumida na tabela 3.1

PUC-Rio - Certificação Digital Nº 0321119/CB

Situação	Atualizou os desejos	Atualizou as intenções	Teria mudado as intenções	Otimalidade de *reconsiderar*( )
1	Não	–	Não	Sim
2	Não	–	Sim	Não
3	Sim	Não	–	Não
4	Sim	Sim	–	Sim

Tabela 3.1: Situações no raciocínio prático

- Na situação (1), o agente escolheu não atualizar os desejos e, como conseqüência, não atualizou as intenções. Mais ainda, se tivesse escolhido fazê-lo, não teria modificado as intenções. Nessa situação, a função *reconsiderar*( ) é ótima.
- Na situação (2), o agente escolheu não atualizar os desejos, mas, se o tivesse feito, teria modificado as intenções. Nessa situação, a função *reconsiderar*( ) não é ótima.
- Na situação (3), o agente escolheu atualizar os desejos, porém não modificou as intenções. Nessa situação, a função *reconsiderar*( ) não é ótima.
- Na situação (4), o agente escolheu atualizar os desejos e modificou as intenções. Nessa situação, a função *reconsiderar*( ) é ótima.

Deve-se notar que há uma importante hipótese implícita nessa discussão: o custo da execução da função *reconsiderar*( ) é muito menor que o custo do processo de atualização dos desejos. Se não, a função *reconsiderar*( ) poderia simplesmente utilizar o processo de atualização de opções como um oráculo, executando-o como uma sub-rotina e escolhendo atualizar os desejos apenas nos casos em que as intenções fossem alteradas. Ou seja, há o problema já apontado anteriormente sobre a questão da reconsideração de intenções.

PUC-Rio - Certificação Digital Nº 0321119/CB