计算机科学1990№6

分布式专家系统中的问题定义与分解

庄庆雨 （中国科学院数学所）

摘要 本文对分布式专家系统中的问题定义与分解作了一般性讨论。作者认为，文中对涉及的问题类型所作的一定限制、给出的定义和方法是否具有普遍性，仍需实践检验。

在分布式专家系统中，各个专家系统（以下简称专家）分布在计算机网络的各个节点上，它们可以合作为用户服务。分布式专家系统的核心问题是怎样组织专家来解决一个复杂的问题。通常的做法是把一个问题分解成若干个较简单的子问题，并把这些子问题分配给相应的专家来完成解题任务。一般来说，分布式专家系统接受的问题是十分复杂的，一个专家难以胜任整个解题任务，必须发挥各个专家的特长，共同解决问题。

目前，有各种各样的分布式专家系统，这些系统都针对特定的需要来建立问题分解与分配的方法。本文不打算介绍具体的方法，而从一般性角度来讨论分布式专家系统。首先探讨现实世界中的各种问题，给出问题的一般性定义，并在此基础上讨论问题的分解；然后讨论分布式专家系统中专家与问题的关系，并给出一个实例；最后介绍评价专家能力的方法。

1.问题的定义与分解

现实世界中的问题是多种多样的，根据问题的不同特点，我们可以例出以下几类。

1）解释性问题：对已知的数据进行分析和解释，确定它们的涵义，如自然语言理解，图像分析等领域的问题。

2）预测性问题：对过去和现在的已知情况进行分析，推断出将来可能出现的情况，如天气预报，人口预测等领域的问题。

3）诊断性问题：从已知的情况出发，推断出机体（或设备）失常的原因，如医学诊断，设备故障诊断等领域的问题。

4）设计性问题：从已知的约束条件出发，设计出一个系统的最佳配置，如工程设计，电路设计等领域的问题。

5）规划性问题：从初始条件出发，设计出达到一给定目标的动作序列，如机器人、交通运输等领域的问题。

6）其它。

尽管问题的类型很多，但它们都有一个共同的特点：从已知的一组数据出发，找到与这组数据相关的结果。基于这一点，我们可以给出问题的一般性定义。

定义1.1 一个问题是一个三元组（P，E，S），E和S是集合，P是从E到S的映射，即：

$$P: E \to S$$

记为P(E;S)。也称E为证据集合，S为解集合。

若对E中的某个元素a，S中存在元素b，满足P(a)＝b，则称a与b构成P的一个问题元，记为P(a→b)。可以认为，问题P是由多个问题元构成的。例如，程序A计算函数f：y＝x＋1，x是奇数，那么该程序所解决的问题 P：E→S，E是奇数集，S是偶数集，问题元是P(x→x＋1)。

我们可以把定义1.1推广到多个证据集的情况，这样，得到多元问题的定义。

定义1.2 一个n元问题是一个多元组

$\langle P;E_1,\cdots E_n;S\rangle$，其中，$E_1$，$E_2$，$\cdots E_n$是n个证据集合，S是解集合，P是映射：

$$P:\ E_1\times E_2\times\cdots\times E_n\to S$$

记为$P(E_1,E_2,\cdots,E_n;S)$。相应地，n元问题P的一个问题元记为$P(\langle e_1,e_2,\cdots,e_n\rangle\to s)$，$e\in E_i$，$s\in S$。

在医学上，给病人诊断病情可抽象为一个映射：$P:\ E_1\times E_2\times\cdots\times E_n\to I$，其中$E_1$，$E_2$，…，$E_n$是n个症状集合，I是疾病构成的集合。在实际问题中，当前提条件不确定时，得出的结论也是不确定的，为了刻画这种不确定性，我们可以定义不确定性问题。

定义1.3 一个不确定性问题是一个四元组$\langle P,E,S,B\rangle$，E是证据集合，S是解集合，$B=[0.0,\ 1.0]\subset R$，P是映射：

$$P:\ E\times B\to S\times B$$

它的意义是，当某个$a\in E$出现的可能性为b_1时，P(a)出现的可能是b_2。不确定性问题P的问题元记为$P((a,b_1)\to(b,b_2))$，$b_1,b_2\in B$。

