第24章 agent 之间的通信

24.1 交谈

我们的agent可以通过两类方法来有意地影响另一个 agent的动作。假如我们的agent知道另一个agent如何对其环境的变化做出反应，那么，我们的 agent就可以通过改变环境来达到所需的效果。

或者我们的agent可以试着改变另一个 agent的目标、知识（或信念）或者动作选择机制，它可以通过直接“写”另一个 agent的认知结构中的这些元素来达到目的。例如，这个方法能被一个人（我们的“ agent ”）使用以控制一个拥有这种接口通道的 agent。更有趣的是，我们的agent能用这种方法和另一个 agent通信，以使另一个 agent对它的信念和目标（最终使另一个agent采取所希望的动作）做出改变。通信媒体将依靠另一个 agent的感觉和知觉装置。例如，可能包括“写”（基于另一个 agent的视觉能力），“听”（基于另一个 agent的听觉能力）或者“广播“（基于另一个 agent的电磁接收能力）设备。当一个 agent采取这样一个动作以企图影响另一个agent的认知结构时，我们说agent已经参与了一个通信动作(*communicative act*)。

人类之间的交谈常涉及到说话采用的语言（用声波作为介质）。因此，语言学家把各种类型的交流动作称为交谈(*speech act*)。他们把讲话一方称为讲话方(*speaker*)，把受话方称为听众(*hearer*)。根据哲学家John Searle [Searle 1969]的理论（他是第一个研究这种基础理论的人），交谈可分为几类：表示型（*representatives*，它陈述了一个主张或建议）、指示型（*directives*，要求或命令）、委托型（*commissives*，许诺或威胁）、表达型（*expressives*，感谢或道歉）和声明型（*declarations*，事实上它改变了世界的状态，如：“我现在宣布你们成为夫妻”）。

交谈可能有各种不同的物理表现形式，它们可以是一个动作序列（如在手势语言中）、一串符号（在文本中）、一个声音扰动（尖叫、讲话）或闪光。无论何种表现形式，交谈的表现都被称为讲话（*utterance*）。如Searle 所说的，讲话不但要表达讲话的内容还要表达讲话的类型。例如，如果所写的英语文字是媒体，那么把积木 A放在B上既表达了自然的要求也说明了它的要求内容，叫做On (A, B)（这种讲话方式被讲话方用来改变听众的目标结构）。

假定交谈被认为会对听众的知识产生影响。如果我们的 agent $A1$通过一个表示型交谈来通知一个听众$A2$，一个由ϕ指称的建议为真，那么$A1$认为这个交谈的结果是$A2$知道$A1$打算通知$A2$关于ϕ。没有描述$A2$是如何表达这个意图的，也没有描述$A1$如何表达以让$A2$知道。目标表示和意图表达涉及到的装置比在这本书中提到的还要多一些。幸运的是，通过谈话（ *talking*）代替$A1$打算施加于$A2$的影响巧妙地解决这个问题，$A1$想让它的交谈使$A2$相信ϕ。讲话方想施加给听众的影响被称为交谈的言语表达效果（*perlocutionary*）的影响 —— 和它的illocutionary相对照，后者是交谈的实际效果。当然，言语表达效果影响的实现完全受控于听众，只有听众是极端地轻信或者听众相信讲话方是值得信赖的情况下，才能假定言语表达的效果。回到例子中，可以通过假设$A1$假定它的动作（通知$A2$关于ϕ）有言语表达效果来简化我们的讨论。我们假定 $A1$用公式$\mathbf{K}(A2,\phi)$表达它对$A2$的影响。

当人类使用语言时，一个语句的言语表达（打算的）效果有时会产生另外的效果。例如，在句子You left the refrigerator door open中，讲话方的真正意图是要求受话方关上冰箱。交谈的言语表达效果不同于所谓的间接交谈（*indirect speech act*）。一个使用间接交谈的讲话方假定一个受话方能从交谈的环境中推断出讲话方的意图，并用这个意图去决定交谈的言语表达效果。因此，Do you have the time?是一个间接的问话方式（因此被认为更有礼貌），它要求受话方告诉讲话方当前的时间，而并非受话方是否知道时间。

24.1.1 计划交谈

我们能像对待其他的agent动作一样对待交谈。我们的agent能用一个计划产生系统产生由交谈和其他动作构成的计划。为此，需要一个这些动作的结果模型。例如，考虑一个表示型的交谈Tell(α, ϕ),在这里，我们的agent通知agent α，ϕ是真的。通过STRIPS规则，对该动作的结果建模：

Tell(α，ϕ)

PC: Next_to (α)∧ϕ∧¬**K**(α, ϕ)

D: ¬**K** (α, ϕ)

A: **K** (α, ϕ)

前提Next_to (α)确保我们的agent α是充分的接近agent α，能够进行可靠的通信；强加前提ϕ以确保我们的agent在通知另一个agent那个事实前确实相信ϕ；而前提条件¬ K(α, ϕ)确保我们的agent不会传达多余的信息。假定动作有其言语表达影响，即α知道ϕ。假定在环境On(A, B) ∧On (B, C) ∧On(C, Fl)中，我们的agent的目标是On (B, Fl)，还假定我们的agent 知道无论何时当*B*在*C*上、且*B*上无任何东西时，agent *A*1能相应地将*B*移到地板上。用普通的积木世界的STRIPS规则和前述的Tell规则，我们的agent可构造计划{Move(A，B，Fl), Tell (A1, Clear (B)∧On (B, C))}。

