南京大学
人工智能学院
SCHOOL OF ARTIFICIAL INTELLIGENCE, NANJING UNIVERSITY

第三部分：通信与合作

章宗长

2023年3月28日

内容安排

通信

- 言语行为
- 通信方式
- **基于消息的Agent 通信语言：KQML**
- 基于消息的Agent 通信语言：FIPA ACL

KQML

- 知识查询与操纵语言：Knowledge Query and Manipulation Language
- KQML对信息定义了公共的格式，每条消息可认为是一个对象，包含一个语用词以及若干参数
  - 语用词：可看作消息的类
  - 参数：可看作属性/值对
- 一个KQML消息的例子：

```
(ask-one
   :content (PRICE IBM ?price)
   :receiver stock-server
   :language LPROLOG
   :ontology NYSE-TICKS
)
```

参数

语用词，表示询问，且需要给出一个回答

content：消息的内容
receiver：消息的接收方
language：表达内容的语言
ontology：消息中使用的术语本体

KQML – 部分语用词

S：发送方　　R：接收方　　C：信息内容　　VKB：虚拟知识库

语用词	含义
ask-one	S想从R中知道关于问题C的一个答案
ask-about	S想知道R的VKB中所有相关的内容
ask-if	S想知道R的VKB中是否有问题C的答案
evaluate	S希望R评价（简化）C
eos	前一个询问响应流结束
reply	传输一个预期的回答
tell	S向R声称C在S的VKB中
stream-about	ask-about的多个响应形式

KQML – 部分消息参数

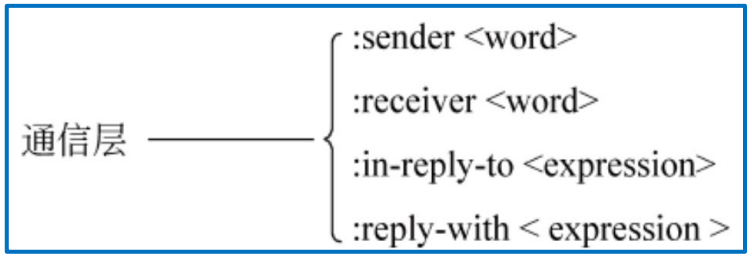
参数	含义
:content	消息内容
:force	消息的发送者是否会拒绝相信消息的内容
:reply-with	发送者是否期待回答，如果是，给出回答标识符
:in-reply-to	参考“:reply-with”参数
:sender	消息发送者
:receiver	消息的预期接收者

KQML消息的层次结构

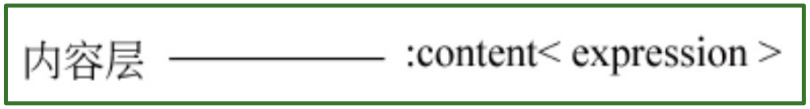
- 消息层用于描述：
  - Agent间交互的消息类型（语用词，performative）
  - 内容层描述所采用的内容描述语言
  - 内容层所描述的信息基于什么样的本体来解释其语义

KQML消息的层次结构（续）

- 通信层描述了消息通信相关的一组通信参数：
  - 消息的发送者
  - 消息的接收者
  - 消息的标识
    - 当前消息的标识：reply-with
    - 被应答消息的标识：in-reply-to



- 内容层描述了Agent之间交互的消息内容：
  - 消息内容可以用任何内容描述语言进行表述
  - 典型的内容描述语言有：XML、KIF等



KQML消息示例1

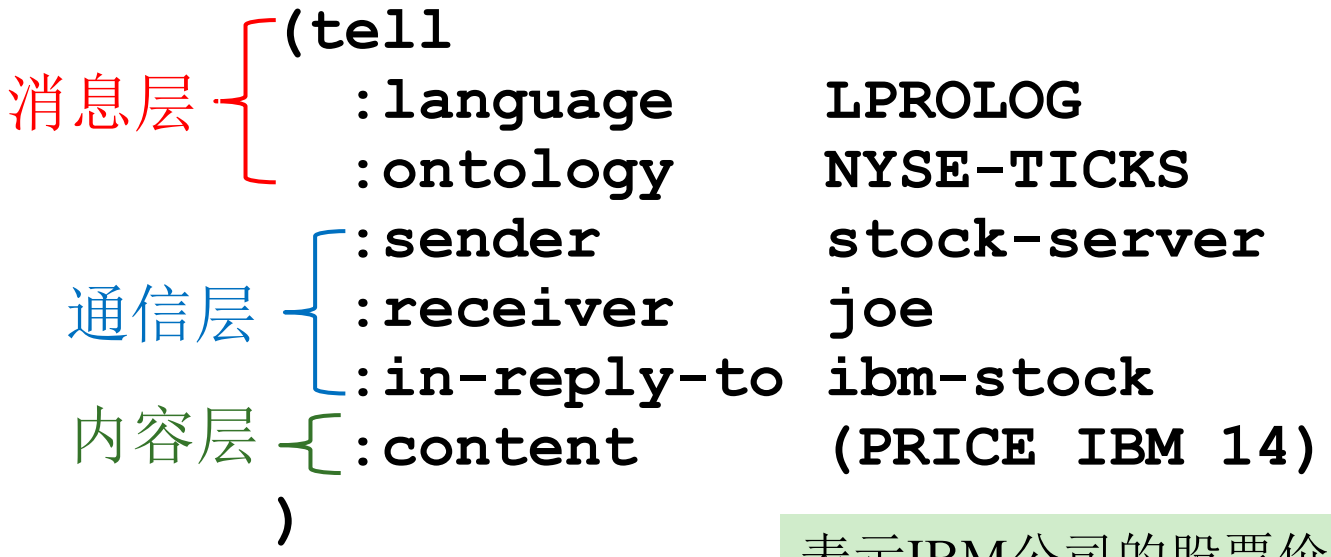
消息层
通信层
内容层

```
(ask-one
   :language     LPROLOG
   :ontology     NYSE-TICKS
   :sender       joe
   :receiver     stock-server
   :reply-with   ibm-stock
   :content      (PRICE IBM? price)
)
```

