第一部分　响 应 机 器

当你思考这一领域的问题时，会不自觉地不知所措。思考中的你正处于实际生活中的某一周里，那正是阳光明媚的时候。一旦深入这一游戏，你会恍然大悟：你只需对你所看到的作出响应。

——Albert Lewis,cornerback of the Oakland Raiders,"……Sam Farmer 引自《San Jose Mercury News》, 1D页，1996年8月30日。

第2章　刺激响应agent

2.1　感知和动作

在本章中，我将介绍不具有内部状态而仅对其所处环境的即刻刺激有所反应的机器。我们称之为刺激响应（*stimulus-response*, S-R）agent。各种机器人应运而生，它们能通过电机对传感器即刻输入的十分简单的功能反应而展示相当有趣的行为。这种机器人中最早的例子是Grey Walter的*Machina speculatrix* ——一个有轮子、配有电机、光电管和两个真空管的装置[Walter 1953]，能够朝光度适中的地方移动而避免强光。Braitenberg也描述过类似的机器[Braitenberg 1984]。

用一个图例来开始我们的讨论。先来看这样一个在二维网格空间世界里的机器人，如图 2-1所示。这一机器人的世界有完整的边界线，可能还包括如图所示的其他庞大的固定物体，这一世界里没有"稠密空间（*tight space*）"（即物体与边界线之间的距离只有一个单元格）[⊖]，我们设计机器人时可利用这一点。

我们要求这个机器人完成以下动作：走到与一边界或物体毗邻的单元格中，然后沿着它的边界一直走下去。要能够完成这一沿边界运动的行为，机器人必须能够感知一个单元格是否空缺而可以向其移动，并且必须能够做一些基本的动作。

这个机器人能够感知出它周围八个单元格是否空缺。这些传感器输入用二进制变量 s_1、s_2、s_3、s_4、s_5、s_6、s_7和s_8表示。当相应的单元格能被机器人占据时（如图 2-1所示），它们的变量值为0，反之为1。如果机器人处于有 X标记的地方，传感器的输入值（从 s_1开始顺时针计算）为（0，0，0，0，0，0，1，0）。