24.1.2 实现交谈

交谈的实现或物理表现是讲话，现在必须讨论一下通信动作，如 Tell(α, ϕ)，是如何像讲话方和受话方之间的交谈一样被传输的。考虑两种可能性：(a) 从讲话方到受话方的一个逻辑公式的直接传输；（b）讲话方讲到一些符号字符串，然后受话方将它们翻译成它的认知结构（也许翻译成一个逻辑公式）[⊖]。

如果讲话方和受话方共享同一类基于特征的世界模型，使用相同符号的逻辑公式，那么一个交谈就能通过传播一个逻辑公式来实现（加上交谈的类型信息）。在这种情况下，例如，交谈Tell ((A1, Clear (B)∧On (B, C))就能通过让我们的agent向*A*1发送公式Clear (B)∧On (B, C) 和一个表示类型的指称来实现。注意，这样做要假定公式 Clear (B)∧On (B, C)对*A*1表达的意思与它对我们的agent 表达的意思相同（短语“ means the same thing ”暗示了数据库中的所有公式对我们的agent 和*A*1的逻辑模型是充分的相同）。甚至在我们正在对所有的agent（我们的和其他的agent）构建和编程的情况下，相同的知识表示词汇的假设，对一些有趣的环境来说也是非常难以令人相信的。即使两个 agent的知识表示词汇和模型在启动时是一致的，如

⊖ 符号字符串只是可以被传输的东西的一个例子，人和人之间也可以通过图表、图片和其他的媒体传输信息。

果遇到任何新的对象，这些agent也几乎不可能会给这些对象相同的内部名字。如果agent能够发明新的谓词，这些谓词按照更原始的方式定义，而这些等价的谓词也可能会有不同的符号。因为我们的agent刚好要对另一个agent采取一个有意的姿态，这样另一个agent就不一定要用逻辑公式对它的世界模型编码。

在这些条件限制下，agent之间的通信如何发生呢？一种答案涉及到早期阐述过的两种选择中的第二个：用一个一致的、通用的通信语言，通过设计、使用和指令，通信agent能学会在这种通用语言方式下传输的符号字符串是如何改变其他agent的认知结构的。采用一个有意的立场，我们的agent能够预测，例如，如果传输给agent *A*1的字符串是`block B is on block C and block B is clear`，那么对*A*1的认知结构的言语表示效果影响能用我们的agent的知识库中的公式**K**(A1, On(B, C)∧Clear (B))描述（不管*A*1实际上是如何表示那个知识的）。在这个例子中，我们的agent通过发送讲话`block B is on block c`和`block B is clear`来执行交谈`Tell (A1, On (B, C)∧Clear (B))`。听众把讲话翻译成它用来表示这个知识的任何内部形式。当然，由于假定正在设计这些agent，我们能选择我们喜爱的任何通用通信语言。如果我们的agent也需要用一些像英语一样的语言与人通信，就可以发明一个如例子中所说的类似于英语的语言。

使用基于符号字符串的语言预示了两个问题的解决方案：给定一个交谈，如何生成一个符号字符串；如何把一个符号字符串翻译成对一个认知结构的影响。虽然鼓励在人工agent中使用符号字符串作为通信媒体，但是机器生成和自然语言（人类使用的语言，像英语、法语和汉语等）的理解是依各自的兴趣的。通信使用的符号字符串的生成和理解主要在自然语言的领域学习研究。因此，对agent之间通过字符串的通信，处理将主要集中在类似于英语的语句讲话之中。自然语言的处理、生成和理解极其困难，到目前为止只取得了有限的进展。人类级的能力毫无疑问是解决大量AI问题的先决条件，在下一章我们将看到这个，不过已有一些应用已经是可行的。在下一部分对使用语言理解系统的一些重要的技术进行了归纳（省略了讨论语言生成，对这方面感兴趣的读者可以参见[Appelt 1985, Sadek, et al, 1996]，它谈到了使用计划和推理生成自然语言语句，[McKeown & Swartout 1987]对这个技术做过评论，[McDonald & Bolc 1988]关于这个主题出过一本书。已有几个商用计算机接口系统具备一定的产生文本和讲话的能力）。

24.2 理解语言字符串

下面要考虑一个符号字符串——由一个企图传递一定命题的讲话方传播，是如何被一个受话方翻译成一个指称（我们希望）相同命题的公式的。这个翻译过程所需的一些（但不是全部）信息被嵌入在字符串的语法属性中。例如，一个讲英语的讲话方可能希望将字符串`block B is on block C`翻译成On (B, C)而不是On (C, B)。这些语法特性用句法描述最方便，句法不仅限定了哪些字符串是该语言的合法句子，而且定义了句子的结构。使用一个技术简单的“原型英语(*proto-english*)”的例子来说明这些思想，它不但是介绍自然语言理解技术的第一步，同时也是可能的agent语言的一个例子。

24.2.1 短语结构语法

短语结构语法定义了符号串的基本组成——符号本身是如何被集成短语，以及这些短语是

如何最终集成为句子的。句子被分解成它的构成短语，最后终止于字符串符号级，这种方式定义了一个句子的结构，这个结构是把一个句子翻译成一个逻辑公式的关键。语言的元件是终结符号和非终结符号。最高级的非终结符是句子，由符号 S表示。每个非终结符被定义为一组其他的符号 —— 非终结符或终结符或者两者的组合。在所谓的上下文无关（*context-free*）的语法中，一个非终结符的定义在一个字符串中独立于它周围的符号。下面示例了一个上下文无关的语法例子，它适合于堆积木问题中的有限通信。

