计算机科学 1998Vol.25№.2

软件 Agent 的构筑*)

The Construction of Software Agent

刘 弘 曾广周# 林宗楷

(中国科学院计算技术研究所 CAD 开放实验室 北京 100080)

(山东工业大学计算机科学技术系)#

摘 要 This paper analyses the intelligent activities of mankind and the four levels of cognitive process. Based on the analysis, the basic structure of software agent and the organized framework of the multi-agent system are proposed.

关键词 Software agent, Intelligent activity, Information process, Basic structure, Organized framework

1 引言

软件 agent 的研究已经在计算机科学的各个领域引起极大的兴趣。由于软件 agent 的研究者来自许多不同领域,使得软件 agent 的含义也具有多重性,但有一点是明确的,即软件 agent 是计算机程序,具有自主性、协作性,并且能帮助人类完成一些特定的任务。

分布式人工智能的研究对软件 agent 采用了拟人的描述方法,即软件 agent 是组成 agent 社团的成员,agent 是包含了信念、承诺、义务、意图等精神状态的实体[1]。而软件工程的研究则从模型角度考察了 agent,认为面向 agent 的软件开发方法是为了更确切地描述复杂的并发系统的行为而采用的一种抽象描述形式,与象面向对象一样,是观察客观世界及解决问题的一种方法[2]。

无论采用哪种软件 agent 的定义,都存在着如何构造软件 agent 的问题。构造具有自主性、协作性及智能的软件 agent,是人们最终的研究目标,而对人类智能行为的模拟,则是构筑软件 agent 的必由之路。本文提出了一种模拟人类智能活动过程构筑软件 agent 的方法。

2 Agent 的基本结构

2.1 人类智能的活动过程

从认知心理学的角度来看,人类智能包括感觉、注意、表象、学习、记忆、思维和语言等,均可理解为信息的获取、存储、加工和使用的过程,即人类智能的活动过程都是信息加工过程,人脑是类似于计算机的信息加工系统。包括人和计算机在内,信息加工系统都是由感知器、效应器、记忆和加工处理器组成的。其一般过程为:感知器接收外界信息,效应器作出反应。信息加工系统都以符号结构来标志出输入和输出,记忆可以储存和提取符号结构,如图 1 所示。

图 1 信息加工系统的一般结构

在信息加工系统中,环境作为信息源,可以是人、机器、自然界的物体等等,并以各种符号或信号的方式表现出来。人通过感官接受外界的刺激,获得信息,经过大脑的加工处理,然后通过手、足、身这些效应器官输出,反作用于环境,表现为以主体为中心对客体(环境)的能动作用。

人对于外界(包括物理环境和其它人)的信息获取,按人所处的角色不同可分为两类:一类是人主动地用器官去感知,另一类是外界作用于人的器官,迫使人去感知,然后由大脑去理解。即一种是"感知"方式,另一种是通讯方式。

在人类社会中,人观察问题的视角及其所采取

*)国家九五攻关项目及山东省自然科学基金资助

的行为往往与其背景、职业等有关。不同职业的人在观察同一件事物时，往往会产生不同的结论，并引发不同的行为。同时，由于人类社会是一个有机结合的整体，社会中的成员必须受到一定的约束，明确自己的职责，为完成一个共同的目标而努力。

2.2 Agent 的基本结构

由人类智能活动的过程我们知道，一个可以进行智能活动的实体，应该具有与外界交互的感知器、通讯机制，及对信息进行存储加工的信息处理器、记忆库，同时还应该具有根据共同的目标及自己的职责所产生的目标模块及反作用于外部环境的效应器。Agent 是一个具有信息处理能力的主动实体，其基本结构也应包含上述模块，如图 2 所示。