询问IBM公司的股票价格

发送者joe向接收者stock-server发送了一个ask-one类型的消息，该消息的标识为ibm-stock，内容是(PRICE IBM? price)，消息内容用LPROLOG语言来描述的，其本体论为NYSE-TICKS。

KQML消息示例2



表示IBM公司的股票价格是14美元

发送者stock-server向接收者joe发送了一个tell类型的消息，该消息没有显式的标识，它是对标识为ibm-stock消息的响应，内容是(PRICE IBM 14)，消息内容用LPROLOG语言来描述的，其本体论为NYSE-TICKS。

KQML消息示例3

```
(tell
   :language      KQML
   :ontology      kqml-ontology
   :sender        facilitator
   :receiver      agent3
   :in-reply-to   id0
   :reply-with    id1
   :content       (tell
                   :receiver    C
                   :language    Prolog
                   :ontology    foo
                   :content     "bar(X, Y)")
)
```

KQML消息可相互嵌套

发送者facilitator向接收者agent3发送了一个tell类型的消息，该消息的标识为id1，它是对标识为id0消息的响应，消息内容是一条KQML消息，其本体论为kqml-ontology。

KQML – 对话实例

Dialogue (a)

```
(evaluate
  :sender A :receiver B
  :language KIF :ontology motors
  :reply-with q1 :content (val (torque m1)))
```

A→B: 查询m1的力矩值

```
(reply
  :sender B :receiver A
  :language KIF :ontology motors
  :in-reply-to q1 :content (= (torque m1) (scalar 12 kgf.m)))
```

B→A：回复m1的力矩值大小

Dialogue (b)

```
(stream-about
  :sender A :receiver B
  :language KIF :ontology motors
  :reply-with q1 :content m1)
```

A→B: 查询所有关于m1的信息

```
(tell
  :sender B :receiver A
  :in-reply-to q1 :content (= (torque m1) (scalar 12 kgf.m)))
(tell
  :sender B :receiver A
  :in-reply-to q1 :content (= (status m1) normal))
```

B→A：发送信息

```
(eos
  :sender B :receiver A
  :in-reply-to q1)
```

B→A: 通信流结束

KQML的一些问题

- 基本的KQML语用词太容易改变
- KQML的消息传递机制没有严格定义
- KQML的语义没有严格定义
- KQML忽略了承诺语用词
- KQML的语用词集合过于庞大、过于具体

KQML的这些问题导致智能物理Agent基金会（FIPA）组织开发了一种新的，但是有密切关系的语言：

ACL（Agent Communication Language）

通信

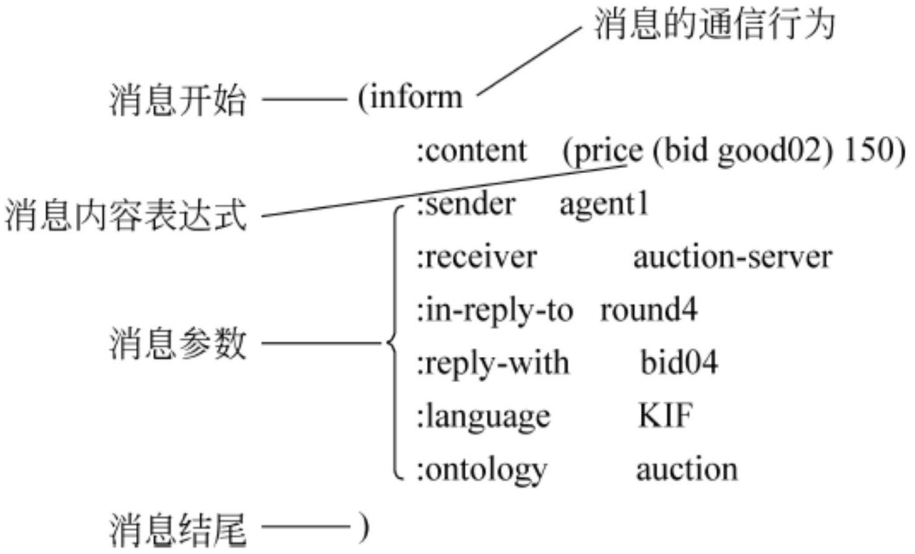
- 言语行为
- 通信方式
- 基于消息的Agent 通信语言：KQML
- **基于消息的Agent 通信语言：FIPA ACL**