一个句子*S*被定义为某个名词短语（*NP*），其后面跟随一个动词短语（*VP*）。这个定义由下面的规则指称：

S ← *NP VP*

我们也允许由两个句子的合取（*Conj*）递归地构成一个句子：

S ← *S Conj S*

这两个规则能简化为巴科斯—诺尔范式（BNF）：

S ← *NP VP* | *S Conj S*

其中“|”读作“或”

现在，我们必须定义刚刚引入的非终结符号 *NP*、*VP*和*Conj*。其中一些非终结符将由另外的非终结符定义，依次类推，直到所有的定义都终止于终结符号。一个合取可以被定义为终结符`and`或者`or`：

Conj ← `and` | `or`

一个名词短语被定义为一个名词（*N*）或者一个由名词跟随的形容词（*Adj*）

NP ← *N* | *Adj N*

对于简单的三积木块问题，一个名词是终结符`A`、`B`、`C`或者`Floor`之一（在积木问题中实体的通用名称）。我们也允许终结名词`block A`, `block B` 和`block C`[⊖]，这些定义能用下面的规则指称：

N ← `A`|`B`|`C`|`block A`|`block B`|`block C`|`Floor`

对形容词的定义如下：

Adj ← `clear`|`empty`|`occupied`

动词短语的定义如下：

VP ← `is` *Adj* | `is` *PP*

最后，介词短语（*PP*）和介词（*Prep*）被定义为：

PP ← *Prep NP*

Prep ← `on` | `above` | `below`

为方便起见，把所有的这些规则收集在表 24-1中，在自然语言的处理系统中，终结符号是语言中的所有单词，它们被存在一个被称为词典（*lexicon*）的数据库中。

这些语法规则允许的句子结构可以方便地用树来显示。例如：句子 `block B is on block C and block B is clear`的结构如图24-1所示。在一个完整的结构中，每一个非叶节点由一个非终结符号标识，且有由语法规则生成的后代。每个叶节点由一个终结符号标识。由非终结符号标识的节点是方形的，终结符标识的节点是圆形的。

⊖ 从技术上讲，`block` 和`A` 的组合被称为名词—名词组合。我们仅把它们做为附加的名词来简化讨论。

根据语法规则，可以确定下面的字符串是合法的句子[⊖]。

```
block C is occupied
A is on C
B is on occupied C
occupied B is on clear C
```

决定一个任意的字符串是否是一个合法的句子被称为解析(*parsing*)字符串，解析过程被称为语法分析（*syntactic analysis*）。存在着各种分析算法。在一个自上而下（*top-down*）的算法中，语法规则应用于（在一个“向后”的方向）非终结符 *S*，按照它的构成短语重写它，直到产生一组匹配给定字符串的终结符。这个过程可以通过一个 AND/OR树进行搜索，直到找到一个解决方案树（图24-1中的树就是这样一个解决方案树）。在一个自底向上（*bottom-up*） 的算法中，被分析的字符串的子串用非终结符代替，这些非终结符被其他的非终结符代替（均依照语法规则），直到产生单个非终结符*S*。为了提高搜索效率，这个过程常常顺着字符串按从左到右的方式进行。这个过程能用深度优先、回溯搜索实现。

表24-1 堆积木问题的通信语法

```
S ← NP VP|S Conj S
Conj ← and|or
NP ← N|Adj N
N ← A|B|C|block A|block B|block C|floor
Adj ← clear|empty|occupied
VP ← is Adj|is PP
PP ← Prep NP
Prep ← on|above|below
```

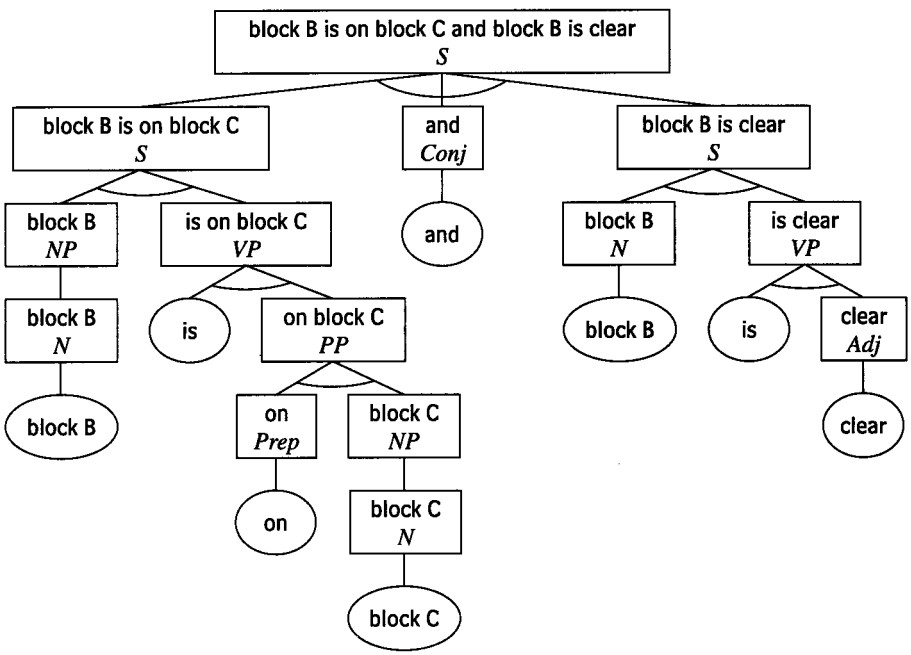

图24-1 一个句子的结构

24.2.2 语义分析

回忆一下，传输一个符号字符串的目的是该字符串能被一个受话方翻译成一个逻辑公式。前面已经指出字符串的语法结构能表达很多的预期含意。在此描述一种翻译方法，它是语法分

⊖ 为简单起见，这里忽略了大写和标点符号，尽管引入这些特征的文法也能写出来。

析过程本身的一个附加的效果。这个附加的好处通过把逻辑公式的元件与句子的每个短语连接起来而获得。短语结构语法的每一个规则指定了和一个给定的短语相关的公式元件是如何由与组成的子短语相关的公式元件构成的。例如：规则