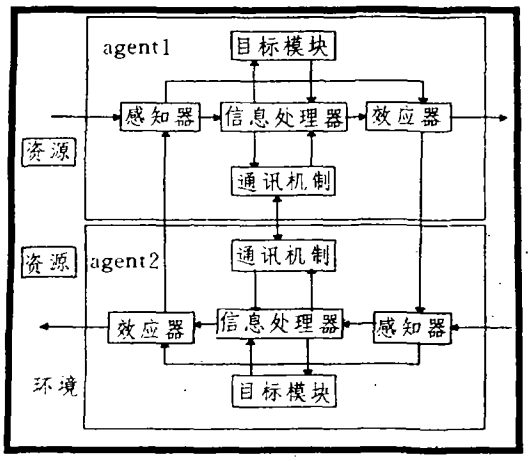

图 2 Agent 的基本结构

定义 1 Agent 可以表示为一个六元组：(Aid，目标，感知器，通讯机制，信息处理器，效应器)，即

〈Agent〉::=〈Aid〉〈目标〉〈感知器〉〈通讯机制〉〈效应器〉〈信息处理器〉
〈Aid〉::=〈Agent 名〉
〈目标〉::=〈任务表〉
〈感知器〉::=〈激活条件〉〈信息流〉
〈通讯机制〉::={〈通讯原语〉(〈通讯内容〉)}
〈通讯原语〉::=〈Command〉|〈Require〉|〈Accept〉|〈Reject〉|〈Inform〉|〈Cancel〉
〈通讯内容〉::=〈发送者〉〈接收者〉〈时间〉〈信息流〉
〈效应器〉::={〈事务处理名〉(〈事务处理描述〉)}
〈事务处理描述〉::=Begin
{〈TPDL 语句〉;}
End

Agent 所采取的一切行为都是面向目标的。目标以任务表的形式来表示，初始由用户静态建立，然后通过通讯而动态地改变。任务表只指明 agent 必须做的事，并不是怎样做。任务的实现由效应器及与其它 agent 合作的形式来完成。

通讯是交互的手段，由通讯原语及通讯内容两部分组成，发送任务、表达各 agent 对任务的态度及传递被处理的信息。

感知器表示 agent 的感知能力。当触发条件被满足时，它被激活并接收外部信号/信息流。

Agent 有两种行为：认知行为及效应行为。内部执行机制实现认知行为，它们改变 agent 的内部状态及 agent 的知识库。其它 agent 感觉不到认知行为的执行。

效应行为被效应器执行。我们用一组 TPDL 语句描述效应行为[3]。效应行为的执行改变系统的状态，并且其它 agent 能感觉到。

信息处理器的详细描述见下节。

3 Agent 的信息处理器

3.1 信息处理过程的四层结构

任何一个系统处理外部信息的过程都是将其转换为内部形式进行的。人类的大脑进行信息处理的过程可以分为四层结构，如图3所示。

1. 信号级：认知过程的第一级是通过感知器与环境直接交互，感知外部动态信息。在这一级，感知器在物理级感知外部信号/信息流，这些原始信号通常是很粗糙的，不是与事务的名称或意义联系起来的符号或概念，必须经过特征级对这些信号进行过滤、聚合后才能影响人的行为或活动。这一级是外部世界信息流人大脑的唯一渠道，其它更高级的信息

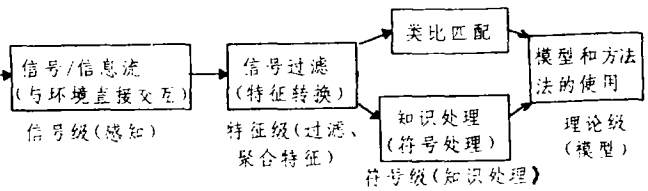

图 3 信息处理过程的四层结构

处理工作都是以这一级为基础的。

2. 特征级：来自第一级的信号/信息流在特征级被过滤成为特征值或特征符号(如语言理解中的音素)，这些值或符号可以被存储起来，直接用来引起行为，即人类的直觉反射，也可以被进一步聚合成第三级的基本建筑块(如自然语言中的词)。这一级处理的重要结果是形成第三级处理所需要的符号，而这些符号是可以与客观世界的对象联系起来的有意义的概念或名称。

3. 符号级：在这一级，信息流经过前面两级的处理已被分类并形成符号。该级主要利用模糊规则、限制等，在高度抽象及压缩的基础上进行高层次的行为计划及信息存储。人们通常进行的归纳及推理，都是在这一级上实现。推理除了对当前的行为进行决策以外，还形成高度聚合的知识-知识块，并存储在该级的知识库中。归纳及推理产生的知识块要经过多次去粗取精及实践检验，才转入理论级。

