Agentes Inteligentes

Capítulo 2

Sumário

- Agentes e ambientes
- Racionalidade
- PEAS: caracterização de um agente
- Tipos de ambientes
- Tipos de agentes

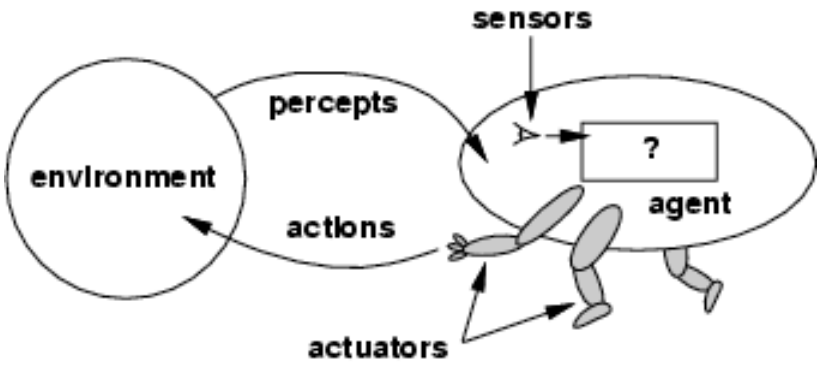
Agentes

- Um agente é tudo o que é capaz de captar/ perceber o ambiente onde se encontra através de sensores e actua nesse ambiente através de actuadores

- Agente humano
  - Sensores: olhos, orelhas e outros órgãos
  - Actuadores: mãos, pernas, boca e outras partes do corpo

- Agente robótico
  - Sensores: câmaras e infravermelhos
  - Actuadores: partes motoras

Agentes e ambientes

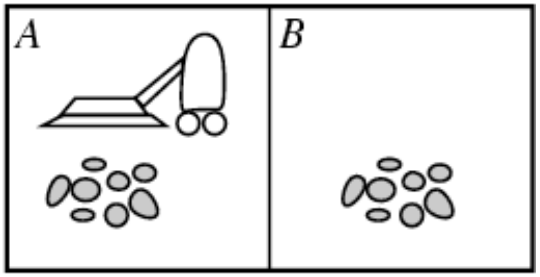


- A função agente mapeia uma sequência de percepções em acções:

$$[f: \mathcal{P}^{\star} \rightarrow \mathcal{A}]$$

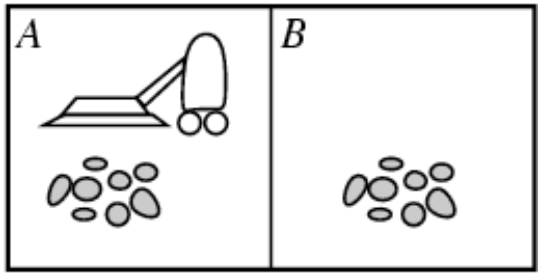
- O programa agente é executado numa plataforma para produzir *f*
- agente = plataforma + programa

Mundo do aspirador



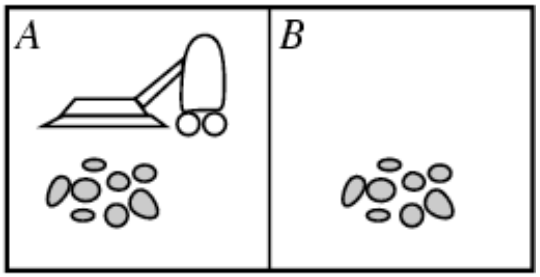
- Percepções [localização, conteúdo]
  - Localização: A, B
  - Conteúdo: Limpo, Sujo
- Acções: Esquerda, Direita, Aspirar

Agente aspirador



Sequência de Percepções	Acção
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo],[A, Limpo]	Direita
[A, Limpo],[A, Sujo]	Aspirar
…	…

Agente aspirador



Função AgenteAspirador ([posição, estado])

devolve acção

Se estado = Sujo **então devolve** Aspirar

Senão se posição = A **então devolve** Direita

Senão se posição = B **então devolve** Esquerda

Agentes racionais

- Um agente deve procurar fazer “o que está certo", baseado nas suas percepções e nas acções que pode tomar
- A acção certa é aquela que dá maior expectativa de sucesso ao agente

- Medida de desempenho: critério objectivo que mede o sucesso do comportamento do agente
- Por exº, medida de desempenho do agente aspirador pode ser a sujidade aspirada, tempo utilizado, electricidade consumida, ruído gerado, etc.

Agentes racionais

- Agente Racional: Por cada sequência de percepções possível, um agente racional deve seleccionar uma acção que é suposto maximizar a sua medida de desempenho, dada a informação disponibilizada pela sequência de percepções e eventualmente pelo conhecimento que o agente possui.