$PP \leftarrow Prep\ \ NP$

必须按照*Prep*和*NP*的语义关系指定*PP*的语义关系。这些语义关系通过把每个非终符作为一个函数表达式来表示，函数表达式用语义关系作为参数；例如，*PP*（*sem*）。

作为分析的结论，和非终结符*S*相关的公式就是字符串的含意。使用这些关系的，语法称为扩充短语结构语法(*augmented phrase-structure grammar*)，分析过程完成了所谓的语义分析(*semantic analysis*)（以及常说的语法分析）。

通过扩充我们的短语结构语法的例子和对已经分析过的例句进行语法分析来说明这个过程。从将逻辑结构和每个终结符联结开始，先看一下那些语法规则分类为名词的终结符号。从直觉上讲，名词应该和谓词演算中对象常量相关。对这个简单的语法，这个关系通过将规则 *N* ← A|B|C|block B|block C|Floor 如下表达来实现：

A → *Noun*(E(A))

B → *Noun*(E(B))

C → *Noun*(E(C))

block A → *Noun*(Block(A))

block B → *Noun*(Block(B))

block C → *Noun*(Block(C))

floor → *Noun*(Floor(Fl))

例如其中第一个规则，它陈述了和名词“ A ”相关的语法元件是原子E（A）（E(A)的预期意思是由A指称的对象是一个“实体”）。在语法分析中，这个规则陈述了终结符 A在符号串的出现能被重写为*Noun* (E(A))。按从左到右的方式写规则（用一个向右的箭头），因为这些规则打算应用到“自底而上”(用箭头右边的公式代替终结符）的分析中。

如果在我们的例子中执行由这些规则指定的一些替换，会得到下面的部分已分析的句子：

Noun(Block(B)) is on *Noun*(Block(C)) and *Noun*(Block(B)) is clear

下面是我们为终结符and和or写的扩展规则：

and → *Conj*(∧)

or → *Conj*(∨)

应用and的规则生成：

Noun(Block(B)) is on *Noun*(Block(C)) *Conj*(∧) *Noun*(Block(B)) is clear

公式的元件并非与其他的终结符直接相关。形容词 clear、empty和 occupied都陈述了一些对象的属性，因此它们应该用适当的关系常量和对象常量引入谓词演算原子。但是，在我们代替这些形容词时，可能不知道它们正在描述哪个对象的属性。因此，对形容词 clear，它应当有如下的规则

clear → *Adj*(Clear(*x*))

表达式clear（x）将被解释为一个谓词演算范式，为了生成一个谓词演算公式，该范式需要

应用到一些对象常量上。就像计算机科学中的其他领域一样，我们用 lambda表达式定义这种范式。因此，对clear规则的一个更精练的描述是：

$$\text{clear} \rightarrow Adj(\lambda x\ \texttt{Clear}(x))$$

在这里我们看到一个短语的“含意”有时用另一个短语的含意来表达。例如，如果我们把该范式应用到对象常量B，就会得到：

$$(\lambda x\ \texttt{Clear}(x))\texttt{B} = \texttt{Clear(B)}$$

把clear规则应用到我们已经部分解释的句子，产生：

$$Noun(\texttt{Block(B)})\ \text{is on}\ Noun(\texttt{Block(C)})\ Conj(\wedge)\ Noun(\texttt{Block(B)})\ \text{is}\ Adj(\lambda x\ \texttt{Clear}(x))$$

(对其他的形容词我们有相似的规则。)

在给出其他终结符的规则之前，我们先看看非终结符——名词短语。首先，一个名词是一个名词短语（在语法上没有任何改变）：

$$Noun(\phi(\sigma)) \rightarrow NP(\phi(\sigma))$$

其中ϕ (σ)是一个一元原子（带有模式变量σ)。应用这个规则产生

$$NP(\texttt{Block(B)})\ \text{is on}\ NP(\texttt{Block(C)})\ Conj(\wedge)\ NP(\texttt{Block(B)})\ \text{is}\ Adj(\lambda x\texttt{Clear}(x))$$

一个名词短语也是一个跟随着名词短语的形容词。由于将在后面的归纳表中给出，在此就不需用该规则来分析我们的示例句子。

在语法例子中，只有一个动词 is。它总是和一个形容词或一个介词短语合用。当使用一个形容词时，我们有规则

$$\text{is}\ Adj(\lambda x\ \phi(x)) \rightarrow VP(\lambda x\ \phi(x))$$

在这个规则中，ϕ 是一个一元关系常量，指称该形容词的相关属性（在这个简单语言中的单词没有加任何附加的语法内容)。

在给出动词短语的其他规则之前，先给出一个名词短语是如何与一个动词短语组合而产生一个句子的：

$$NP(\phi(\sigma))VP(\lambda x\ \psi(x)) \rightarrow S((\lambda x\ \psi(x) \wedge \phi(\sigma))\sigma)$$

其中，ψ是任何关系常量（在我们的语法中它可以是任何元)，当应用lambda表达式时，ψ的lambda变量x被约束。这里我们有了组合两个构成短语的语法的第一个实例。这个组合通过把来自动词短语的一个lambda 表达式应用到由名词短语给出的对象常量上而获得。这个应用能在语法分析产生下面的精简规则之前被使用：

$$NP(\phi(\sigma))VP(\lambda x\ \psi(x)) \rightarrow S(\psi(\sigma) \wedge \phi(\sigma))$$