4. 理论级：这是认知层中最复杂的级，代表了人类的文化和遗产的继承。这里，现实世界以精确的方式被建模，其基本元素是可以独立处理事物的知识块。知识块与一般的知识不同之处在于它不是简单的符号或规则，而是由问题的初始状态到目标状态的路径。这些知识块有两个来源，一个是在符号级归纳推理形成的知识块；另一个是通过学习而得到的。例如我们可以用优化和决策理论的模型及公式，处理统计资料，而这些模型及公式是我们从前人的研究及实践中学习来的，不需要经过逐步加工。经过对大脑重复刺激，这些知识块被保存在记忆库中。在处理问题时，人们往往先考虑查找与要处理问题具有类似特征的知识块，然后用知识块所对应的方法进行处理。只有当的确没有充足的知识可用时，才采用低层的内部规则来对动作的选择作出决策。

3.2 Agent 的信息处理器

与人类的大脑类似，agent 的信息处理器是体现 agent 智能行为的最重要的部件。由人类信息处理的分层结构可知，agent 的信息处理器应由信号/信息过滤器、控制器、符号推理机制、类比匹配机制、内部执行机制及知识库组成。其中，知识库中包含两类知识，一类是规则，另一类是知识块。信息处理器在接收到信号/信息后，先对其进行过滤、抽象、聚合，使其形成可以与客观世界的对象联系起来的有意义的符号，然后由类比匹配机制将这些符号及特征与知识库中的知识块进行模糊匹配。如果能查找到高度匹配的知识块，相应的知识块被用来处理信息并产生决策；如果只能部分匹配，则控制器驱动内部执行器将被匹配的部分知识作为符号，然后运用推理机制及知识库中的规则处理信息，并形成新的知识块；如果的确没有可用的知识块，则知识库中的规则被用来处理信息。知识库中的规则及知识块随着问题的处理不断被添加及更新(见图4)。

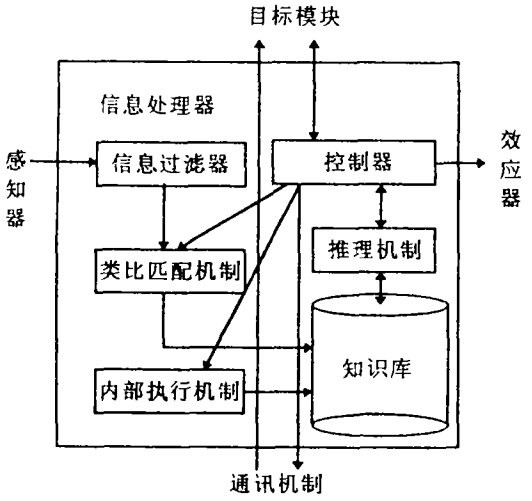

图4 信息处理器结构

综上所述，agent 的信息处理器由信息过滤器、控制器、类比匹配机制、内部执行机制、推理机制及知识库组成，即：

〈信息处理器〉::=〈信息过滤器〉〈控制器〉〈类比匹配机制〉〈内部执行机制〉〈推理机制〉〈知识库〉
〈信息过滤器〉::={〈特征值〉}
〈控制器〉::={〈控制命令〉}
〈类比匹配机制〉::=〈映射规则〉〈匹配度计算方法〉
〈内部执行机制〉::={〈内部执行动作〉}
〈内部执行动作〉::=Begin
{〈TPDL 语句〉;}
End
〈推理机制〉::={〈推理规则〉}
〈知识库〉::={〈领域知识〉〈控制知识〉〈问题求解知识〉〈通讯知识〉}
〈领域知识〉::={(对象,属性,值)}
〈控制知识〉::={〈产生式规则〉}
〈产生式规则〉::=〈规则名〉〈条件〉〈动作〉〈确证度〉
〈问题求解知识〉::={〈与或解图〉}
〈通讯知识〉::={接收信息〉〈当前状态〉〈动作〉}

Agent 的信息过滤器包含一组用户定义的信息特征，只有与 agent 局部相关的信息才能通过过滤器，agent 的控制器根据其推理机制及知识产生的结果进行下一步动作。Agent 的内部执行机制根据控制器的命令改变 agent 的内部状态。