该机器人能够向与它同行或同列的毗邻的（空缺）单元格移动，共有如下四种动作：

`north`：机器人在网格中向上移动一个单元

`east`：机器人在网格中向右移动一个单元

`south`：机器人在网格中向下移动一个单元

`west`：机器人在网格中向左移动一个单元

⊖ 有关"不存在稠密空间"的具体论述请参阅 www. mkp.com/nils/clarified。

所有这些动作均有其效果，除非该机器人企图向非空缺的单元格移动；如果这样，则此动作无效。

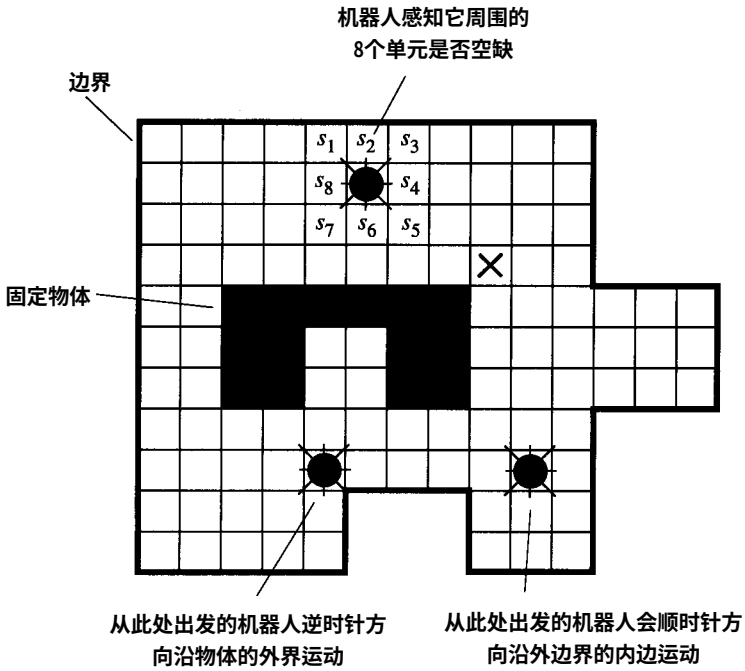


图2-1 一个在二维网格空间中的机器人

给定了机器人适应的某种世界的特性（如图 2-1所示）、机器人完成的任务（沿边界移动）和机器人传感器和电机的功能，设计者的工作就是说明为此任务选择适当动作的传感器输入（示例中表示为s_1，…，s_8）的功能。通常我们把从传感器信号中计算动作的过程分为两个分开的阶段，如图2-2所示。知觉处理阶段产生一个特征向量 X（x_1,… x_i,…,x_n）。动作计算阶段选择一个以特征向量为基础的动作：各特征值既可以是真正的数字（ *numeric feature*, 数字特征），也可以是范畴（*categorical feature* ,范畴特征）（范畴特征的值是名字或特性，譬如：特征值“颜色”可能是“红”、“蓝”或“绿”）。二进制特征这一特殊例子既可视作数字（0，1），也可视作范畴（真，假）。设计者选择特征来将其与机器人的环境特性相联系，而此环境特性又与由此特征描述的状态中机器应做的动作密切相关。

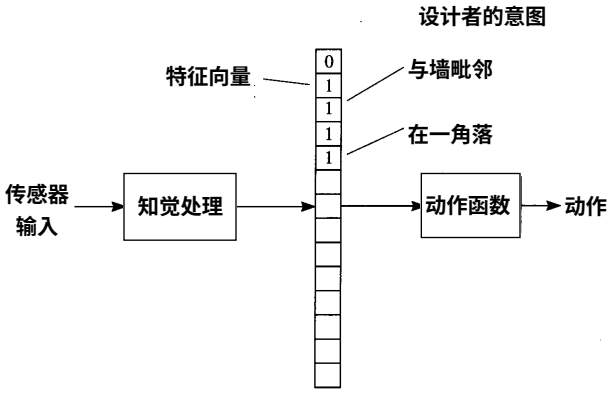


图2-2 知觉和动作部分

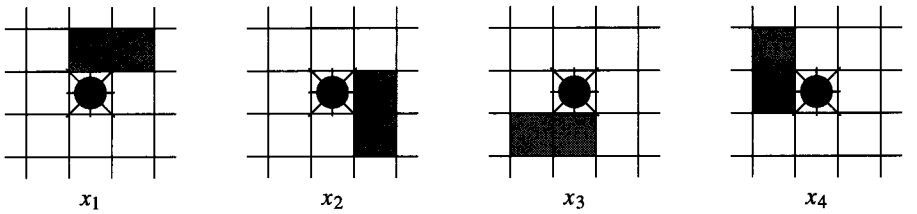
当然，知觉与动作的分界是完全任意的。可以把整个过程要么看作知觉（世界被感知，处于一个应该采取动作“北”的状态），

也可看作动作（根据原始传感器的数据而计算得出，应该采取动作“北”）。通常，分界可使得在完成各项预期任务时重复使用相同的特征。这样，不同的任务可拥有相同的特征向量和不同的动作函数。这些计算机程序对传感器信号中的特征进行的计算通常被视作重复使用库程序——被许多不同的动作计算所需要。如何分界这两个过程是这些机器的设计艺术，对此就无庸赘述了。

完成分界工作后，我们还有两个问题要解决：一是把传感器的数据转换成特征向量；另一个是确定动作函数。在后面的示例中会分别简述这两个问题。

2.1.1 感知

沿边界运动的机器人的传感器输入包括 s_1，…，s_8的值，这些值共有 $2^8 = 256$种组合方式。但在我们的环境中，由于稠密空间的限制，一些组合方式已被排除在外。对当前的任务，刚好有四个对计算适当动作有用的传感器的二进制特征值，分别用 x_1、x_2、x_3和x_4来表示。它们的定义如图2-3所示。例如，$x_1=1$，当且仅当$s_3=1$或$s_2=1$。



在每个图中，当且仅当至少一个阴影单元为非空时，所示的特征值为1

图2-3 沿边界运动的特征

本例中知觉处理过程的计算相对比较简单，而为更复杂的世界以及具备更复杂的传感器和任务的机器人设计这一过程，将极富挑战性。而且在许多实际任务中，知觉处理过程偶尔会发出有关机器人所处环境的错误、模糊或不完全的信息。尽管这样的错误会导致不恰当的动作，但只要不经常发生，这些依任务和环境而定的动作并不会造成太大的损失。在第 6章中，我会再回到知觉这一问题中来。