继续应用这些规则会产生

$$NP(\texttt{Block(B)})\ \text{is on}\ NP(\texttt{Block(C)})\ Conj(\wedge)\ NP(\texttt{Block(B)})\ VP(\lambda x\texttt{Clear}(x))$$

$$NP(\texttt{Block(B)})\ \text{is on}\ NP(\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

接下来，我们解决介词和介词短词

$$\text{on} \rightarrow Prep(\lambda x y\ \texttt{On}(x, y))$$

(注意on引入一个二位（*two-place*）谓词。Lambda符号跟踪哪个变量是哪一个)。

$$Prep(\lambda x\gamma\ \psi(x,\gamma))NP(\phi(\sigma)) \rightarrow PP(\lambda x\ (\lambda\gamma\ \psi(x,\gamma) \wedge \phi(\sigma))\sigma)$$

它能被简化为：

$$Prep(\lambda x\gamma\ \psi(x,\gamma))NP(\phi(\sigma)) \rightarrow PP(\lambda x\ \psi(x,\sigma) \wedge \phi(\sigma))$$

其中ψ是与介词相关的任何二位关系常量。

动词短语的其他规则是

$$\text{is } PP(\lambda x\ \psi(x,\sigma)) \rightarrow VP(\lambda x\ \psi(x,\sigma))$$

(注意，ψ是一个以模式变量σ 和lambda变量x为参数的二元关系常量)。我们继续应用这些规则和前面对S的规则产生：

$$NP(\texttt{Block(B)})\ \text{is}\ Prep(\lambda x\gamma\ \texttt{On}(x,\gamma))\ NP(\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

$$NP(\texttt{Block(B)})\ \text{is}\ PP(\lambda x\ \texttt{On}(x,\texttt{C})) \wedge (\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

$$NP(\texttt{Block(B)})\ VP(\lambda x\ \texttt{On}(x,\texttt{C})) \wedge (\texttt{Block(C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

$$S(\texttt{Block(B)} \wedge \texttt{Block(C)} \wedge \texttt{On(B,C)})\ Conj(\wedge)\ S(\texttt{Clear(B)} \wedge \texttt{Block(B)})$$

最后，我们有联接两个句子的规则：

$$S(\gamma_1)Conj(\wedge)S(\gamma_2) \rightarrow S(\gamma_1 \wedge \gamma_2)$$

应用这个规则， 重新整理，简化后产生

$$S(\texttt{On(B,C)} \wedge \texttt{Clear(B)} \wedge \texttt{Block(B)} \wedge \texttt{Block(C)})$$

显示这个句子的所有语义分析过程的语义分析树如图 24-2所示。为了清楚直观，谓词演算公式显示在相邻的对应节点上（而不是做为相关的非终结符的参数）。注意和短语相关的语义结构是如何由子短语语义结构构成的。我们说这类扩充的语法是合成语义（ *compositional semantics*），为了参考方便，这个语法的所有规则被收集在表 24-2中。