Agent 的知识库中有四类知识，领域知识、控制知识、问题求解知识及通讯知识。

·领域知识是一种描述性知识，表示对象及概念的特征及其相互关系，也称为事实性知识，用三元组(对象，属性，值)的形式表示，对应人类处理问题的第二层知识。

·控制知识是一种判断性知识，表示与领域有关

的问题求解知识，用产生式规则表示，对每个 agent 来说，有其私有知识库与公有知识库，规则名可以体现规则所属知识库，对应信息处理的第三层知识。

·问题求解知识是一种过程性知识，表示问题求解的控制策略，即如何应用判断性知识进行推理的知识，用与或图表示的解图描述，对应大脑进行信息处理的第四层知识。

·通讯知识属于上述三种知识的混合知识，是处理人类交互及 agent 协作的知识。对于人类交互来说，由于存在自然语言的理解问题，所以是一种相当复杂的知识。而对于软件 agent 来说，由于所采用的通讯原语是规范化的，且数量也很有限，所以相对地来说比较容易。我们采用表的形式来表示通讯知识，如表1所示。

表1 通讯知识表

接收通讯原语	表 示	接收 agent 动作
COMMAND	接收者必须按命令要求去做	接收全部信息，并将任务加入目标模块中
REQUIRE	接收者可以根据自己当前的状态及能力决定是否做	当前忙，则根据自己的职权范围，命令或请求其它 agent 去做；不忙，则接收全部信息，向发送者发 'ACCEPT' 信息并执行任务
ACCEPT	发送者已接受请求	等待发送者完成任务后的结果
REJECT	发送者因忙或其它原因不能接受任务	转而请求其它 agent 执行任务
INFORM	发送者已安排其它 agent 执行任务	等待发送者完成任务后的结果
CANCEL	撤消通讯	接收者恢复接受通讯原语前的状态

4 多 agent 的组织结构

Agent 间的关系类型可分为行政管理关系和问题求解关系。

4.1 行政管理组织结构

在现实世界中，领导者总控管理机制，向其下属的职能个体分派互不相同的职能；职能个体无须考虑其上的全局性问题，只需按领导者的要求处理好本职的行为，以及与相关个体之间的关系，倘若所有职能个体都如此正常地运转，整个系统自然协调一致了。上述基本原理对 agent 团体也是适用的。行政管理关系由用户在系统建模时确定，反映了一种静态的组织结构。

全部 agent 构成一个 agent 团体。在团体中，根据 agent 的工作范围，分为若干个项目组，每个项目组都由某些成员和相应的约束关系构成。项目组内部，根据 agent 的职能，将 agent 分为三种类型：管理 agent、功能 agent、应用 agent。三种类型的 agent 分别处于不同的层次，形成一棵 agent 树。整个 agent 团体构成一个森林。类型及层次关系限定了树上各 agent 间的相互地位、权限等。根据这一特征，各层 agent 的职能为：

·管理 agent：担负管理职能。每个项目组中只能有一个管理 agent，与本项目组中的功能 agent 一起形成管理群体。管理 agent 的行为表现为对问题的决策、查询及对其他 agent 的控制和监督。管理 agent 掌握其全部下层 agent 的外部特征，作为管理的内部知识存在内部知识库中。项目组中每增减一个 agent、或 agent 的外部特征发生变化，管理 agent 的这部分知识都要作相应的改动。不同项目组之间的成员进行协作要通过各自所在项目组的管理 agent 进行协调。

·功能 agent：具有通用任务实施职能。功能 agent 可以协助管理 agent 完成一些系统任务，如通讯管理、事件监视、文档生成等，是一类不受应用领域影响的工具 agent，这些 agent 与管理 agent 一起，形成 agent 团体的基本构件，不受用户的影响。

·应用 agent：具有特殊领域的事务处理功能。应用 agent 之间有层次关系及协作关系，子应用 agent 可以继承父应用 agent 的特征。

管理 agent 与功能 agent 构成一个管理群体，它们的层次关系是确定的，只有应用 agent 之间的层次关系由用户来规定。

用 M_i，F_j，A_k 分别表示管理 agent、功能 agent、应用 agent。其中，i 是单字符，而 j 和 k 是字符串。i 表示管理 agent 所在的项目组，j 表示功能 agent 所属的项目组及其序号，而 k 除了体现应用 agent 所在的项目组外，还能体现应用 agent 之间的层次关