Agentes racionais

- Racionalidade ≠ omnisciência
  - Percepções podem não disponibilizar conhecimento que é importante
- Racionalidade ≠ clarividência
  - Resultado de uma acção pode não estar de acordo com o esperado
- Logo, racionalidade ≠ sucesso

- Racionalidade ⇨ exploração, aprendizagem, autonomia

Agente autónomo

- Um agente é autónomo se o seu conhecimento for determinado apenas pela sua experiência (com capacidade de aprender e adaptar-se)
  - Agentes podem tomar acções para obter informações úteis (recolha de informação, exploração)

Caracterização de um agente

- PEAS
  - Performance (desempenho)
  - Environment (ambiente)
  - Actuators (actuadores)
  - Sensors (sensores)

PEAS: agente profª de Inglês

- Desempenho
  - Notas dos alunos no teste
- Ambiente
  - Conjunto de alunos
- Actuadores
  - Monitor: exercícios, sugestões, correcções
- Sensores
  - Teclado

PEAS: agente médico

- Desempenho
  - Saúde do paciente, custos
- Ambiente
  - Paciente, hospital, funcionários
- Actuadores
  - Monitor: questões, testes, diagnósticos, tratamentos
- Sensores
  - Teclado: sintomas, respostas

PEAS: agente taxista

- Desempenho
  - Segurança, destino, lucros, legalidade, conforto
- Ambiente
  - Clientes, estradas, trânsito, transeuntes, tempo
- Actuadores
  - Volante, acelerador, travão, buzina, pisca
- Sensores
  - GPS, conta km, velocímetro, nível do depósito, temperatura do óleo

Sumário

- Agentes e ambientes
- Racionalidade
- PEAS: caracterização de um agente
- Tipos de ambientes
- Tipos de agentes

Tipos de Ambientes

- Completamente observável (vs. parcialmente observável): Os sensores do agente dão acesso ao estado completo do ambiente em cada instante de tempo, pelo que não é necessário manter um estado interno sobre o mundo.
- Determinístico (vs. estocástico): O estado seguinte do ambiente é determinado somente em função do estado actual e da acção executada pelo agente – não há incerteza para o agente. (Se o ambiente é sempre determinístico excepto para as acções de outros agentes, então o ambiente é estratégico)

Tipos de Ambientes

	Xadrez com relógio	Análise de Imagem	Condutor de táxi
Completamente observável?	Sim	Sim	Não
Determinístico?	Estratégico	Sim	Não

Tipos de Ambientes

- Episódico (vs. sequencial): A experiência do agente está dividida em episódios atómicos (em que cada episódio consiste em percepção +acção do agente) e a escolha de cada acção em cada episódio depende apenas do próprio episódio.
- Estático (vs. dinâmico): o ambiente não é alterado enquanto o agente decide que acção vai tomar. (O ambiente é semi-dinâmico se o ambiente permanece inalterado com a passagem do tempo mas a qualidade do desempenho do agente é alterada)

Tipos de Ambientes

	Xadrez com relógio	Análise de Imagem	Condutor de táxi
Episódico?	Não	Sim	Não
Estático?	Semi	Semi	Não

Tipos de Ambientes

- Discreto (vs. contínuo): O agente tem um número limitado de percepções e acções distintas que estão claramente definidas.
- Agente único (vs. multi-agente): Só existe um agente no ambiente.

Tipos de Ambientes

	Xadrez com relógio	Análise de Imagem	Condutor de táxi
Discreto?	Sim	Não	Não
Agente único?	Não	Sim	Não