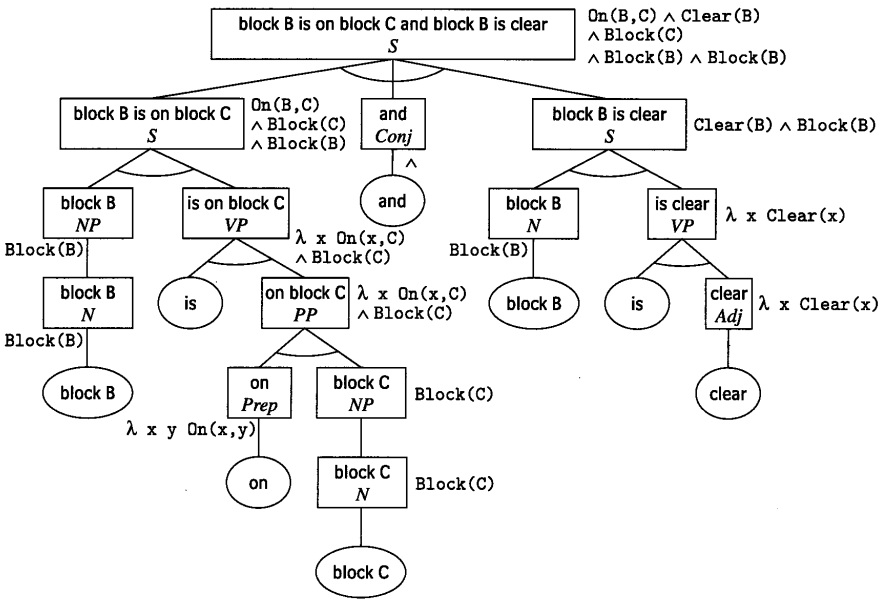

图24-2 一个语法分析树

表24-2 一个扩充的短语结构语法

A → *Noun*(E(A))
B → *Noun*(E(B))
C → *Noun*(E(C))
block A → *Noun*(Block(A))
block B → *Noun*(Block(B))
block C → *Noun*(Block(C))
floor → *Noun*(Floor(Fl))
and → *Conj*(∧)
or → *Conj*(∨)
clear → *Adj*(λx Clear(x))
empty → *Adj*(λx Clear(x))
occupied → *Adj*(λx ¬Clear(x))
on → *Prep*(λxy On(x, y))
above → *Prep*(λxy On(x, y))
below → *Prep*(λxy On(y, x))
is *Adj*(λx $\phi(x)$) → *VP*(λx $\phi(x)$)
is *PP*(λx $\psi(x, \sigma)$) → *VP*(λx $\psi(x, \sigma)$)
Prep(λxy $\psi(x, y)$)*NP*($\phi(\sigma)$) → *PP*(λx $\psi(x, \sigma) \wedge \phi(\sigma)$)
Noun($\phi(\sigma)$) → *NP*($\phi(\sigma)$)
Adj(λx $\phi(x)$)*NP*($\psi(\sigma)$) → *NP*($\phi(\sigma) \wedge \psi(\sigma)$)
NP($\phi(\sigma)$)*VP*(λx $\psi(x)$) → *S*($\psi(\sigma) \wedge \phi(\sigma)$)
S(γ_1)*Conj*(∧)*S*(γ_2) → *S*($\gamma_1 \wedge \gamma_2$)

24.2.3 扩展语法

虽然表24-2中的语法能够把很多积木世界中的句子翻译成逻辑公式，但构造它只是为了说明这种类型的语法分析的一般思想。在不会产生过于复杂的情况下，可以加几个明显的附加规则。例如：为附加的积木和agent加上其他的规则，使具有更多形容词(red, heavy ,...)、更多的介词（next to, between,...）和更多的名词。这个语言中的惟一动词is没有起到重要的作用。可以加上动词 know，它会翻译成一个使用 K的模式范式。翻译句子 Sam moves block A from blocktO block B中的其他动词将使我们决定如何构思这种动作和如何用逻辑公式描述它们。加入时态动词（如moved）将要求翻译成一个能够描述时态事件的公式。

含有冠词，如the和a的句子常常涉及到翻译成量化公式。例如：block A is on a block应该被翻译成(x) On(A, x)。在考虑量词的范围时，包含every、all和some这些单词的英语句子是具有二义性的。例如，句子 all blocks are on a block 可能被译成（ x）（ y）On(x, y)或者（ y）（ x）On (x, y)。常常通过参考其他的知识源 —— 不包含在该句子中的知识来解决这种二义性。在后面，更详细地讨论了二义性问题和它们的解决方案。语义分析通过先把它翻译成一个特殊的准逻辑范式（*quasi-logical form*）[Russell & Norvig 1995, pp, 676以后]语言（能够保留一定的二义性），而不是直接翻译成一阶逻辑公式（一般不能容纳二义性）来推迟对二义性的消解。然后，准逻辑范式到一阶逻辑的翻译可以在一个单独的阶段进行，它可以利用附加的知识来消解剩下的二义性。

通常，不能用上下文无关语言充分定义自然语言中的句子。例如，考虑句子 block A and block B are on block，接受一个带有上下文无关的规则 $S \leftarrow NP\ VP$ 的语法句子也可以接受 block A and block B is on block，因为 is on block C 将作为一个合法的 *VP* 被接受。在主语和动词之间加强单—复数协议也需要附加的上下文无关规则来定义单数（复数）名词短语和单数（复数）动词短语，或者它需要一种上下文有关文法。有趣的是，允许相关的短语语义范式的相同扩充技术也能被用来加强各种上下文相关约束，包括那些涉及到数量（单数、复数）、人称（第一、第二或第三）、时态、动词类型（及物的或不及物的）和情形（主观、客观）的情况。为了加强单—复数协议，例如我们可以加一个额外的参数 n（数字）给适当的短语名称。于是对句子的规则就变成 $S(n) \leftarrow NP(n)\ VP(n)$，为简单起见，这里忽略了指定语义范式的参数。当应用这个规则时，对变量 *n* 的绑定，不管 s（单数）还是 p（复数）都必须一致。在 *VP*（*n*）中绑定 *n* 将由分析树底部的下述规则决定：are ← *verb*(*p*) 或 is ← *verb*(*s*)。在 *NP*（*n*）中绑定 *n* 也被同样确定。使用这种机制的文法叫做合一文法（*unification grammars*），因为术语“合一”就是在变量之间的强制统一。

24.3 有效通信

通信的实际效果常常能通过受话方使用其自身的知识来帮助决定一个谈话的意思来获得。“对一个聪明人而言一个单词就够了（A word to the wise is sufficient）”。如果一个讲话方知道一个受话方能明白他讲的意思，那么他就能发送更短更少的信息。计算机难以理解自然语言的一个主要原因就是理解需要很多知识源，它包括关于通信发生的上下文知识，还有讲话方和受话方共享的“共识”知识。在接下来的几个小节中，将简要谈论一下这些主题。

24.3.1 上下文的使用

如果受话方和讲话方共享相同的上下文（也就是说他们都知道彼此正在交谈的环境），那么该上下文能被作为一个知识源来决定一个谈话的意思。使用上下文允许语言中有代名词（如 he 和 it）和索引词（*indexical*）（如 here, now, I 和 you）。上下文可以包括前面的通信、当前的环境状态，或者两者都有。例如，考虑谈话 block A is clear and it is on block B。受话方在已经听到且理解了短语 block A is cle 的上下文中处理短语 it is on block。在这个上下文中，受话方可以假定 it 指的是已经提到的那个积木 A[⊖]。在示例句子中，单词 it（代替重复的单词 block A）总共只使用了一次，只是一个很小的节约，但是大量相同的节约加起来就能提高效率。当然，为了得到这个效率，讲话方必须知道受话方将能够明白那个指代。