系。Agent 的一个组织关联图如图5所示。

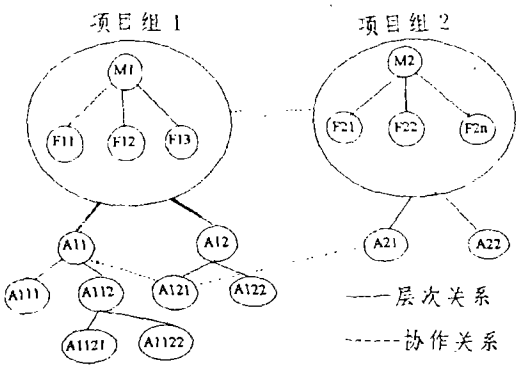

图5 Agent 的组织关联图

4.2 问题求解组织结构

问题求解组织结构是 agent 为了协同完成一个任务而形成的一种动态组织结构。各 agent 之间的关系是动态变化的，一但具体任务完成，这种关系就自行消失。当新的任务到来后，会形成由其它 agent 构成的新的问题求解结构。例如，对于合同网式的问题求解结构[4]，只有当某个 agent 接受任务时，它才有可能成为 manager，其它 agent 根据自身能力因素决定自己是否成为该 manager 的 contractor，从而构成 manager 与 contractor 的关系。任务完成，这种关系也随之消失。

结束语 本文分析了人类智能行为的活动过程及信息处理的四个层次，并在此基础上提出了软件 agent 的基本结构及多 agent 系统的组织结构。目的是通过对人类智能活动的模拟，构筑具有自主、协作及智能的软件 agent，使计算机更好地为人类服务。我们的下一步任务是开发相应的构筑环境。

（下转第48页）

（上接第81页）

中进一步发展了这个思想用于面向对象多数据库系统（主要是增加了导航查询的局部处理代价的获取能力）。很显然，使用这种方法必须知道每个具体局部系统所支持的存取方法，[8]中认为这明显地违背了局部自治性。因此，在[8]中提出了一组方法，其中之一是查询取样技术。其中心思想是对于局部场地的查询根据查询的特点、操作数的基数、可用的索引信息，将局部系统的特点进行分类。对于每一类查询设计一个取样查询以获得这一类查询的参考代价参数。使用这种方法的关键是查询的分类技术。目前该项研究的局部系统全部为关系数据库。

4.3 探索策略

对于一个给定的查询，其全局执行计划不是唯一的，而且往往可以有大量的执行计划可以选择。用代价模型采用穷举法分别地评价每一个执行计划，从中选择一个最优的执行计划将极大地恶化查询执行效率。因此需要一些探索策略，来尽可能减少枚举的次数。由于异构性，各局部场地的处理速度、语言的表达能力相差很大，因此这样的探索策略是很难设计的。

目前关于这方面的研究非常少。[7]中列举了一些在设计这样的探索策略时应考虑的一些问题和可选的策略，包括：(1)贪婪地分配投影、选择和利用半连接策略；(2)在有大量的连接次序可以考虑时，采用带参数的爬山式启发策略；(3)过滤不能在局部场地执行的操作（如连接）。其中策略1，2是传统分布数据库系统探索策略的延续，策略3只是异构分布环境下应考虑的最明显的策略。而对于异构分布环境下导航查询应采取什么样的探索策略成为今后应考虑的重要问题。

5 局部查询处理

在局部处理步骤执行的单场地查询必须经局部优化来选择有效的存取路径；经优化的单场地查询被转化为局部数据库管理系统的数据语言。

目前关于查询转换的研究较多，而关于局部优化的研究较少。其原因是局部优化也面临一个与全局优化同样的问题，即无法取得需要的全部优化参数。局部优化是多数据库系统的设计者而不是局部数据库管理员解决的问题，有关的研究可参见[12]。

总结 由于异构性和自治性，多数库系统中的查询处理是十分复杂的，但目前关于这个问题的研究还比较少，而关于引入面向对象技术的多数据库系统中的查询处理的研究就更少。尽管已有一些研究探讨了查询修改、查询优化、局部查询处理这些问题的解决方案，但对于每一个问题都没有一个理想的解决方案，因此在国家863资助项目“基于 CORBA 的面向对象的多数据库集成平台系统 SCOPE/CIMS”中，关于这些问题的解决策略是我们正在进行的研究内容之一。（参考文献共14篇略）