FIPA ACL - Agent通信语言

- FIPA提出的Agent通信语言ACL
  - 定义了Agent之间交互的一组消息类型，对消息的语法、语义和语用做出了严格、形式化的描述和定义

- ACL消息的结构和表示方式

FIPA ACL与KQML在消息结构和消息的属性域上都十分相似



FIPA ACL 与 KQML

- FIPA ACL与KQML的最重要区别在于它们提供的语用词

FIPA ACL把语用词的数量减少到了20个

通信行为	直观含义	类型				
		信息传递	信息请求	协商	动作执行	错误处理
accept-proposal	接受以前提交的建议以执行一个动作			√		
agree	同意执行某一动作				√	
cancel	取消以前请求执行的动作				√	
cfp	请求一个建议以执行某个动作			√		
confirm	告诉接收者某个命题成立	√				
disconfirm	告诉接收者某个命题不成立	√				
failure	告诉接收者试图执行某个动作，但是执行失败					√
inform	通知接收者某个命题成立	√				
inform-if(macro act)	通知接收者某个命题是否为真	√				
inform-ref(macro act)	通知接收者某个描述例子所对应的对象	√				
not-understood	不能理解消息					√
propose	提交一个建议以执行某个动作			√		
query-if	询问某个命题是否成立		√			
query-ref	询问某个表达式所指的对象		√			
refuse	拒绝执行一个动作				√	
reject-proposal	在协商中拒绝接受一个动作执行的建议			√		
request	请求执行一个动作				√	
request-when	请求当某个命题成立时执行某个动作				√	
request-whenever	请求每当某个命题成立时就执行某个动作				√	
subscribe	请求一个引用的值		√			

FIPA ACL消息示例1

```
(request
   :sender      i
   :receiver    j
   :content     (action j (deliver box017 (location 12 19))
   :protocol    fipa-request
   :reply-with order567
)
```

发送者i向接收者j发送了一条request类型的消息，该消息的标识为order567，请求Agent j执行动作以将box017送到位置(12 19)处；该交互采用的是fipa-request协议。

FIPA ACL消息示例2

```
(agree
   :sender        j
   :receiver      i
   :content       ((deliver j box017 (location 12 19)))
   :in-reply-to   order567
   :protocol      fipa-request
)
```

发送者j向接收者i发送了一个agree类型的消息，该消息对标识为order567的消息做出响应，Agent j同意执行动作以将物体box017递送到位置(12 19)处；该交互采用的是fipa-request协议。

FIPA ACL消息示例3

```
(failure
   :sender       j
   :receiver     i
   :content
   (
            (action j "open(\"\foo.txt\")"
            (error-message "No such file: foo.txt")
   )
   :language     s1
)
```

发送者j向接收者i发送了一个failure类型的消息，告诉它打开文件foo.txt动作失败，失败的原因是没有相应的文件；内容描述语言为s1。

FIPA ACL消息示例4

```
(inform
   :sender        i
   :receiver      j
   :content       "weather(today.raining)"
   :language      Prolog
)
```

发送者i向接收者j发送了一个inform类型的消息，告诉Agent j今天的天气是有雨；内容描述语言为Prolog。

FIPA ACL的语义

- 对KQML最多的批评之一是它缺少满足要求的语义
- FIPA ACL中采用Cohen和Levesque提出的“作为理性动作的言语行为理论”，通过一个名为SL的形式语言表示Agent的信念、愿望与意图
- FIPA ACL的语义定义包含两方面的内容：
  - 通信行为实施的可行前件（Feasibility Precondition, FP）
  - 通信行为实施后的理性结果（Rational Effect, RE）
- FIPA ACL采用以下方式来定义通信行为的语义：
  - 用自然语言描述通信行为的直觉含义
  - 用SL语言描述通信行为的FP和RE部分

通信行为inform的语义

- inform通信行为$< i, \text{inform}(j, \phi) >$的语义定义描述：
  - 直觉含义：Agent i通知Agent j， ϕ为真
  - 可行前件： $B_i\phi \wedge \neg B_i\left(Bif_j\phi \vee Uif_j\phi\right)$
    - Agent i相信ϕ成立并且它不清楚Agent j是否知道关于ϕ的信息
    - $B_i\phi$表示i相信ϕ
    - $Bif_j\phi$表示j能够判断ϕ的真假
    - $Uif_j\phi$表示j不确定ϕ是什么
  - 理性结果： $B_j\phi$
    - 该消息成功实施后的理性结果是Agent j相信ϕ成立

通信行为request的语义

- request通信行为$< i, \text{request}(j, \alpha) >$的语义定义描述：
  - 直觉含义：Agent i请求Agent j执行动作α
  - 可行前件：$B_i\text{Agent}(\alpha, j) \wedge \neg B_i I_j \text{Done}(\alpha)$
    - Agent i相信执行动作α的Agent是j，而且Agent i相信Agent j当前没有打算执行α
    - $\text{Agent}(\alpha, j)$表示做动作α的Agent是j
    - $\text{Done}(\alpha)$表示已经执行了动作α
  - 理性结果：$\text{Done}(\alpha)$
    - 该消息成功实施后的理性结果是已经执行了动作α

小结

- 言语行为
  - 传达某种交互的意图和内容
  - 影响其他Agent的内部状态以及相应的行为实施
- 言语行为理论
  - 把通信建模成可以改变通信参与者的思维状态的动作
  - 三分说、主要类型、基于规划的言语行为理论等
- 通信方式
  - 消息、黑板、邮箱
- Agent通信语言
  - KQML、FIPA ACL

内容安排

合作

- Agent之间**为什么**要而且**如何**进行合作？
- “合作”一词时常使用在并发系统的文献中
- 与传统的分布式系统相比，在多Agent系统中的“合作”存在如下特点：
  - 多Agent系统中的Agent可能由不同的个体进行设计和实现，具有不同的目标
  - 由于Agent是行为自治的，他们必须在运行时自己进行决策，而且具有动态协调动作的能力，以及与其他Agent合作的能力