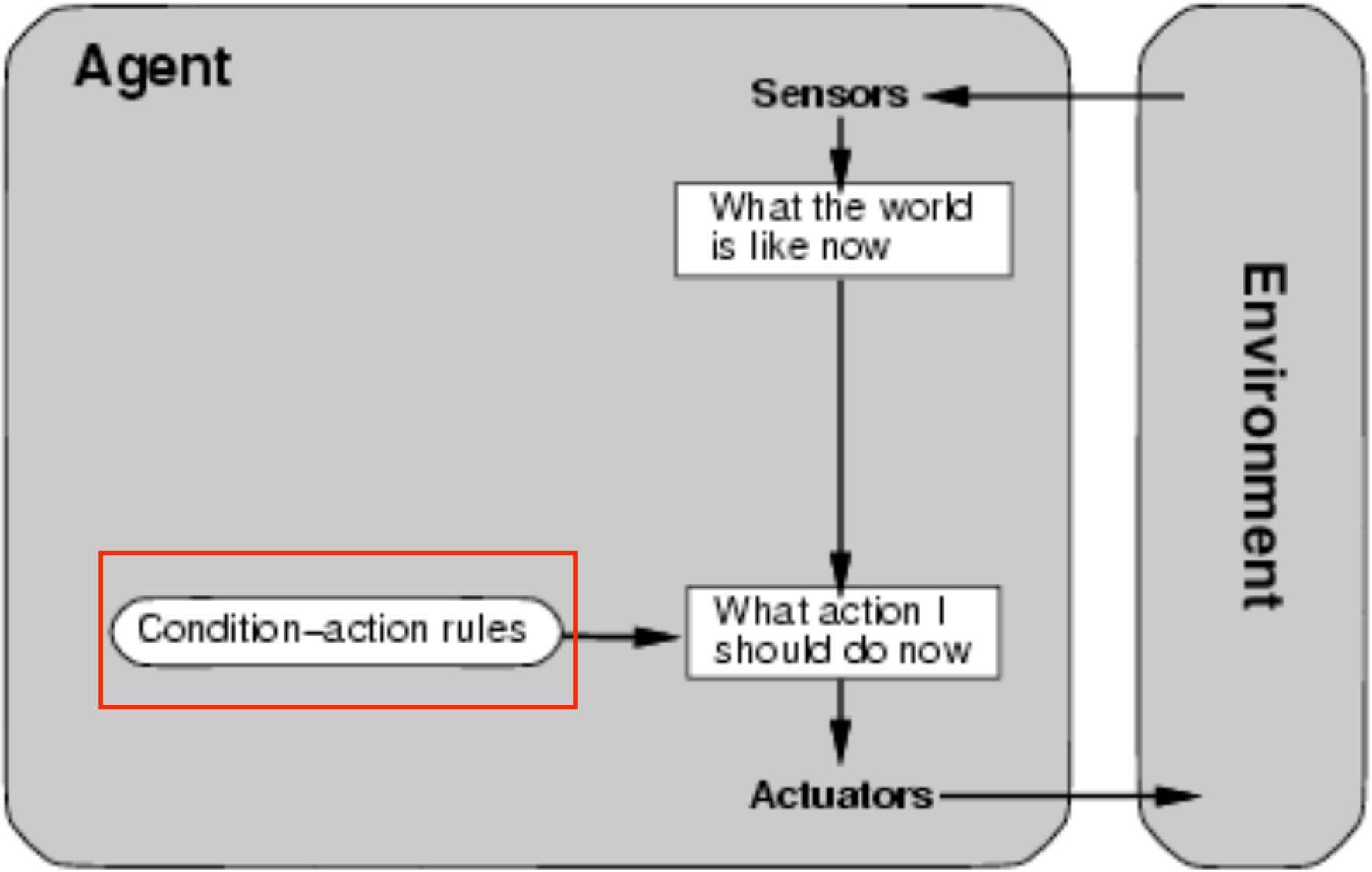
Tipos de Ambientes

- O tipo de ambiente determina o tipo de agente
- O mundo real é:
  - Parcialmente observável (vs. completamente observável)
  - Estocástico (vs. determinístico)
  - Sequencial (vs. episódico)
  - Dinâmico (vs. estático)
  - Contínuo (vs. discreto)
  - Multi-agente (vs. agente único)

Tipos de Agentes

- Agentes de reflexos simples
- Agentes de reflexos baseados em modelos
- Agentes baseados em objectivos
- Agentes baseados em utilidade
- Agentes com aprendizagem