2.1.2 动作

有了这四个特征，现在我们必须确定它们的函数以便选择适当的沿边界的动作。我们首先提出，如果特征值都不是1（即机器人感知到它周围的单元格全部空缺），则它可向任一个方向移动直至遇到边界。我们先让它向北移动。若至少有一个特征值为 1，沿边界的行动则按以下规则完成：

若$x_1 = 1$且$x_2 = 0$，则向东移；

若$x_2 = 1$且$x_3 = 0$，则向南移；

若$x_3 = 1$且$x_4 = 0$，则向西移；

若$x_4 = 1$且$x_1 = 0$，则向北移。

其中，这些机器人可做出不同动作的条件恰好是特征值的布尔组合，这些特征值本身也是传感器输入的布尔组合。既然一些重要的知觉和动作选择方法涉及到布尔函数，那么在继续讨论实

例之前我们有必要讨论一下它们。

2.1.3 布尔代数

布尔函数f（$x_1, x_2,..., x_n$）表示n元组（每个值为（0,1））与{0，1}集合之间的对应关系。布尔代数是说明布尔函数的一种便捷的表达方法。布尔代数使用“·”、“+”、“-”三个连接符号。两个变量的“与”函数记作$x_1 \cdot x_2$，通常连接符号可以省略，即可以记作$x_1 x_2$。函数$x_1 x_2$值为1，当且仅当x_1和x_2的值均为1；否则，其值为0。两个变量的或记作x_1+x_2，x_1+x_2值为1当且仅当x_1、x_2两者均为1或其中任一个为1；否则值为0。变量x的补(*complement*)或非(*negation*)记作$\bar{x}_1$。$\bar{x}_1$的值为1当且仅当x的值为0；否则其值为0。

以下便是布尔代数所给出的严格定义：

$1 + 1 = 1$，$1 + 0 = 1$，$0 + 0 = 0$

$1 \cdot 1 = 1$，$1 \cdot 0 = 0$，$0 \cdot 0 = 0$

$\bar{1} = 0$，$\bar{0} = 1$

譬如，表达式$\bar{x}_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4 + x_1\bar{x}_1$给出了沿边界运动的机器人应向北移动的情况。在这个例子中，从传感器信号中计算特征值的函数也恰好是布尔函数。如，$x_4 = s_1 + s_8$。其他特征值和动作规则也由相似的函数给出。

有时，布尔函数的参数和值不用1和0表示，而用相应的常数T（真）和F（假）来表示。

连接符“·”和“+”具交换性。即，$x_1x_2=x_2x_1$；$x_1 + x_2 = x_2 + x_1$。它们同时具有结合性，即$x_1(x_2x_3)=(x_1x_2)\ x_3$；$x_1+(x_2+x_3)=(x_1+x_2)+x_3$。这样，我们可去掉括号，书写如下：$x_1\ x_2\ x_3$和$x_1 + x_2 + x_3$。

由单个变量组成的布尔函数，如x_1，称为原子(*atom*)。由单个变量或其补组成的布尔函数，如$\bar{x}_1$，称为文字(*literal*)。

在复合表达式中，先“·”后“+”的顺序不能颠倒。而且，布尔代数遵守 DeMorgan定律（用前面的定义可证明此定律）：

$$\overline{f_1 f_2} = \overline{f_1} + \overline{f_2}$$

$$\overline{f_1 + f_2} = \overline{f_1}\,\overline{f_2}$$

DeMorgan定律可简化布尔函数。如，$x_1\overline{x_2} = (s_2 + s_3)\overline{(s_4 + s_5)} = (s_2 + s_3)\overline{s_4}\,\overline{s_5}.$

另一个重要定律为分配律：

$$f_1(f_2 + f_3) = f_1 f_2 + f_1 f_3$$

2.1.4 布尔函数的类别和形式

布尔函数有多种形式，其中一种重要的形式为：$\lambda_1\lambda_2...\lambda_k$，这里$\lambda_i$为文字，这样书写的函数称为“文字合取式（*conjunction*）”或“单项式(*monomial*)”。这一合取式本身称为一个项式。x_1x_7和$x_1x_2\bar{x}_4$均为项式。项式的大小即为其所含文字的总数。刚才所举的两个项式的大小分别为2和3。