合作

- **合作分布式问题求解**
- 任务共享和结果共享
- 不一致性
- 协调

合作分布式问题求解（Cooperative Distributed Problem Solving, CDPS）

- CDPS研究问题求解器的松耦合网络如何能合作求解问题，这些问题超出了个体能力的范围。网络中每个问题求解器具有复杂问题求解的能力并可以独立工作，但是在没有合作时，这些求解器不可能完成所面临的问题。合作是必需的，因为没有一个求解器具有问题求解所需的足够专长、资源和信息，并且不同的求解器具有求解问题不同部分的专长。

(Durfee, Lesser, and Corkill, 1989)

仁慈的（Benevolent）Agent

- 如果系统中所有的Agent是由同一个组织或个体拥有时，可以设计这些Agent随时可以互帮互助
- 在这种情况下，可以假设Agent是仁慈的，它们共享一个共同的目标且不存在潜在的冲突性
- 仁慈系统中的问题求解即是CDPS
- 仁慈假设能极大地简化设计者的任务

自利的（Self-interested）Agent

- 通常情况下，Agent代表着不同的个体或组织，我们就不能做出仁慈性假设
- Agent会根据自己的利益采取行动，而且有可能导致别的Agent的利益损失
- 潜在冲突性
- 这种情况有可能极大地复杂化任务设计
- 尽管Agent之间存在潜在的利益冲突，但最终会像人类社会一样，为了实现各自的目标进行合作

CDPS与并行问题求解的区别

- 并行问题求解简单地包括问题求解中并行机制的研究
- 在并行问题求解中，计算部件是简单的处理器，通常假设这些处理器是同构的，没有不同的专长

一致性（Coherence）

- 这是用来评价一个基于Agent的系统的标准
- 一致性

> 根据某些衡量标准，多Agent系统构成的一个整体的表现有多好
>
> (Bond and Gasser, 1988)

- 我们可以依据解的质量、资源利用效率、操作的概念清晰度等方面进行一致性的度量

协调性（Coordination）

- 协调性

> Agent能避免“外来”活动（如：同步和调整等活动）影响的程度
>
> (Bond and Gasser, 1988)

- 在一个完全协调的系统中，在努力实现共同目标的过程中，Agent不会阻碍到别的Agent的行动；它们也不必进行明显的通信，因为可以通过维护各自的内部模型来进行相互预测。

CDPS中的主要问题

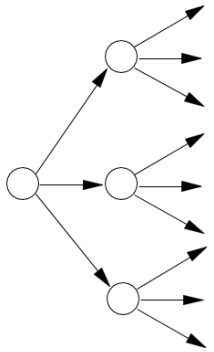
- 如何将问题分解为能在Agent之间分配的子任务？
- 如何有效地将子问题的解综合为问题的解？
- 如何进行Agent问题求解活动的整体优化，以产生一个最大程度上符合某些一致性度量的解？
- 使用什么样的技术来协调Agent的活动，以避免发生破坏性（且无用）的相互影响，并优化系统效率（通过利用任何积极的相互影响）？

合作

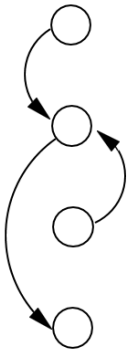
- 合作分布式问题求解
- **任务共享和结果共享**
- 不一致性
- 协调

CDPS的三个阶段

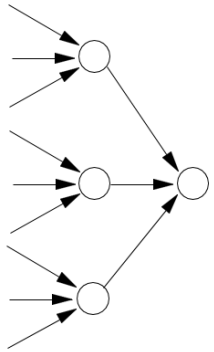
- 一组Agent如何合作求解问题？
- 将CDPS过程分成以下三个阶段：
  - 问题分解
  - 子问题求解
  - 解综合



问题分解



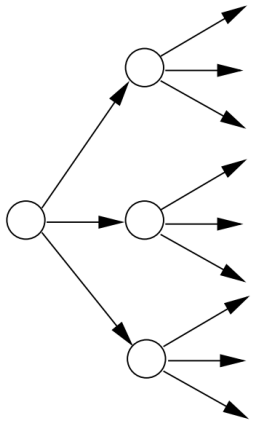
子问题求解



解综合

接下来具体介绍这三个阶段

问题分解



- 这个阶段将需要求解的问题分解为更小的子问题

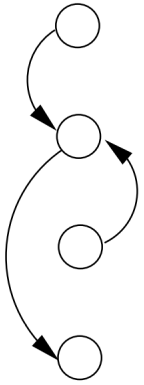
- 这是一个递归的、分层的分解：
  - 子问题可以被进一步分解为更小的问题
  - 子问题的粒度大小是重要的
  - 在ACTOR规范中，为每个子问题再派生一个新的Agent，问题被分解为像加、减等这样单个的程序指令为止

问题分解（续）

- 如何完成分解？
  - 一种可能是由一个Agent完成问题的分解
  - 前提：该Agent具有合适的专长来做这个分解
  - 分解过程本身也可以作为一个合作活动来更好地处理

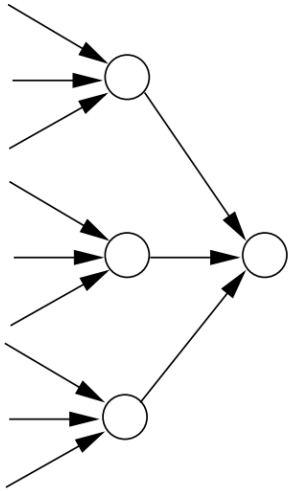
- 谁来完成任务分解？
  - 中心化执行？
  - 哪些Agent有关于任务结构方面的知识？
  - 哪些Agent来求子问题的解？