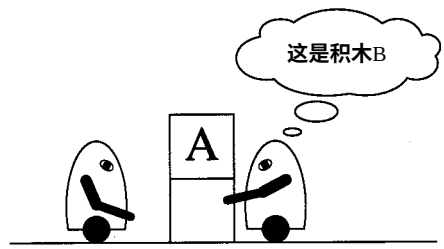

图24-3 指示建立一个索引词的含意

当一个讲话方使用索引词 I 时，在相同上下文中的受话方知道单词 I 指的是哪个人或机器。例如，如果机器人 R1 说 I know that

[⊖] 语言学家用术语 anaphora 描述那些已经提到过的短语（包括代名词）。一般地讲，决定首语重复项的意思极其困难，这个例子是一个非常特殊的简单的例子。

block A is on block B，那么当R1正在讲话时，一个知道机器人R1身份的受话方能把这个谈话解释为K（R1，On (A, B)）。有时，像this这样的索引词被用来与指示动作结合使用。例如，在图24-3中，我们看到一个机器人指着一个积木说this is block B。看到讲话的机器人指向一个积木，收听的机器人把那个讲话解释为On（A，B）。讲话方这样讲因为它想让受话方知道On（A，B），并且它知道受话方看不到积木 *B*上的标签，但能看到积木 *A*上的标签，且能看到*A*在那个被指定的积木上面。我们会再次得到一个小的节约，但对有效的通信还是有点贡献的。

24.3.2 使用知识解决歧义性

即使能够设计agent通信语言，使得命题能被传输和无二义性地、单一地理解，这种无二义性的通信将需要很难管理的大量词汇和极其复杂的表达。使用自然语言的人能用各种知识源来洞悉有歧义性的单词和短语的预期含意，这种能力允许人类在使用我们的语言时能更有效，而且它也许能赋予agent通信相似的好处。

计算语言学家已经对自然语言交谈中的歧义性进行过分类。下面用积木问题和机器人（和人）在积木问题中的通信来描述这些类型（在这些例子中用到的语言和语义学比表 24-2中更复杂）。

1. 词汇歧义

同一个词可能有几个不同的意思，由此导致的歧义性叫做词汇歧义（*lexical ambiguity*）。句子robot R1 is hot既能表示R1非常擅长它的工作，也能表示R1非常热。解决这种歧义性将需要关于R1的附加知识和句子出现的环境。

2. 句法歧义

有时句子能用不止一种方式表达，即短语能被不同地组合。由此产生的歧义性称为句法歧义（*syntactic ambiguity*）。句子I saw R1 in room 37既能暗示讲话方正在进行看这个动作，也能指R1是在37号房间。如果受话方已经知道讲话方或者 R1所指的位置，就能解决歧义性（这种问题的一个典型例子是I saw the Grand Canyon Flying to New York，这个句子的歧义性可通过常识知识有人飞走了而不是Canyons来解决）。词汇歧义性会产生不同的表达结果，在句子clear block A and B and ace on the Floor中，clear既能是动词也能是形容词。这类问题的一个典型例子是 time flies like an arrow。

3. 引用歧义

代名词和其他首语重复（*anaphora*）的使用可能会产生歧义性。解决这类引用歧义性（*referential ambiguity*）的过程涉及到对讲话方和受话方的部分内容进行复杂推理。在句子block A is on block B and it is not clear，讲话方可能想让it指的是积木 *A*而不是积木 *B*。讲话方可以合理地假定受话方将使用常识知识从短语 blockA is on block 中推断出¬ clear (B)，因此也会得出结论it在短语it is not clear中指的是积木*A* (因为讲话方经常不会传送冗余的信息)。

4. 语用歧义

使用上下文知识和其他知识解决歧义性的处理过程常被称为语用分析（pragmatic analysis）(相对于句法和语义分析过程)。但是，如果一个受话方的共识知识和上下文知识是不确定的，即使语用分析也不能解决歧义性。这种歧义性称为语用歧义（*pragmatic ambiguity*）。在句子

R1 is in the room with R中，如果受话方认为R2是在37号房间或38号房间中，但不能确定是哪一间，那么短语the room的意思将是歧义的。

24.4 自然语言处理

在积木问题中，agent的通信仅仅是在一个能够理解由人类讲的（或写的）句子的系统中所需要的很少的一部分。自然语言处理（NLP）是一个极有潜力的应用领域，包括语言翻译，从数据库中浏览信息、人机接口和自动听写（*automatic dictation*）。具有语言能力的计算机系统将能够理解和产生在文本（即句子字符串）或在交互式交谈中所写和所说的句子。尽管已有一些研究系统和一些能力有限的商用产品，但是这个领域还远远不能达到上面的那些目标。

NLP已被描述为“AI 难题”（类似于NP 难题）。也就是说，为了产生一个具有和人类一样语言能力的系统将需要解决“AI问题”。大部分困难在于解决语用歧义性，它要求对大量的常识知识库进行推理。当然，在建立词汇表、文法和适合处理自然语言的分析系统中也有一些较大的困难。

几乎所有来自文本和交谈的句子都能用来说明这些问题。让我们看看一个有语言理解能力的系统为了分析一个取自交谈的句子都需要些什么（这里只略微谈到很少的几个问题！）考虑下面的在一个教授和一个研究生之间的一段谈话。

P: well, I'll need to see your printout
S: I can't unlock the door to the small computer room to get it.
P: Here's the key.