我们很容易证明n个变量可组成3^n种可能的单项式（参看习题2.4）。大小为k或更小的单项式的总数由以下公式约束：

$$\sum_{i=0}^{k} C(2n, i) = O(n^k)$$

其中，$C(2i,j)=\frac{i!}{(i-j)!j!}$为二项式系数。

一个子句是形如 $\lambda_1+\lambda_2+\ldots+\lambda_k$的表达式，其中 λ_i为文字。这样的形式称作“文字析取式(*disjunction*)”。$x_3+x_5+x_6$, $x_1+\bar{x}_4$ 均为子句。子句的大小也是其所含文字的总数。共存在 3^n种可能的子句和少于 $\sum_{i=0}^{k} C(2n,\ i)$ 种大小为k或小于k的子句。若f为项式，则（根据DeMorgan定律）$\bar{f}$为子句，反之亦然。这样，子句和项式互为对偶（*dual*）。

若一个布尔函数可以书写成项式析取式则称为“析取范式（*disjunctive normal form*, DNF）”。$f=x_1x_2+x_2x_3x_4$和$f=x_1\bar{x}_3+\bar{x}_2\bar{x}_3+x_1x_2\bar{x}_3$ 均为DNF。任何布尔函数都能写成DNF形式。由k个项式组成的DNF析取式称为“k项DNF表达式”；若其中最大的项式的大小为k，它就属于k-DNF这一类。刚才所举的两个例子分别称为二项式和三项式，二者均属 3DNF这一类。

析取范式有一个对偶：合取范式（*conjunctive normal form*, CNF）。若一个布尔函数可书写成子句合取式(*conjunction*)，则称为CNF。$f=(x_1+x_2)(x_2+x_3+x_4)$即为CNF的一例。所有布尔函数均有CNF形式。如果一个表达式是k个子句的合取，那么它叫做k子句CNF表达式；如果其中最大子句的大小为k, 那么它属于kCNF类。上面的例子是一个2子句表达式，属于3CNF类。若f是DNF形式，则根据DeMorgan定律，其CNF为$\bar{f}$，反之亦然。

2.2 动作函数的表达和执行

若存在R种可能的动作，则我们必须找出一个恰当的 R值的特征向量函数来计算动作。人们已对多种表达和执行动作函数的方法进行了研究，下面介绍其中的一部分。

2.2.1 产生式系统

产生式系统是动作函数的简单表达形式之一。一个产生式系统包含一个有序规则序列，称为产生式规则(*production rule*)或产生式(*production*)。每一规则写作“$c_i\rightarrow a_i$”，其中c_i是条件部分(*condition part*)，而a_i是动作部分(*action part*)。一个产生式系统包含以下规则集：

$$c_1 \longrightarrow a_1$$
$$c_2 \longrightarrow a_2$$
$$\vdots$$
$$c_i \longrightarrow a_i$$
$$\vdots$$
$$c_m \longrightarrow a_m$$

一般来说，一个规则的条件部分可能是对传感器输入的感知处理而产生的特征的任一二进制值（0，1）函数。通常，它是一个单项式——一个布尔合取。为了选择一个动作，选择规则如下：从第一个规则$c_1\rightarrow a_1$开始，按顺序寻找第一个条件部分值为 1的规则，并选择那个规则的动作部分。该动作部分可以是一个简单动作、对另一个产生式系统的调用或者是一个要同时执行的动作集合。通常，序列中最后一条规则的条件部分的，值为 1；若在此之前没有其他条件部分的值为1的规则，则缺省地执行最后一条规则中的动作。传感器的输入和以其为基础的特征值随着动作的执行也不断改变。假设条件被不停地检测，使在任一时刻执行过程中的动作与