Agentes de reflexos simples



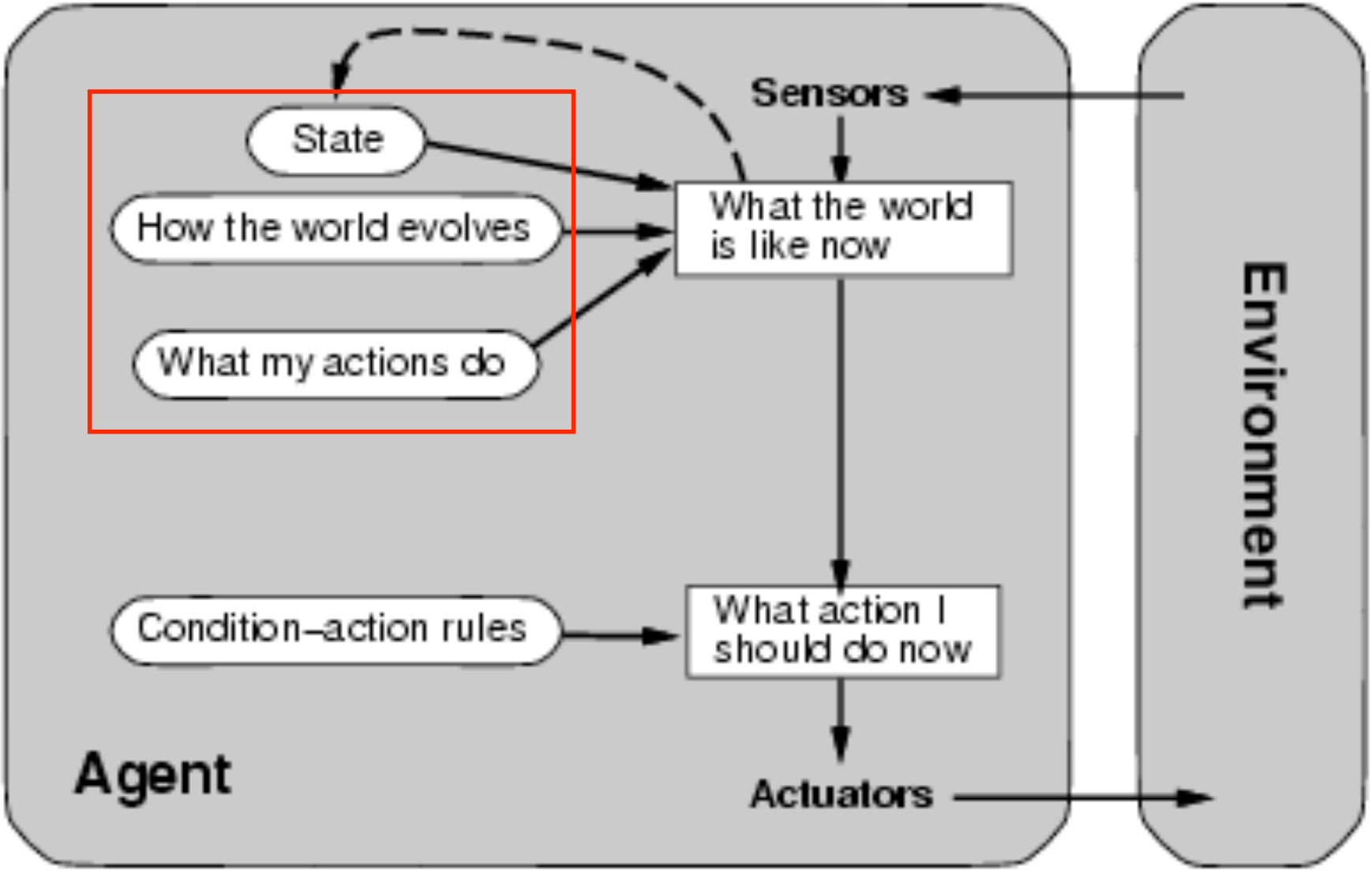
Agentes de reflexos simples

Função AgenteReflexosSimples (*percepção*)
devolve *acção*

Estático: *regras* (conjº de regras condição-acção)

estado ← InterpretaInput(*percepção*)
regra ← EmparelhaRegra(*estado,regras*)
acção ← RegraAcção[*regra*]
devolve *acção*