子问题求解



- 这阶段，对问题分解阶段已识别的子问题进行单个求解
- 该阶段主要包括Agent之间信息的共享
  - 如果一个Agent具有对其他Agent有用的信息，那么Agent可以帮助其他Agent
  - 可能需要动作同步，如合作推一个箱子

解综合



- 这个阶段，单个子问题的解综合成为整体的解

- 像问题分解一样，这个阶段可以是分层的
  - 子问题的解在不同的抽象级别上集成

任务共享和结果共享

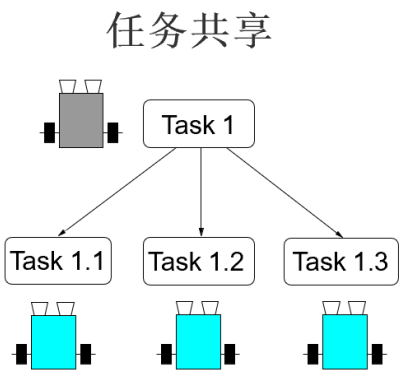
- 在三个阶段的CDPS框架中，有两个活动可能在合作中出现：
  - 任务共享：

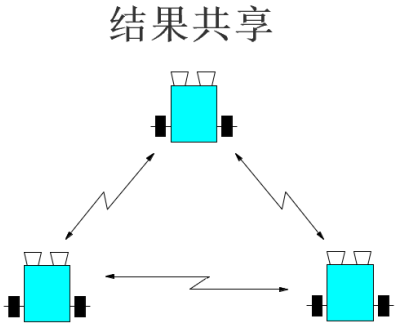
    一个任务的各个部分被分配给不同的Agent

    如何决定如何分配？
  - 结果共享：

    信息（如部分结果）被分配

    我们如何从部分合成一个完整解？

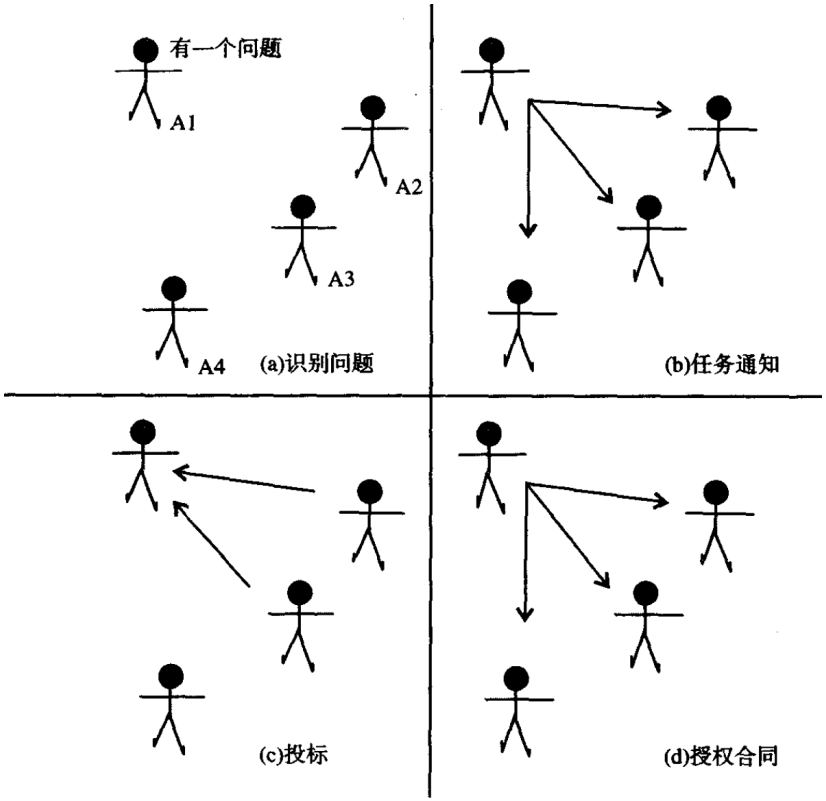




合同网

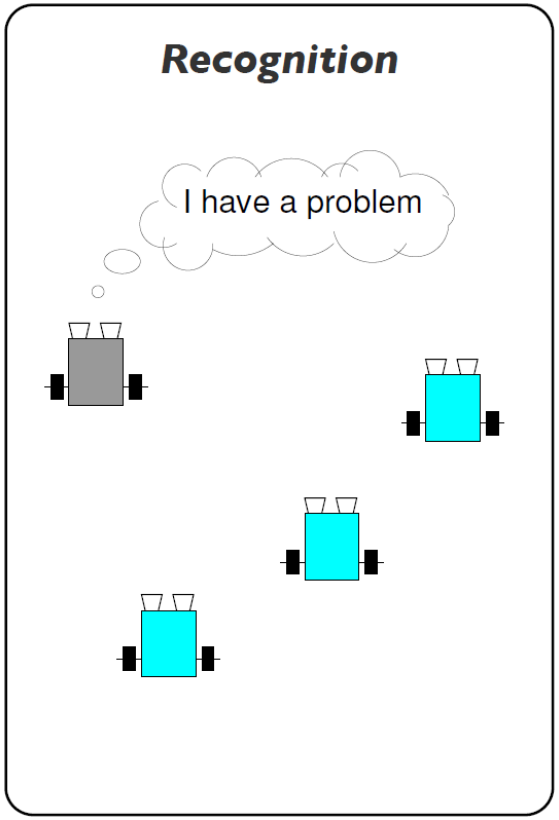
- 合同网（Contract Net, CNET）协议是通过任务共享实现有效合作的高级协议