映射是可以合成的，如果有映射$A:\ Y\to V$，$B:\ V\to Z$，那么我们能定义一个新的映射$C:\ Y\to Z$，它满足：

1) domain(A)=domain(C)，codomain(B)=codomain(C)；

2) $\forall y\in$domain(C)，$\exists z\in$codomain(C)，C(y)=z，并且A(y)∈domain(B)，B(A(y))=z∈codomain(B)。

合成映射C也记为$B\circ A$。把合成映射的概念推广，我们可以定义合成问题。

定义1.4 问题P^c是合成的，如果存在两个问题P^m和P^S，满足$P^c=P^m\circ P^s$，并称P^m是主问题，P^s是子问题。

合成问题中的子问题个数不一定只为1，当子问题个数大于1时，合成问题表示为：

$$P^c=P^m\circ\langle P_1^s,\ P_2^s,\ \cdots,\ P_r^s\rangle,$$

其中，$P^c:\ V_1\times V_2\times\cdots\times V_n\to S$，

$P_1^s:\ V_{11}\times V_{12}\times\cdots\times V_{1m_1}\to S_1$，$V_{1i}\in\{V_1,\cdots,V_n\}$，

$P_2^s:\ V_{21}\times V_{22}\times\cdots\times V_{2m_2}\to S_2$，$V_{2i}\in\{V_1,\cdots,V_n\}$

⋮

$P_r^s:\ V_{r1}\times V_{r2}\times\cdots\times V_{rm_r}\to S_r$，$V_{ri}\in\{V_1,\cdots,V_n\}$。

$P^m:\ S_1\times S_2\times\cdots\times S_r\to S$

并且$(\bigcup_{1\le i\le m_1}V_{1i})\cup(\bigcup_{1\le i\le m_2}V_{2i})\cup\cdots\cup(\bigcup_{1\le i\le m_r}V_{ri})=\{V_1,V_2,\cdots,V_n\}$。

例如，医生诊断一个病人是否患有肝炎时，不仅要进行问诊和面诊，还要取病人的血样本进行化验。定义下面几个问题：

P^c：诊断肝炎。

P_1^s：问诊，也就是询问病人的病情。

P_2^s：面诊，也就是观察病人的体态和面部

P_3^s：化验。

P^m：综合判断。

那么，$P^c=P^m\circ\langle P_1^s,\ P_2^s,\ P_3^s\rangle$。

一个合成问题有可能作为另一个合成问题的子问题，从而得到一个多级合成问题，可以用下式表示：

$$P_{k-1,i}^s=P_{k-1}^m\circ\langle P_{k,1}^s,P_{k,2}^s,\cdots,P_{k,m_k}^s\rangle\quad k\ge 1。$$

定义1.5 假设合成问题P^c可表示为$P^c=P_0^m\circ\langle\cdots,\ P_1^s,\ \cdots\rangle$，其中，

$$P_1^s=P_1^m\circ\langle\cdots\cdots,\ P_2^s,\ \cdots\cdots\rangle$$

⋮

$$P_{n-1}^s=P_{n-1}^m\circ\langle\cdots\cdots,\ P_n^s,\ \cdots\cdots,$$

则称P^c与P_n^s的合成距离为n，$t(P^c,P_n^s)=P_0^m P_1^m\cdots\cdots P_{n-1}^m$为从$P_n^s$到$P^c$的合成路径。

显然，合成问题P^c可解的必要条件是不存在一条合成路径t，某一个主问题P^m出现了两次以上。

定义1.6 设W是一个问题集，称问题P在W下是可分解的，如果能找到一个变换T，T作用于P后得到P^c，P^c中的每一个子问题都是W中的元素，并且$P=P^c$。这里，$(P=P^c)\Longleftrightarrow P:\ V\to S$，$P^c:\ V\to S$，并且对任意的$a\in V$，若P(a)有定义，那么$P(a)=P^c(a)$。