Agentes de reflexos baseado em modelos



Agentes de reflexos baseado em modelos

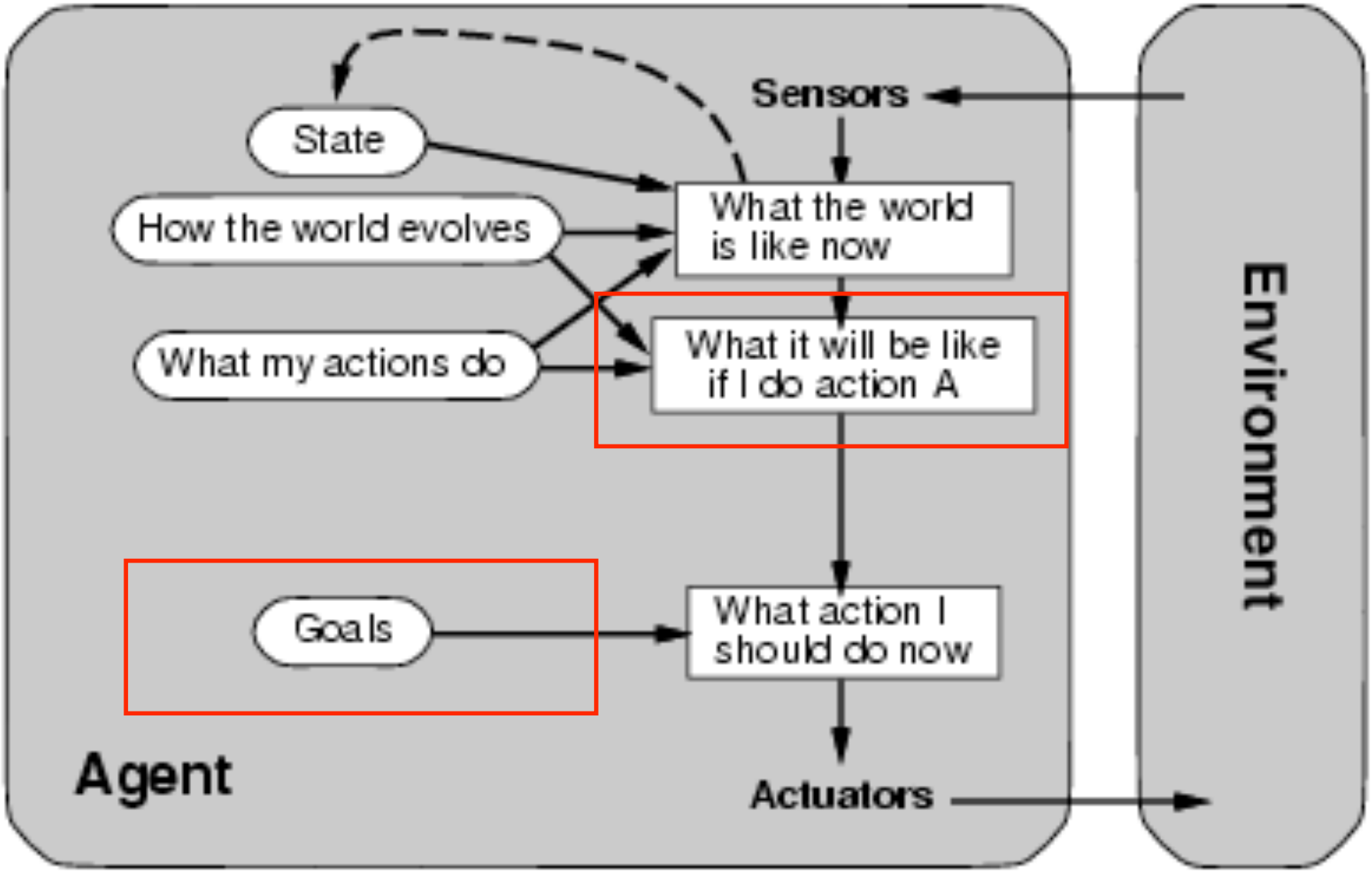
Função AgenteReflexosBaseadoEmModelos (*percepção*)
devolve *acção*

Estático: *estado* (descrição do estado do mundo)
*regras (*conjº de regras condição-acção)
acção (a acção mais recente)

estado ← ActualizaEstado(*estado,acção,percepção*)
regra ← EmparelhaRegra(*estado,regras*)
acção ← RegraAcção[*regra*]
devolve *acção*

Também chamados agentes de reflexos simples com estado interno.