在那一时刻的条件值为1的第一个规则相匹配。

运用布尔代数以及先前为沿边界行动的机器人而定义的特征文字，可以产生下面的沿边界行动路线的产生式系统表示：

$x_4\overline{x_1} \longrightarrow$ north

$x_3\overline{x_4} \longrightarrow$ west

$x_2\overline{x_3} \longrightarrow$ south

$x_1\overline{x_2} \longrightarrow$ east

$1 \longrightarrow$ north

沿边界行动是持续过程的一例——一个永不停止的(*durative*)程序，机器人永远执行这一动作。与此相反，一些任务要在满足目标(*goal*)条件后停止运动，这一目标通常用基于特征的布尔条件来表示。譬如，我们不让机器人永远沿边界行动而让其在走到一个角落后停在那儿。给出一个角落搜索特征c，当且仅当机器人在一个角落时其值为1，下面的产生式系统会让机器人去一个角落（如果确实存在一个可到达的角落）：

$c \longrightarrow$ nil

$1 \longrightarrow$ b-f

这里，nil表示什么都不做，而b-f表示沿边界运动过程，对此我们已经做了定义。

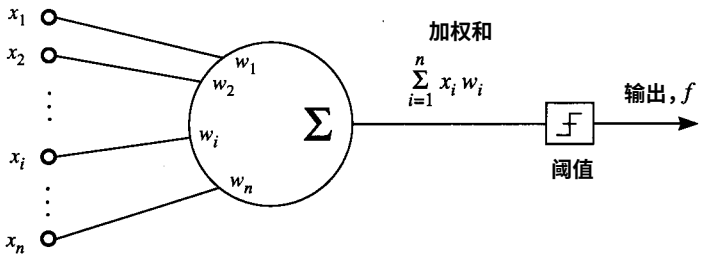
在目标实现的产生式系统中，处于规则序列首位的规则c_1的条件部分说明了我们要让动作完成的总体目标。一旦达到这一目标，agent便停止动作。当状态不满足c_1而满足c_2，通常选择条件c_2和动作a_2，然后动作a_2的完成将最终达成c_1的目标。以此类推，这种产生式系统形成了一个被称作*teleo-reactive*（T-R）程序[Nilsson 1994]的形式化基础。在这个T-R程序中，每正确完成规则序列中一条规则中的动作，就满足了此规则中一个更高的条件。若给出为agent设置的总体目标（使用基于特征的条件说明），书写此特征的产生式系统将十分容易。T-R程序也十分健全，动作一直朝着目标行进。只要感知精确度合理，由错误感知、不恰当的执行动作或对环境信息处理的偏差所造成的偶而的错误是可以得到修正的，而且动作通常能实现其设计效果[⊖]。除这些特征外，当T-R程序被调用时，可以带上参数，而且可以调用其他T-R程序，还能够递归调用自己。

2.2.2 网络

用计算机程序执行布尔函数和产生式系统并不困难。另外，它们也能直接通过电路实现[⊜]，电路输入可以是传感器信号本身。在逻辑电路中，布尔函数由逻辑门（AND，NAND，OR等）网络系统实现。一种常用的电路包括阈值元件或其他能计算其输入加权总和的非线性函数元件的网络系统，其例子之一便是如图2-4所示的阈值逻辑单元（*threshold logic unit, TLU*）。它对其输入的加权总和进行计算，并将结果与一阈值比较；如果超过阈值则输出1，否则输出0。

⊖ T-R程序不仅不断地向目标“行进”，而且能对它们所处的环境做出反应。前缀teleo起源于希腊语“终极”或“目的”。

⊜ 尽管如此，这种电路的行为通常被某种电路模拟计算机程序所模拟。然而，把这种行为视为由一个电路而不是一个程序产生，能帮助我们理解动作对即刻传感器输入的状态依赖。



如果 $\sum_{i=1}^{n} x_i w_i \geq \theta$，$f$=1

否则，f=1

图2-4 一个阈值逻辑单元

可由TLU实现的布尔函数称为线性可分函数(*linearly separable function*)（TLU用一个线性平面——在n维空间中称为超平面——将产生高于阈值响应的输入向量空间与产生低于阈值响应的输入向量空间分开）。许多（但并非全部）布尔函数都是线性可分的。譬如，任一单项式（文字的合取）或任一子句（文字的析取）都是线性可分的，如图 2-5所示，我给出了一个单项式的TLU实现。TLU权值绘制在其响应的输入线旁，而画在圈内的阈值表示 TLU。两个变量的“异或”函数（$f = x_1\bar{x}_2 + \bar{x}_1 x_2$）就是一个线性不可分的函数的例子。