- 合同网包含五个阶段：
  - 识别
  - 通知
  - 竞标
  - 授予
  - 实现



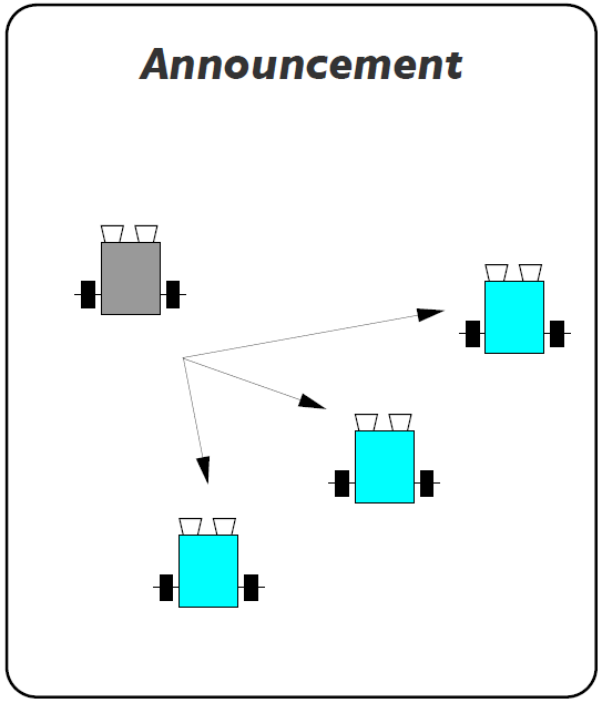
识别

- 在这个阶段，Agent识别出需要别的Agent帮助的任务
- Agent有一个目标
  - 意识到它自己不能单独完成目标—没有这个能力;
  - 意识到它更希望有别的Agent来帮忙达到目标
    - 如：解的质量、截止日期等因素
- 于是，这个Agent需要别的Agent参与进来



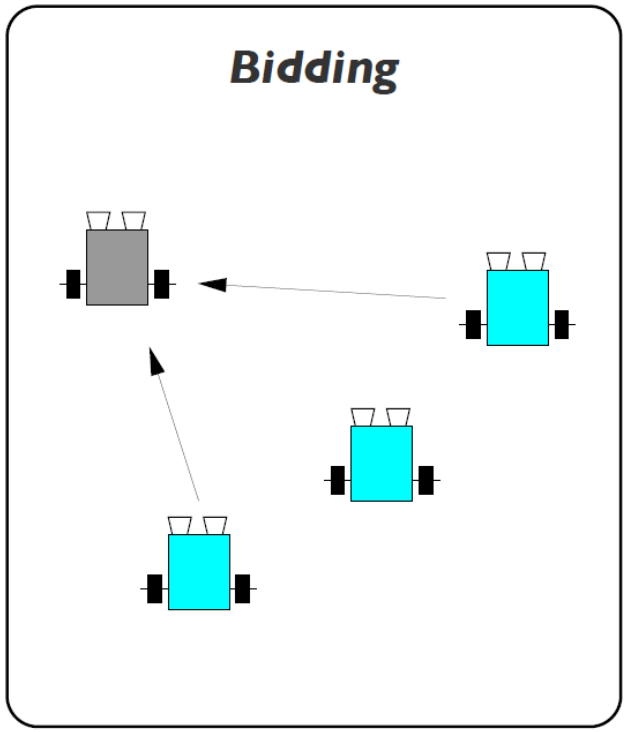
通知

- 在这个阶段，有这个任务的Agent发出一个通知，这个通知里包含了这个任务的明细单
- 明细单必须记录：
  - 对任务本身的描述
  - 约束条件
    - 如截止日期、解的质量要求
  - 元任务信息
    - 如“投标必须由……来提交”
- 然后将这个通知广播出去

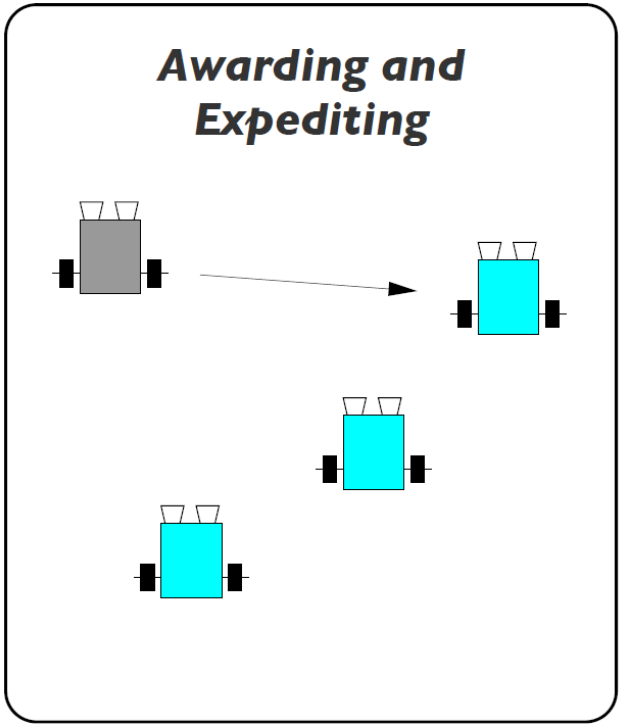


竞标

- 接收到通知的Agent决定是否想要竞标
- 考虑因素：
  - 自己是否有能力完成任务
  - 质量约束和价格等信息
- 如果选择竞标，那么Agent就投标



任务授予和实现



- 发出通知的Agent必须在投标的Agent中间选出“授予合同”的对象
  - 选择的结果需要通知给投标的Agent
  - 竞标成功的Agent，即任务承包商，负责完成相应的任务
- 实施过程中可能生成进一步的“经理-承包商”关系
  - 子合同：可能包含另一个合同网