问题分解的目标就是要把一个可分解的问题P变换成$P^c=P^m\circ\langle P_1^s,P_2^s,\cdots,P_t^s\rangle$的形式，当然也包括对每一个可分解的子问题的分解，从而把一个较复杂的问题变成多个较简单的子问题的合成。

2.问题与专家

在上一节，我们把问题定义为一个映射，为了把这种定义下的问题与专家联系在一起，我们引入下面的定义。

定义2.1 设P是问题P：$V_1\times V_2\times\cdots\cdots\times V_n\to S$，如果对domain (P) 中的任一元素d，专家ES从d出发，经过推理后得到结果r，并且P(d)=r，则称ES解决了问题P，并称ES为P型专家。也称V_i为ES解决问题所需要的属性，V_i中的元素称为V_i的值。

很明显，专家的功能完全可以通过映射P来描述，用程序设计语言的术语来说，P定义了专家ES的语义。一个专家所能解决的问题往往不是一个，例如，有的专家不仅能给患肝炎的病人治病，而且也能给患肺炎的病人治病。假如一个专家ES不仅能解决问题P_1，而且也能解决问题P_2，则称ES为$(P_1\cup P_2)$型专家。推广到一般，我们能定义$(P_1\cup P_2\cup\cdots\cup P_n)$型专家。

定义2.2：设P是问题P_1：$V_1\times V_2\times\cdots\times V_n\to S$，它可分解成$P^c=P^m\circ\langle P_1^s,P_2^s,\cdots,P_t^s\rangle$，其中，

P_j^s：$V_{j1}\times V_{j2}\times\cdots\cdots\times V_{jt_j}\to S_j$，$V_{ji}\in\{V_1,V_2,\cdots\cdots,V_n\}$，$1\leqslant j\leqslant t_i$，

$1\leqslant i\leqslant t$，

P^m：$S_1\times S_2\times\cdots\times S_t\to S$

若存在专家组ES，ES_1，ES_2，…，ES_t，它们分别解决了问题P^m，P_1^s，P_2^s，…，P_t^s，则称ES，ES_1，ES_2，…，ES_t联合解决了问题P，并称ES为主专家，ES_i为子专家，$1\leqslant i\leqslant t$。

在现实生活中，若干个专家联合解决一个问题的情况是很多的。例如，某医院的内科病房里住着患有内分泌系统、消化系统、血液系统、呼吸系统四个方面疾病的病人，主治医生B_1，B_2，B_3，B_4分别擅长医治与内分泌系统、消化系统、血液系统、呼吸系统相关的疾病；当碰上疑难病和综合症时，由高级主治医生A领导大家进行会诊，确定病因和制定治疗方案。在有些情况下，并不需A，B_1，B_2，B_3，B_4都参加会诊，为了适应这种需要，我们对问题P的每一属性V_i进行扩充，使得V_i变成$V_i\cup\{\varepsilon\}$。下面给出与ε相关的定义。

定义2.3 设有问题P：$V_1\times V_2\times\cdots\times V_n\to S$，ES为P型专家，若$d_0=(a_1,a_2,\cdots,a_n)\in V_1\times V_2\times\cdots\times V_n$，$d_0$的第k个分量$a_k=\varepsilon$，$1\leqslant k\leqslant n$，ES从$d_0$出发得到r，$r\in S$，则称问题元$P(d_0\to r)$的解决与属性$V_k$无关。

定义2.4 设有问题P：$V_1\times V_2\times\cdots\times V_n\to S$，P可分解成$P^c=P^m\circ\langle P_1^s,P_2^s,\cdots,P_t^s\rangle$，ES，$ES_1$，$ES_2$，…，$ES_t$分别为$P^m$，$P_1^s$，$P_2^s$，…，$P_t^s$型专家。对于问题元$P(d_0\to r)$来说，经过分解后，$ES_k$得到问题元$P_k(d_k\to x)$，$1\leqslant k\leqslant t$，$d_k=(\varepsilon,\varepsilon,\cdots,\varepsilon)$，x为任意值，则称问题元$P(d_0\to s)$的解决与$ES_k$无关。