沿边界行动的机器人使用的函数能通过 TLU实现。譬如，图2-6显示了由单个TLU对函数 $\bar{x}_1\bar{x}_2 = (s_2+s_3)(\overline{S_4 + S_5}) = (s_2+s_3)\ \bar{S}_4\ \bar{S}_5$ 的实现过程。

在仅有两种动作可能的应用中，若给出特征向量的代码表示输入，那么单一的 TLU便可计算出正确的动作，而对更复杂的问题，则需要由这样的元件组成的神经网络系统 (*neural network*)。由于TLU被认为是生物神经元的简单模型，这样的电路称为神经网络系统。生物神经元是否被激励取决于若干输入的强度的加权和。我们将在下一章中对神经网络系统进行更深入的研究。

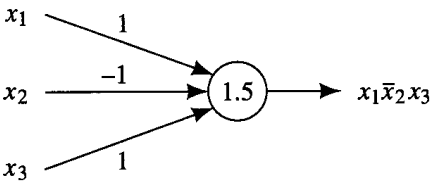


图2-5 用一个TLU实现一个单项式

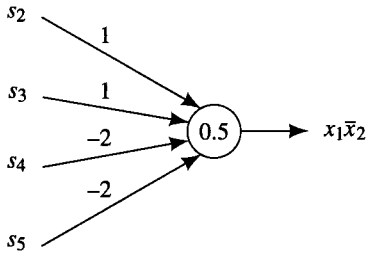


图2-6 沿边界运动的函数的实现

Katz[⊖]提出通过反相器和逻辑与门(AND)组合的简单网络系统结构，可用来实现任一T-R程序。图2-7中表达了我对这种网络系统的理解。网络系统的输入是状态 c_i的二进制值（0，1），网络系统的输出引发相应的动作 a_i（这里并未显示产生c_i的计算，既然c_i通常是文字的合取，那么它们也可以通过TLU实现）。T-R程序中的每一条规则由一个拥有两个输入和两个输出的子电

⊖ Edward Katz, private communication, 1996年4月。

路（称为TISA，即测试、禁止、压制和动作）实现。TISA中的一个TLU计算其中一个输入和另一个输入的补的合取值；另一个TLU计算两个输入的析取值。当以上规则均不满足正确的状态时，禁止输入为0；当对应规则的条件c_i满足时，测试输入为1；若测试输入为1而禁止输入为0时，动作输出为1（即引发相应的动作a_i）。当测试输入和禁止输入任一为1时，压制输出为1（即禁止以下的所有单元）。运用可编程门阵列或等价形式的动态电路结构，可通过调用一个T-R程序来动态构造运行时的电路。