实现合同网时会碰到的问题

- 如何
  - 指明任务？
  - 指明服务质量？
  - 决定是否竞标？
  - 在竞争报价中挑选？
  - 在多重标准的基础上将不同的报价区分开？

- 合同网的代码实现
  - https://github.com/Sina-Baharlou/Contract-Net-Protocol

决定如何竞标

- 在时刻t一个Agent i计划执行一系列任务 τ_i^t
  - Agent i拥有资源 e_i，需要使用代价 $c_i^t(\tau)$ 来执行这些任务
- Agent i收到了任务$\tau(ts)$的通知
  - 执行 $\tau(ts)$ 的边际成本：

$$\mu_i\left(\tau(ts)\middle|\tau_i^t\right) = c_i\left(\tau(ts) \cup \tau_i^t\right) - c_i(\tau_i^t)$$

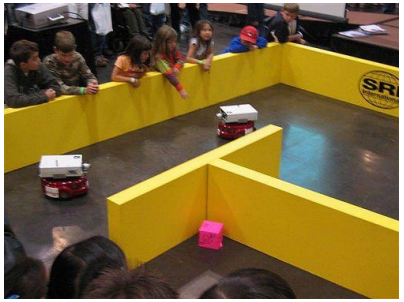
  - 由于协同作用，它可能为0——额外的任务可能可以免费完成，如顺风车额外带一个人去上班的代价
  - 只要 $\mu_i(\tau(ts)|\tau_i^t) < e_i + e(ts)$，Agent i去竞标是合理的
    - $e(ts)$表示执行任务$\tau(ts)$能得到的额外资源

结果共享

- 在结果共享中，Agent之间在共同合作求解时，互相给予信息
- 结果共享在以下几个方面有利于问题求解：
  - 信心（Confidence）：独立来源的解可以进行交叉检查，揭示可能存在的错误和增加对整体解的信心
  - 完整性（Completeness）：通过局部解视图的共享，得到一个更好的、整体的解视图
  - 精确度（Precision）：共享的结果可以提升解的精确度
  - 及时性（Timeliness）：共享结果使得在某些情况下可以动用并行资源求解

结果共享的一些例子

- 黑板系统
  - 通过共享的数据结构（黑板）实现结果共享
  - 多个Agent可以读取黑板上的数据
  - 多个Agent可以向黑板写数据（一部分解）



- 订阅/通知模式
  - 一个对象订阅另一个对象
    - 当事件e发生的时候，告诉我
  - 信息以预动的方式在对象间共享

机器人合作来绘制地图和寻找对象

合作

- 合作分布式问题求解
- 任务共享和结果共享
- **不一致性**
- 协调

出现不一致性的原因

- 一组Agent可能在以下两个方面产生不一致性：
  - 信念
  - 目标/意图
- 产生不一致信念的原因
  - Agent不能看到全局信息
  - Agent拥有的传感器有毛病，或者Agent访问的信息源也可能有毛病
- 产生不一致目标的原因
  - Agent是自治的，没有共享的目标

处理不一致性的方法

三种可能的方法：

- 不允许出现
  - 例：在合同网中，任务共享总是由管理者Agent来驱动
- 通过协商解决不一致性
  - Agent之间通过协商，讨论不一致的信息和目标直至不一致性消除
- 建立一个不一致性出现后能平稳退化的系统
  - 三种方法中最理想的