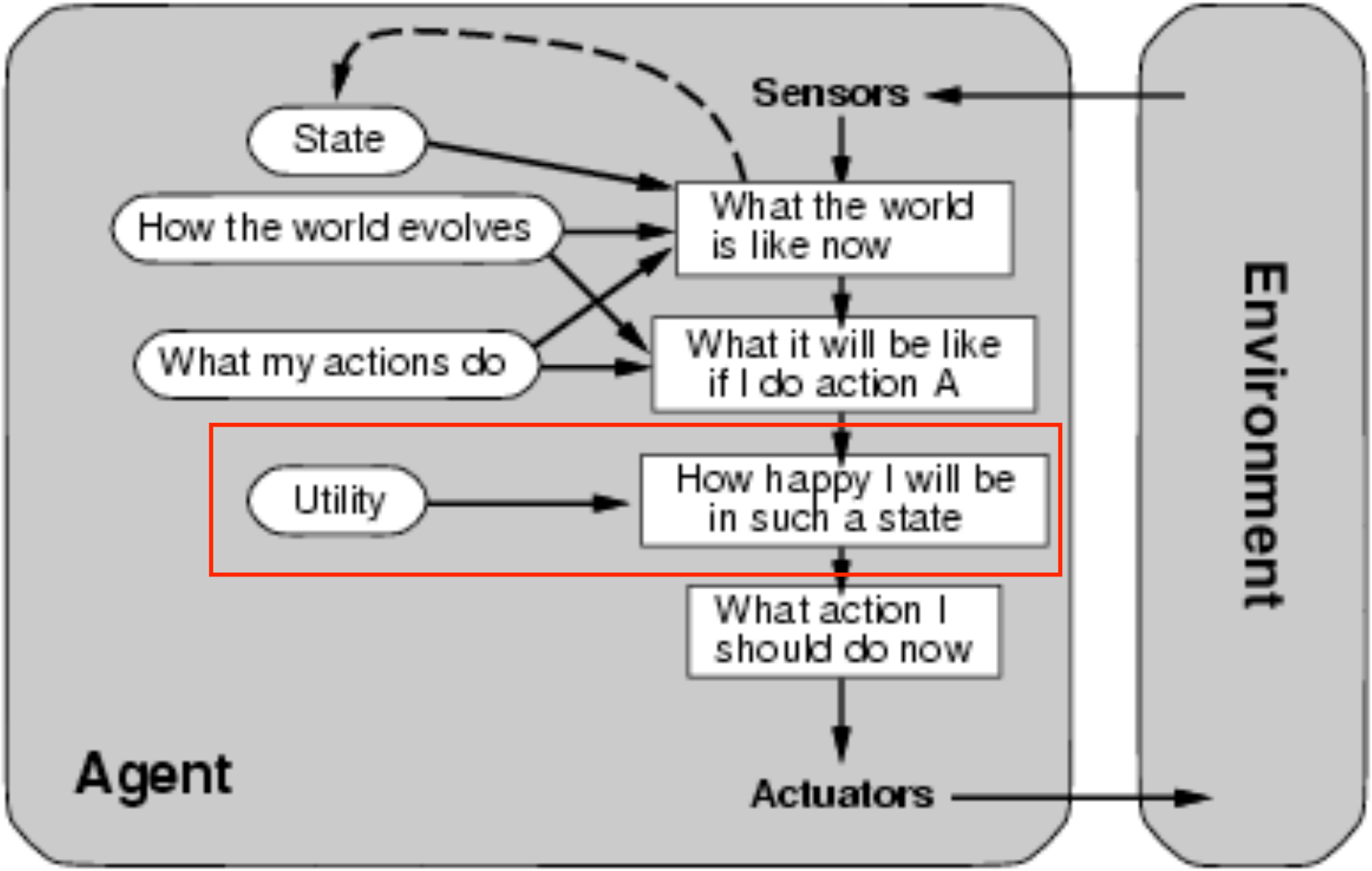
Agentes baseados em objectivos



Agente baseado em objectivos

- Agente tem um (ou mais) objectivo(s)
- Por exº, considere-se um agente taxista que cujo objectivo é chegar a um destino
  - Chegando a um cruzamento, o agente decide avançar, virar à direita ou virar à esquerda em função do objectivo
- Acrescenta a um agente de reflexos simples considerações sobre o futuro, a fim de alcançar os objectivos