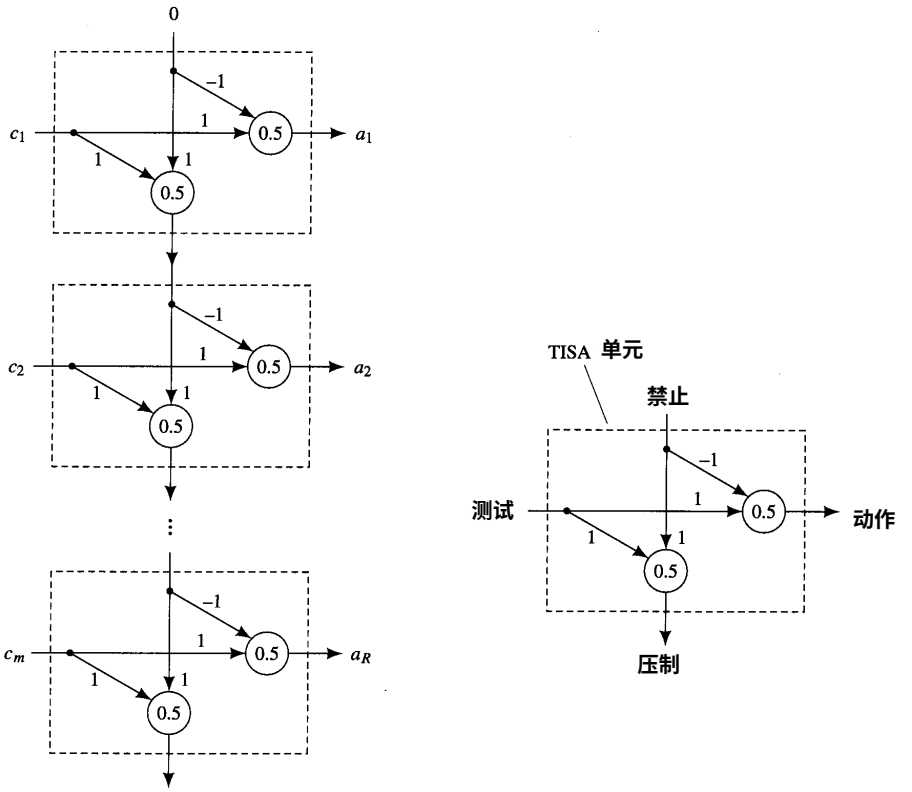


图2-7 由TISA单元组成的网络系统

2.2.3 包含体系结构

实际上还存在其他几种把传感器的即刻输入转换成动作的形式化描述。其中之一便是Rodney Brooks提出的包含体系结构（*subsumption architecture*）[Brooks 1986, Brooks 1990, Connell 1990]。虽然对如何组成这一体系结构还没有确切的定义，但其大体含义是，用一组“行为模块”来控制agent的行为。每一个模块直接从世界接受传感器信息，当传感器输入满足此模块特定的先决条件时，就执行此模块特定的行为程序。一个行为模块可将另外一个行为模块包含其中。在图2-8中，上层模块均可将下层模块包含其中。当模块i将模块j包含其中时，如果满足i的条件，就用模块i的程序替换模块j的程序。与“垂直”体系结构相反，Brooks称其为“水平”体系结构。Brooks和他的学生演示证明，相对简单的反应机器和复杂环境的相互作用可产生极其复杂的行为[Matatic 1990，Connell 1990]。与其他人工智能机器相比，Brooks的机器并不依赖于其所处环境的复杂的内部表达或对这些表达进行的推理 [Brooks 1991a, Brooks

1991b]⊖。然而，必须意识到，尽管所有的S-R机器能做出许多有趣的动作，但它们不可避免地存在相当大的局限性([Kirsh 1991]对Brooks的方法写了一个有趣的评论)。

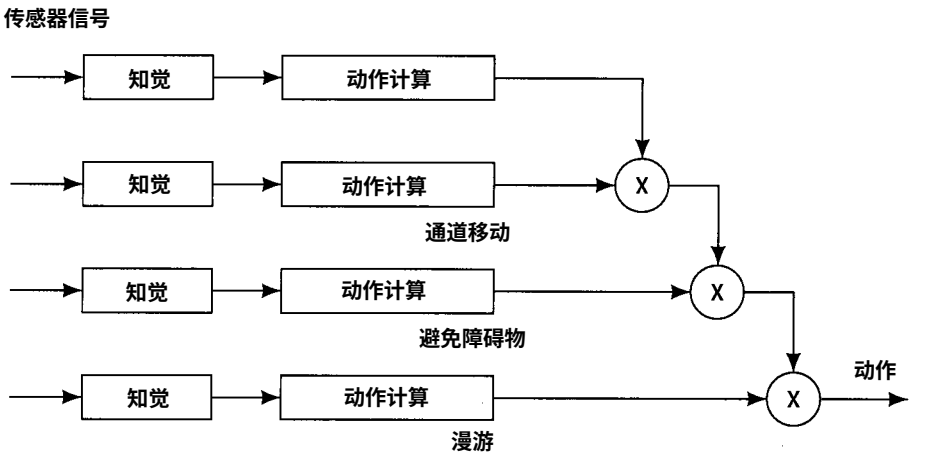


图2-8 包含模块

2.3 补充读物和讨论

S-R agent在现代电子世界中无处不在。恒温器、维持车速的巡逻控制、计算机操作系统中的中断驱动部分以及工厂里成千上万的自动装置均是 S-R agent。一般来说，这种系统和装置并不被视为人工智能的主要论题。我之所以将其纳入本书的论述，是因为它是更高级智能系统的基础。