一个例子

UNION是一个分布式专家系统。在UNION环境下，有一群专家分布在计算机网络的各个节点上，它们可以联合起来为用户服务。UNION呈树型结构，每个叶节点是一个专家，每个非叶节点是一个专家管理员，专家管理员负责管理以它为根的子树中的全部专家以及较低级别的专家管理员。当一个专家管理员接受一个问题后，首先检查该问题能否由本管理员属下的专家来完成。如果不能，则转交给上级管理员；如果能，则制定解题规划，并给各下属成员分派任务。这里，上级管理员相当于主专家，它所解决的问题是主问题；下级成员（管理员或专家）相当于子专家，它所解决的问题是子问题。

我们可以用一个多元组来描述UNION的静态结构，即：(N, C, M, ⩾, E, F_1, F_2, F_3)，其中，

1)N是网络节点构成的集合；

2)C是关系，即：$C\subseteq N\times N$，它描述节点之间的连接关系，如果$(n_1,n_2)\in C$，则从n_1可以发送信息到n_2；

3）M是管理员集合；

4）$\geqslant$是M上的偏序关系，如果$m_1 \geqslant m_2$，则m_1是m_2的上级；

5）E是专家组成的集合；

6）F_1是映射，即：F_1：$N \to 2^M$，它描述管理员在每一个节点上的分布情况；

7）F_2是映射，即：F_2：$N \to 2^E$，它描述专家在每一个节点上的分布情况；

8）F_3是映射，即：F_3：$M \to 2^E$，它描述管理员与专家的关系；如果$m \in M$，那么，$F_3(m)$是m的下级成员集。

UNION支持三种用户要求。1）指定某个专家为自己服务。例如，病人王某患有青光眼，他知道UNION中的专家A治好青光眼的成功率最高，故要求UINON指派专家A为自己服务。2）问题明确．但专家不明确。例如，病人陈某患有肝炎，他不知道UINON中的各专家情况，只好向UNION提出要求，由UNION为他指派专家。3）问题不明确，专家也不明确。例如，病人张某突然发热、呕吐、不知自己患了什么病，只好向UNION陈叙自己的病情，由UNION来组织专家为他服务。对于第一种要求，不存在问题分解的过程，但对于后两种要求，问题分解比较复杂。

在具体实现时，我们为每一个管理员设置了一个类似框架的结构，框架的一般结构是：

```
Frame <manager-name>
  problem-type   :  <……>
  trigger-data   :  <……>
transform-rule   :  <……>
synthesis-rule   :  <……>
    member       :  <……>
     other       :
```

problem-type记录管理员所能解决的问题；trigger-data记录的数据是管理员解决问题所需的；transform-rule记录一个规则集，这些规则用来分解一个复杂的问题；synthesis-rule记录一个规则集，这些规则用来综合下级成员得到的结果；member记录管理员所领导的下级成员。

每个基层管理员对他所管辖的每个专家保存一张专家登记表，这张表的内容有：

1）专家名称；

2）能解决的问题；

3）成功率；

4）接口方式；

5）运行效率；

6）当前状态。

当某个管理员接受一个问题后，检查是否与problem-type中的内容匹配。如果不匹配，请求上级管理员帮助解决这个问题；如果匹配，向用户询问所需的数据，并使用transform-rule中的规则对问题进行分解，得到若干个子问题，然后把这些子问题分派给member中的下级成员来完成；当所有的下级成员都完成解决问题的任务后，从下往上使用synthesis-rule中的规则综合所得结果。

3.专家能力的评价

在UNION中，当管理员接到一个问题后，首先对问题进行分解，然后把各个子问题交给下属专家来解决。对于某个子问题P^S，当有多个专家都能解决P^S时，该把P^S分派给哪个专家呢？很显然，管理员应指派能力相当的专家来解决P^S，为此，管理员必须了解下属专家的能力大小。一种描述专家能力的方法是使用专家性能表。性能表的内容可以是：