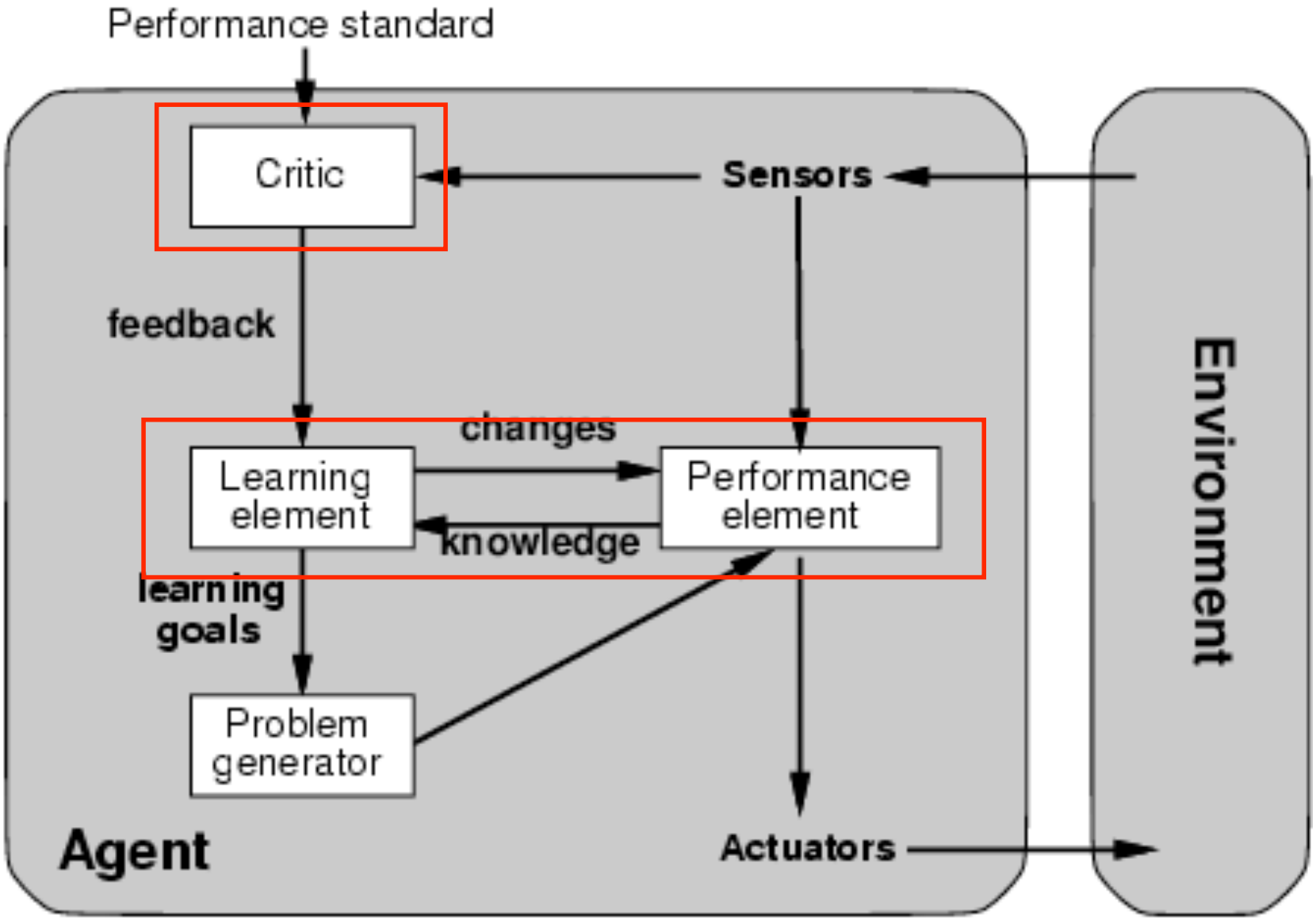
Agentes baseados em utilidade



Agente baseado em utilidade

- Agente tem uma função de utilidade
- A função de utilidade permite estabelecer preferências entre sequências de estados que permitem atingir os mesmos objectivos
- Por exº, considere-se um agente taxista que pretende chegar a um destino
  - A função de utilidade permite distinguir as diferentes formas de chegar ao destino, em função do tempo, da despesa, da segurança, …

Agentes com aprendizagem



Agente com aprendizagem

- Correspondem à ideia de máquina inteligente caracterizada por Turing (1950)
  - Agente pode actuar num mundo inicialmente desconhecido
- Elemento de aprendizagem
  - Torna o agente mais eficiente ao longo do tempo
  - Usa feedback da crítica que avalia actuação do agente de acordo com o desempenho espectável
- Elemento de desempenho
  - Responsável por seleccionar as acções do agente
- Elemento de geração de problemas
  - Sugere acções que podem trazer informação útil

Sumário

- Exemplos de agentes (ex^{os} de exames)
  - Reflexos simples
  - Reflexos simples baseados em modelos (também chamados com estado interno)

Exemplos de Agentes

- Enunciado
  - Caracterização do agente
- Exercício
  - Tipo de agente
    - Reflexos simples
    - Reflexos simples com estado interno
  - Definição da percepção
  - Definição da função agente

Agente Venda Bilhetes Metro

- Considere um agente que faz a venda de bilhetes de metro. Por cada percepção, o agente recebe o custo do bilhete pretendido e a quantia introduzida. A acção devolvida poderá ser **QUANTIA_CERTA** ou o **valor** correspondente ao troco ou à quantia introduzida (no caso desta ser insuficiente).

Agente Venda Bilhetes Metro

- Percepção

```
(defstruct percepcao
    custo quantia)
```

- Acções
  - QUANTIA_CERTA
  - Valor troco
  - Valor quantia

- Agente de reflexos simples

Agente Venda Bilhetes Metro

```
(defun agente (p)
 (let ((custo (percepcao-custo p))
       (quantia (percepcao-quantia p)))
  (cond ((= custo quantia) 'QUANTIA_CERTA)
        ((< custo quantia)(- quantia custo))
        (t quantia))))
```

Agente Venda Bilhetes Metro

```
> (setf p1 (make-percepcao :custo 0.70 :quantia 1))
#S(PERCEPCAO :CUSTO 0.7 :QUANTIA 1)
> (setf p2 (make-percepcao :custo 0.70 :quantia
  0.5))
#S(PERCEPCAO :CUSTO 0.7 :QUANTIA 0.5)
> (setf p3 (make-percepcao :custo 0.70 :quantia
  0.7))
#S(PERCEPCAO :CUSTO 0.7 :QUANTIA 0.7)

> (agente p1)
0.3
> (agente p2)
0.5
> (agente p3)
QUANTIA_CERTA
```

Agente Empacota Brinquedos

- Considere um agente associado a um sensor numa fábrica de brinquedos. O sensor detecta quando é que um brinquedo passou pela passadeira. O agente recebe informação do sensor e quando tiverem passado 10 brinquedos emite a acção de **EMPACOTAR**.