图根据 WorstParts 的内容在关键决策点上选用不同的变化。

显然，评测方法的性能依靠代价函数的实现。代价函数是衡量方案质量的尺度，如果寻找一个准确的尺度十分困难，就会影响系统性能的提高。另外一种方法是算法 3，让专家决定设计方案的哪一部分应该改善。假设 D 是设计方案的集合，suggest 是专家在检查过以图形方式显示的设计方案后输入的建议。SetInitial(Suggest)在指定的决策点采用新的启发性法则。ExpertExamine(d_t)以图形方式显示 d_t，然后返回专家可能提出的改善建议。算法 3 如下：

```
for(each p∈P)do
begin
  suggest=NULL;
  D=NULL;
  InteractiveReDesign(suggest,D);
end

InteractiveReDesign(suggest,D)
begin
  SetInitial(suggest);
  dt=Apply(DS,p);
  D=Append(dt,D);
  suggest=ExpertExamine(dt,D);
  If(suggest<>NULL)
    Interactive ReDesign(suggest,D);
end
```

算法 2 与算法 3 实现了两种可以互相替代的评测方法，一种是自动方法，一种是手工方法。使用者可以任选其一。

2.3 准确性与一致性的评测。这两种评测方法比较一致，因为它们都依赖于上下文相关的信息，如域事实、问题描述。准确性指设计方案满足问题描述提出的要求；一致性指设计方案与域事实、设计方案的各个组成部分之间没有冲突。

假设我们要构造一个系统 DS。P 是要解决的问题的集合。T 是域事实，d 是一个设计方案。Sys 是 S 的一个版本。$S_{incorrect}$，$S_{inconsistent}$分别是被报告错误的集合，它们为空时算法结束。应用域事实与输入事实，从非空的 $S_{incorrect}$或 $S_{inconsistent}$开始提纯 DS。Create(T)应用所有领域专家提供的和说明书中的知识，按照产生式规则以过程的形式生成 DS 的初始版本。Apply(Sys,p)中，Sys 用于构造一个针对 p 的设计方案。Refine(Sys,d,T,$S_{incorrect}$,$S_{inconsistent}$)根据这些变量来修正 Sys 并产生 DS 的新版本。CorrectnessCheck(d,p)针对 p 检查 d，把发现的错误记录在 $S_{incorrect}$中。ConsistencyCheck(d,T)针对 p 检查 T，把发现的错误记录在 $S_{inconsistent}$中。SystemDesign(Sys,p)是一个递归定义的过程，用于提纯 DS 直到 DS 的某一个版本为问题 p 生成的设计方案中没有任何不正确或不一致的地方。其算法 4 如下：

```
Sys=Create(T);
for(each p∈P)do
  SystemDesign(Sys,p);
SystemDesign(Sys,p)
begin
  d=Apply(DS,p);
  Sincorrect=CorrectnessCheck(d,p);
  Sinconsistent=Consistencycheck(d,T);
  If(Sincorrect<>NULL OR Sinconsistent<>NULL)
  begin
    Sys=Refine(Sys,d,T,Sincorrect,Sinconsistent);
    SystemDesign(Sys,p);
  end
end
```

三、结论与进一步的研究

我们致力于寻找一种方法，能指导专家系统设计者客观与全面地完成设计任务，并能够消除知识获取这一瓶颈问题，以及设计时的主观倾向性。我们已经看到评测方法所起的作用。然而还需要把它应用到更多的领域，扩充更多的评测方法。我们将进一步研究评测方法在知识库建模，专家系统验证、优化，机器学习等方面的应用。

参考文献

[1] Weiss S. M. et al. , EXPERT: A System of Developing Consultation Nodes, Artificial Intelligence, 1979, 11

[2] Wong A. K. C. et al. , A Gray-Level Threshold Selection Method Based on Maximum Entropy Principle, IEEE Trans. on Systems, Man and Cybernetics, 1989

（上接第 28 页）

参考文献

[1] Y. Shoham, Agent-oriented programming, Artificial Intelligence, 60, 1993

[2] R. Goodwin, Formalizing properties of agents, TR CMU CS-93-159, Carnegie Mellon University, Stanford, 1992

[3] Zeng Guangzhou, Sun Boqi, Requirements engineering method based on system simulation, In: Proc. of the Changsha International CASE symposium (CICS'95), Changsha, China 1995

[4] R. Davis, R. Smith, Negotiation as a metaphor for distributed problem solving, Artificial Intelligence, 20(1), 1983