1）处理对象；（例如：胃病、肺病）

2）处理能力；（例如：诊断、处方）

3）推理方向；（例如：正向推理）

4）解释能力；（例如：how、why）

5）接口环境；（例如：窗口、英语、汉语）

6）运行时间；（例如：运行一次十分钟）

7）运行费用；（例如：运行一次100元）

8）其它。

但仅有性能表的内容是不够的。例如，有两个专家ES_1、ES_2都能诊断青光眼，但由于知识的来源不一样，它们对某些问题的见解会有很大的差异。当然，我们可以采用冗余方式来解决问题，但当专家资源紧张时，这种方式不再适用。一种合理的方式是指派能力强的专家来解决重要问题，而能力较差的专家来完成一些辅助性工作。

专家的能力由两个因素决定：①具备的知识；②应用知识的能力。对于庞大的知识库和复杂的推理机制，我们怎样对专家进行比较呢？当然，在一些简单情况下，我们是很容易做的。例如，专家ES_1和ES_2的推理

机制一样，如果ES_1的知识库包含了ES_2的知识库，那么，ES_1的能力一般超过了ES_2。我们注意到，一个专家的能力可以体现在它所解决的问题上，例如，医生甲能治疗患肝炎的病人，医生乙不能，因此，就治疗肝炎这个问题来说，医生甲的能力超过了医生乙。下面，我们用专家所解决的问题来定义专家的能力。

定义3.1 一个专家的能力是一个集合，该集合的元素是专家所能解决的问题元，即：

ability(ES)＝{P(d→r) | ES能解决P(d→r)}。

一般来说，专家解决问题的正确性是不同的，为此，我们引入一个正确因子f来评价专家解决问题的好坏。这样，专家的能力定义为：

ability(ES)＝{〔P(d→r)，f〕| ES解决P(d→r)，f∈〔0,1〕}。〔P(d→r)，f〕的含义是ES解决问题元P(d→r)的正确性为f，若f＝1，则表示专家完全正确地解决问题；若f＝0，则表示专家得到的结果完全与事实相反。

已知ES_1和ES_2的能力分别为ability(ES_1)和ability(ES_2)，那么有：

1）如果ability(ES_1)＝ability(ES_2)，那么ES_1与ES_2的能力相同；

2）如果ability(ES_1)⊂ability(ES_2)，那么ES_1的能力弱于ES_2；

3）对于问题元P(d→r)，若〔P(d→r)，f_1〕∈ability(ES_1)，〔P(d→r)，f_2〕∈ability(ES_2)，并且$f_1>f_2$，那么ES_1在解决问题元P(d→r)时能力超过了ES_2。

使用集合来描述专家的能力有很大的困难，这不仅因为我们难于枚举出ability(ES)中的每一个元素，而且也由于存贮这样庞大的集合是很困难的。一种可行的方法是找到一条件表达式c，使得对任意的问题元P，p满足c⟺P∈ability(ES)，为了找到条件表达式c，我们可以从ability(ES)出发，使用归纳法得到c，为此，不仅要求ability(ES)中问题元的表示合理，而且也要求ability(ES)中的问题元个数充分大。

为了获得ability(ES)中的元素，我们可以采用下面两种方法：

1）考试法 假设ES_1，ES_2，…，ES_n是n个同型专家，选取与它们相关的问题元P作为考卷，并把P交给各个专家来完成，记录下各个专家的解题情况。使用这种方法时，考卷的选取一定要得当，以便一份考卷能覆盖多个问题元。

2）积累法 当专家解决实际问题时，记录下专家解题情况，随着专家解决问题的增多，ability(ES)会变得越来越大。

当使用上述两种方法来获得ability(ES)时，ability(ES)逐步变大，因此，表达式c也应由粗到精一步一步地发生变化。假如在t_1时刻，我们从ability(ES)归纳出表达式c_1，在t_2时刻，ability(ES)扩充了新的元素P，那么，我们必须从c_1和P出发，产生新的表达式c_2，以使ability(ES)∪{P}满足c_2。

前面我们使用了ability(ES)来定义专家ES的能力，根据专家的能力大小，我们能够合理地给专家分派任务。评价专家能力的另一种方法是知识基方法，下面给予简单介绍。