Agente Empacota Brinquedos

- Percepção

```
(defstruct percepcao
    (passa-brinquedo nil))
```

- Acções
  - EMPACOTAR
  - NAO_FAZ_NADA

- Agente de reflexos simples com estado interno (nº de brinquedos empacotados)

Agente Empacota Brinquedos

```
(defun cria-agente ()
 (let ((n-brinquedos 0))
  #'(lambda (p)
      (when (percepcao-passa-brinquedo p)
          (incf n-brinquedos))
      (cond ((= n-brinquedos 10)
          (setf n-brinquedos 0) 'EMPACOTAR)
          (t 'NAO_FAZ_NADA)))))
```

Agente Empacota Brinquedos

```
> (setf p1 (make-percepcao))
#S(PERCEPCAO :PASSA_BRINQUEDO NIL)
> (setf p2 (make-percepcao :passa_brinquedo t))
#S(PERCEPCAO :PASSA_BRINQUEDO T)

> (setf ag-emp (cria-agente))
#(FUNCTION :LAMBDA (p) …)
> (funcall ag-emp p1)
NAO_FAZ_NADA
> (funcall ag-emp p2)
NAO_FAZ_NADA
…
> (funcall ag-emp p2)
EMPACOTAR
```

Agente Termóstato

- Considere um agente termóstato. Por cada percepção, o agente recebe o valor da temperatura ambiente pretendida. A acção devolvida poderá ser **AQUECER**, **ARREFECER** ou **MANTER** em função da temperatura ambiente actual.

Agente Termóstato

- Percepção

```
(defstruct percepcao
    temperatura)
```

- Acções
  - AQUECER
  - ARREFECER
  - MANTER

- Agente de reflexos simples com estado interno (temperatura ambiente actual)

Agente Termóstato

```
(defun cria-agente (temp)
 #'(lambda (p)
   (let ((nova-temp (percepcao-temperatura p)))
     (cond ((= nova-temp temp) 'MANTER)
           ((< nova-temp temp)
            (setf temp nova-temp) 'ARREFECER)
           (t (setf temp nova-temp)
              'AQUECER)))))
```

Agente Parque de Estacionamento

- Considere um agente que faz a gestão de um parque de estacionamento. O sensor detecta quando entra um carro e quando sai um carro. A acção devolvida poderá ser **LEVANTAR_ENTRADA, LEVANTAR_SAIDA, LEVANTAR_ENTRADA_SAIDA** (entra um carro e sai outro ao mesmo tempo), **ESPERAR** (não há lugar) ou **NAO_FAZ_NADA**.

Agente Parque de Estacionamento

- Percepção

```
(defstruct percepcao
 entra-carro sai-carro)
```

- Acções
  - LEVANTAR_ENTRADA
  - LEVANTAR_SAIDA
  - LEVANTAR_ENTRADA_SAIDA
  - ESPERAR
  - NAO_FAZ_NADA
- Agente de reflexos simples com estado interno (nº de lugares livres)

Agente Parque de Estacionamento

```
(defun cria-agente (n)
 #'(lambda (p)
  (let ((entra (percepcao-entra-carro p))
        (sai (percepcao-sai-carro p)))
   (cond ((and entra sai)
          'LEVANTAR_ENTRADA_SAIDA)
         (sai (incf n) 'LEVANTAR_SAIDA)
         (entra (if (zerop n)
                    'ESPERAR
                    (progn
                      (decf n)
                      'LEVANTAR_ENTRADA)))
         (t NAO_FAZ_NADA)))))
```

Agente Elevador

- Considere um agente que faz a gestão de um elevador. O sensor detecta o peso do elevador, se a porta está aberta e qual o piso de destino. A acção devolvida poderá ser **ARRANCAR_R** (peso < 200Kg e distância >= 2 pisos), **ARRANCAR_N** (peso < 450Kg), **ARRANCAR_L** ou **NAO_FAZ_NADA** (porta aberta ou peso superior a 650Kg).

Agente Elevador

- Percepção

```
(defstruct percepcao
 peso aberta piso)
```

- Acções
  - ARRANCAR_R
  - ARRANCAR_N
  - ARRANCAR_L
  - NAO_FAZ_NADA

- Agente de reflexos simples com estado interno (piso actual)

Agente Elevador

```
(defun cria-agente (piso)
 #'(lambda (p)
  (let ((peso (percepcao-peso p))
        (aberta (percepcao-aberta p))
        (p-piso (percepcao-piso p)))
   (if (or aberta (> peso 650))
       'NAO_FAZ_NADA
        (let ((dist (abs (- piso p-piso))))
         (setf piso p-piso)
         (cond ((and (< peso 200)(>= dist 2))
                                        'ARRANCAR_R)
               ((< peso 450) 'ARRANCAR_N)
               (t 'ARRANCAR_L)))))))
```