一个计算机系统如何分析那个学生的话呢？很多NLP系统被组织成一个步骤序列，首先，用一个带有语义组合规则的文法和一个词典，将句子翻译成逻辑范式。下面仅是该句子在这个处理级所处理的一些问题：

- 词典可能有像lock这样的词根和它的各种含意。它也需要关于前缀和后缀如何连接到词根的信息以及这些连接如何影响词根的含意的信息（形态学分析 *morphological analysis*）。
- 文法和语义分析需要能够处理比简单的积木问题文法范围更广的各种交谈。像quickly这样的副词会改变它们修饰的动词的含意。考虑一个形容词，如small，它的意思会根据它修饰的名词而变化，一个小屋子可能比一个大计算机更大。
- 短语the small computer room可能是说一个小屋子中有计算机，也可能是小计算机放在屋子中。为了推迟解决这种歧义性，系统或者生成一个保留歧义性的逻辑范式或者生成两个不同的逻辑范式。
- 系统需要能够找到I和it的指代物。I可能会被容易地确定为讲话方。在这一级分析中，it可能指的是门、房子或者前面谈话中提到的任何东西（将要确定）。消解歧义性可能被再次推迟。
- 解释单词can't ，即cannot的缩写形式，要求系统有处理否定的能力。在这种情况下，系统也要能解释can指的是自然的能力还是被允许。

在产生可选的逻辑范式后，系统用一般环境知识推理解决一些歧义性。这里，可以确定it一定指的是前面谈话中的某个东西，因为讲话方不可能去“get”一个门或一间屋子。同样，can可能指的是自然的能力，因为打开一个门需要钥匙，而钥匙是人们常常没有的东西。

系统也用确定的当前环境知识来解释短语small computer room。假如系统知道所有指定的计算机屋子都有小计算机，而只有一个房间是小房间，比如说246号房间；那么small computer room一定指的是246号房间。

这个句子的最终分析需要谈话的知识和对讲话方意图的推理能力。教授要求打印结果，因此，it一定指的是它。这里的推理链相当复杂。对打印结果的要求为学生建立一个目标，可以假定那个学生要制定一个计划去到达那个目标。当那个学生意识到他没有进到放打印结果的屋子时，他变得支支吾吾。那个句子是关于学生不能到达目标的原因，系统也必须那样解释它，而且，系统需要解释学生的讲话是企图协商对目标的变更呢（不要烦我去拿打印结果因为我没有钥匙，也许你（教授）也不是真的需要打印结果），还是想要（间接的）那个屋子的钥匙（学生知道教授有那个房间的钥匙）以便为学生能生成一个可以执行的计划去达到目标。间接要钥匙也包含了为什么要钥匙的原因——学生假定教授需要一个原因被说服以给他钥匙。

执行所有这些推理的方法和表示被推理的知识的方法仍然是人工智能的前沿研究领域。因此，让计算机理解像我们的例子一样的谈话确实是一个A1难题。由于语言理解的这些困难可能还必须与一些反应型处理组合起来，这使语言理解问题变得更加困难了。例如，前述情况下的学生可能只说了一个不合文法规则的Key？来代替那个更长的句子，教授可能的反应是把合适的计算机房的钥匙给学生——几乎不需要什么分析！但是，这个反应要求教授决定移交哪个钥匙，因此不知道究竟是用复杂的推理做出那个结论还是简单地根据相关的上下文反应过程做出结论。

自然语言处理，包括语音、文本的产生以及理解，是一个既有自己的惯例和技术，又与AI相辅相成的领域。在此不做详述，若对更详细的细节感兴趣，可从[Russell & Norvig 1995，第22和23章]和[Allen 1995，Grosz, Sparck Jones, & Webber 1986]开始学习（也可参见NLP的专刊《Artificial Intelligence》，vol.63，nos.1-2，October 1993）。

24.5 补充读物和讨论

[Cohen & Perrault 1979]描述了用AI计划系统如何来计划交谈。为了让一个谈话达到它的预期表达效果（即讲话方的期望效果），一个受话方有时可能必须通过观察讲话方的一系列动作才能猜出讲话方的总体意思。[Kautz 1991]评论了这个问题，并对计划识别和实现提出了一个形式理论。

[Chomsky,1965]介绍了语言的句法和分析的基础。对字符串的语法和语义处理是建立在由[Pereira & Warren 1980]最先研制的明确子句文法之上。还有很多其他的语法形式，包括[Woods 1970]的扩充转换网络（ATN）。一个对英语进行解释的典型大型文法形式是用在SRI国际TEAM系统中的TEAM[Grosz, et al. 1987]。

[Magerman 1993]描述了学习语法的统计方法，[Charniak 1993]推广了可能和那些规则有关的一个语法概念。

[Grosz , Sparck Jones & Webber 1986]和[Waibel & Lee 1990]分别是关于自然语言处理和语音识别的重要论文集。

和语言理解方法相比，基于向量的文档词频统计常常用来根据内容把文档分成有意义的类别[Masand , Linoff & Waltz 1992, Stanfill & Waltz 1986]。

习题

24.1 你能想像这样一种场合吗？一个agent讲了一句tall，而且该agent知道这个动作的命

题内容不是真的。

24.2 用表24-2中的文法（语义规则）对下面的句子进行语义分析：

```
block B is on floor or block B is on C
```

24.3 如何修改表24-2，使它包含单词not以便not既能与形容词（如在not clea中）也能与介词（如在not on中）连用。

24.4 简单讨论在自然语言理解和情景分析问题中大致相似之处。

24.5 机器人*A*有词典和表24-2中的语义句法。解释一下机器人*A*如何建立一个积木环境，并且在那种方式下交谈以便一个有理解能力的机器人 *B*能推导出合适的语法（提示：通过让机器人*A*指向积木*B*同时发出声音*B*开始）。你必须假定机器人*B*具有什么能力？