我们知道，当人们解决实际问题时，需要使用各种知识，例如，一个内科医生给病人治病时，要使用病理知识，诊断学知识，药理知识等等。我们把专家在解决问题时所使用的某类知识称为知识基。病理知识，诊断知识，药理知识都是知识基的例子。假设问题P的解决需要使用知识基B_1，B_2，…，B_n，那么，当问题P提交给专家ES时，ES能够解决问题P的必要条件是具有B_1，B_2，…，B_n。

假设ES_1和ES_2的知识基集合分别为{B_{11}，B_{12}，…，B_{1n}}，{B_{21}，B_{22}，…，B_{2m}}，那么，

1）如果{B_{11}，B_{12}，…，B_{1n}}＝{B_{21}，B_{22}，…，B_{2m}}，则ES_1与ES_2的能力相当；

语义数据模型(2)

Joan Peckham　　Fred Maryanski

2 有代表性的语义模型

以下将讨论一些语义模型，用以表示各种主要的概念建模方法，其中涉及了各模型的主要特征，并用第1节讨论的比较参数进行了分析。

2.1 实体-联系模型

实体-联系(E-R)模型[Chen 1976]是一个早期的语义数据模型，它统一了传统模型的各种特征，有利于处理各种语义信息。顾名思义，两个基本建模结构是实体和联系。从概念上看，被建模的企业模式可视为由图形表示的一组实体类型和联系类型（与网状模型类似）；从表达层次上看，实体和联系实例的信息结构与关系很相像。

前面图6表示了一部分图书馆数据库的E-R图，前图8描述了实体属性和联系类型的属性。从很抽象的观点来看，如图8，联系存在于类型和其属性之间。虽不像在实体-联系模型中所讨论的，但类型及其属性间的联系对数据建模者提出了与聚集相同的抽象概念。

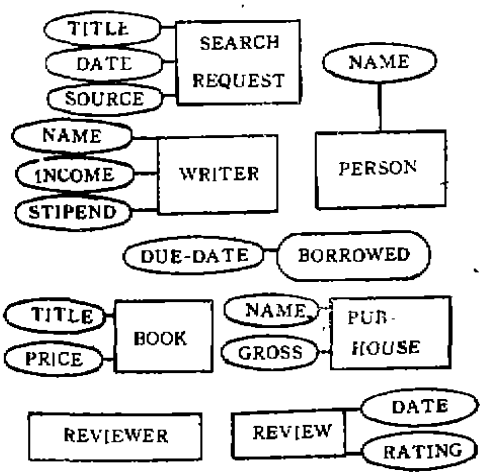

图8 E-R模型中的类型和属性

2) 如果$\{B_{11}, B_{12}, \cdots, B_{1n}\} \subset \{B_{21}, B_{22}, \cdots, B_{2m}\}$，则$ES_1$的能力弱于$ES_2$；

3) 如果$\{B_{11}, B_{12}, \cdots, B_{1n}\} \cap \{B_{21}, B_{22}, \cdots, B_{2m}\} = \phi$，则$ES_1$与$ES_2$不可比。

本文从一般化角度讨论了问题的定义与求解，并讨论了评价专家能力的方法。应该看到，本文对所涉及的问题类型作了一定的限制，所给出的定义、方法是否具有普遍性，还需实践的检验。最后，感谢陆汝钤教授在我研制UNION过程中所给予的精心指导和帮助。

参考文献

[1] 陆汝钤，"分布式专家系统"，新一代计算机文集，P114—124，1988.

[2] Arch W. Naylor, "On decomposition theory: generalized dependence". IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Vol. SMC-11, No.10, 1981.

[3] David M. Himmelblau, "Decomposition of Large-Scale Problems". North-Holland Publishing Company, Amsterdam, 1973.

[4] Michael N. Huhns, "Distributed Artificial Intelligence". Morgan Kaufmann Publishers, Inc., Los Altos, California.

[5] 庄庆雨，"分布式专家联合系统UNION中的任务分解与分配"，89年全国人工智能及其应用学术会议文集，P189—195.

[6] 管纪文、刘大有等，"知识工程原理"，哈尔滨大学出版社