在一些有关人工智能和机器人的实验课程中，学生们从运用 Lego元件构造S-R机器人开始(请参阅[Resnick 1993])。探索基于行为的控制策略的人工智能研究者已经研制出由探测器指引的可漫游、躲避障碍且沿墙运动的S-R机器人[Mataric 1990, Connell 1990]。

我所提出的感知与动作的分界在agent设计中也十分常见。根据[Kaelbling & Rosenschein 1990]，我们将从传感器的原始输入中计算特征向量的任务分配给感知处理，将选择基于此特征向量的动作的任务分配给动作函数。 Kaelbling 和Rosenschein认为与Brooks的“水平”体系结构相反，这种体系结构方式是“垂直”的。他们对这两种方式的比较如下 [Kaelbling & Rosenschein 1990 pp. 36-37]：

> Brooks认为，对感知和动作的水平分离是一种实用的agent设计研究方法……水平研究方法允许设计者同时考虑那些感知—动作的有限方面来得出特定的行为……而另一种垂直策略则建立在分离的系统模块基础之上，一部分从复合信息来源（如：感知）中提取广泛、有价值的信息，其他部分利用这些信息来达到复合目的（如：动作)。这种垂直研究方法由于其对提取信息和引发动作的固有结合而能有效利用程序设计者的成果，从而十分引人注目。

实际上，感知与动作的分离也适用于水平研究方法。特征向量可分为不同的分量——每个分量由特定的感知设备来计算并引发单独的动作或行为，像包含系统结构一样。

⊖ 因为Brooks 的许多包含机器确实有少量的内部状态，本应该在第 5章中对其进行介绍。我们之所以将其归入本章，是因为它与T-R程序有相似之处。

T-R程序和其他响应系统与动物行为学中不同的动物行为模型紧密相关。这些模型中，对动物动作目标的实现顺序的排列表明，一个动作结果成为“释放”序列中下一动作的“刺激物”或“触发器”。T-R程序正是以这种选择条件和执行动作的方式运行。关于T-R的工作部分受到[Deutsch 1960]的启发。

动物所展示的寻觅并向其刺激物运动的特性称为趋向性。如趋光性动物朝有光的方向移动。[Genesereth & Nilsson 1987]称本章所讨论的这种agent为“趋向性agent”。

一些实验研究运用模拟而非实际的S-R机器人。有人讽刺“模拟注定会成功”[Brooks & Mataric 1993, P. 209]，但是，模拟中的不足之处可以帮助我们进一步完善感知和动作策略。而且，模拟有时确实是“真实的”东西，如计算机教育、娱乐和戏剧艺术中活生生的角色与其模拟环境及用户的交互[Bates 1994, Blumberg 1996]（Blumberg的ALIVE系统是另一个运用动物行为学来构造交互角色的例子）。

有关布尔函数的其他介绍，请参阅[Unger 1989]。产生式系统可视为布尔函数的普遍化，称为“策略表”[Rivest 1987]。

习题

2.1 将下面的布尔函数改写为DNF形式：

$f=(x_1+x_2)(x_3+x_4)$

2.2 证明：$x_1 x_2 x_3 + \bar{x}_1 x_2 x_3 = x_2 x_3$。

2.3 指出下列由三个变量组成的布尔函数中，哪个能通过加权输入的单阈值元件实现，你无需求权值和阀值。

1) x_1

2) $x_1 x_2 x_3$

3) $x_1+x_2+x_3$

4) $(x_1 x_2 x_3)+(\bar{x}_1 \bar{x}_2 \bar{x}_3)$

5) 1

2.4 证明：n个变量可组成3^n个单项式，3^n个子句。

2.5 参照2.1.1节中x_1、x_2、x_3和x_4的定义及2.1.2节的动作规则。证明：在二维网格空间世界中不存在“稠密空间”这一假设暗示不可能同时满足任意两个动作规则。

2.6 手工设计一个神经网络，使其接收传感器信号输入（$s_1, s_2, \ldots, s_8$），产生由TISA单元构成的网络所需的条件输出来完成2.1.2节中为沿墙运动的机器人设计的动作规则。

2.7 本章中指出，由T-R程序指挥的行为是“一直向目标行进”。严格地讲，这一陈述正确吗？你能给出T-R程序不在最高层条件结束的情况吗？