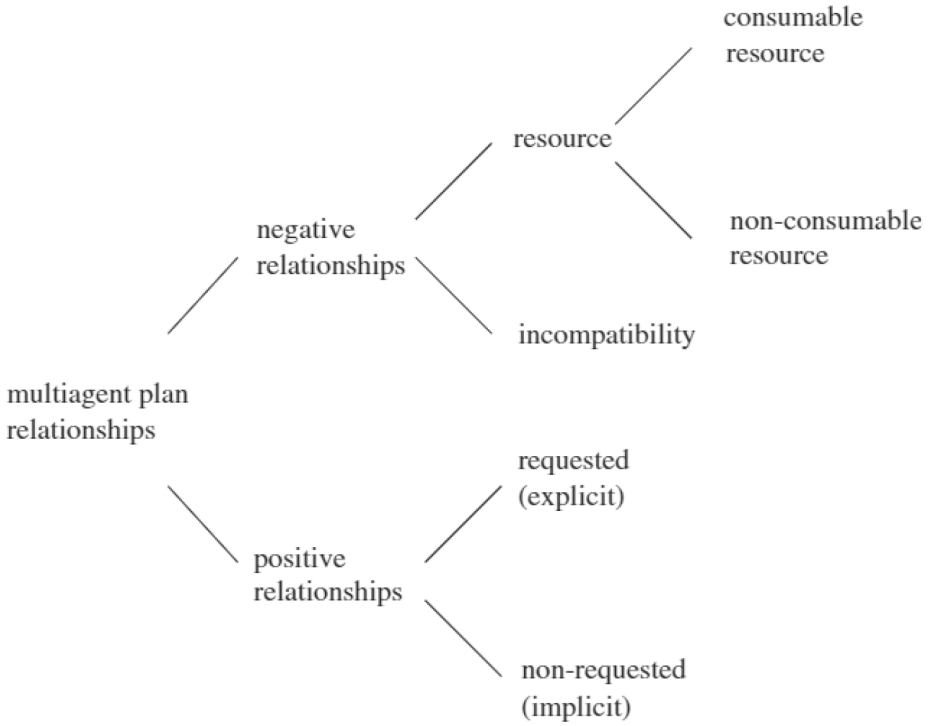
合作

- 合作分布式问题求解
- 任务共享和结果共享
- 不一致性
- **协调**

协调

- 协调问题：如何管理Agent活动（activity）之间的内部依赖关系？
- 如果Agent参加的活动中存在任何形式的相互作用，那么一定的协调机制是必需的
- 例：
  - 你和我都打算离开房间，并且我们各自同时向门走去，而门只允许我们一个人走过去，如何保证我们俩都可以走过门？
  - 我们同时来到打印室要复印一叠材料，谁先使用复印机？

协调关系的分类



- 活动之间的依赖关系分为积极的、消极的
- 积极的关系
  - 使用它能使至少一个Agent获益
  - 请求的
    - 我显式地请求你对我的活动给予帮助
  - 非请求的
    - 由于这样合作的出现，至少一个Agent可以获得更好的解，同时也不会对其他Agent产生坏的影响

非请求的协调关系

三种非请求的协调关系：

- 行动平等（Action Equality）关系：
  - 我们都打算去执行一个相同的动作，并且相互认识到这一点，这样我们中的一个能独立去执行动作，节省另一个人的努力
- 后承（Consequence）关系：
  - 我规划中的动作的执行可以致使你的某个目标的完成，这是我的动作的附加作用，消除了你去完成该目标的需要
- 恩惠（Favour）关系：
  - 我规划中的部分有这样的附加作用，其执行会对你的一个目标的实现产生贡献，使得该目标的实现更容易

动态协调活动的方法

- 通过部分全局规划的协调
- 通过联合意图的协调
- 通过互相建模的协调
- 通过规范和社会法律的协调
- 通过多Agent规划的协调

通过部分全局规划的协调

- 主要原理：在问题求解过程中，合作的Agent需要交换信息来达成一致
- 部分全局规划：将一组Agent的动作和相互作用结合在一起形成的数据结构
- 该结构是通过Agent之间交换信息而合作生成的。它包含如下基本的属性：
  - 目的
  - 活动图
  - 解的结构图

通过部分全局规划的协调（续）

- 规划是部分的
  - 系统不能产生整个问题的规划
- 规划是全局的
  - Agent可以通过局部规划的交换和合作，得到问题求解的全局视图，进而形成全局规划
- 该协调方法包含3个迭代的阶段：
  - 每个Agent决定自己的目标，并且为实现目标产生短期的规划
  - Agent之间通过信息交换，获知其他Agent的规划和目标
  - 为了更好协调它们各自的动作，Agent要修改局部的规划

通过部分全局规划的协调（续）

- 广义的部分全局规划：利用了如下的5种协调活动技术，扩展和完善了基于部分全局规划的协调机制：
  - 非局部观点的更新
  - 结果的沟通
  - 简单冗余的处理
  - 硬协调关系的处理：“消极的”
  - 软协调关系的处理：“积极的”

通过联合意图的协调

- 在协调中，意图扮演了重要的角色
  - 提供了社会交互必需的稳定性和预见性
  - 及应对变化环境的灵活性和反应性

例子：如果你知道我正打算写一本书，那么这为你协调和我有关的活动提供了信息

如：取消和我一起的度假、聚会，因为你知道我正忙于写书的工作

- 就像每个人有个人的意图，一个团队有联合意图

通过联合意图的协调（续）

- 联合持续目标：当一组Agent从事合作活动时，它们必须有一个对整体目标的联合承诺，以及分配给自己的具体任务的个体承诺
  - 联合承诺共享个体承诺的持续特性
- 一组Agent有一个实现某目标φ的联合承诺
  - 例：φ可能是“移动这个沙发”
- 也有目标的动机ψ
  - 例：ψ可能是“西蒙想要移动这个沙发”

通过联合意图的协调（续）

- 具有联合持续目标的Agent队的思维状态：
  - 开始：每个Agent不相信φ已经达成，但相信是可能的
  - 每个Agent i有目标φ直到结束条件满足

- 结束条件是以下的任一个条件被互相相信：
  - 目标φ已经满足
  - 目标φ是不可能的
  - 目标的动机/理由ψ不再存在

通过联合意图的协调

通过通信达成

- 如果结束条件没满足，那么：
  - 如果任何Agent i相信目标被实现，那么它将有一个让该信息成为互相信念的目标，并且保持该目标直到结束条件满足
  - 如果任何Agent i相信目标是不可能的，那么它将有一个让该信息成为互相信念的目标，并且保持该目标直到结束条件满足
  - 如果任何Agent i相信目标的动机ψ不再存在，那么它将有一个让该信息成为互相信念的目标，并且保持该目标直到结束条件满足

这个机制保证了Agent之间的活动是协调的

基于队工作的CDPS模型

- 基于队工作的协调模型，模型分4个阶段：
  - 识别
    - Agent认识到合作动作对其达成目标的潜力
  - 队形成
    - 这个阶段的结果是，队伍中Agent对实现的目标达成一致，但还未就实现方法达成一致
  - 规划形成
    - 在这个阶段，Agent就应该执行什么活动的过程达成一致
  - 队活动
    - 在这个阶段，Agent执行新的达成一致的联合活动的规划，并维持编织完全紧密的关系

通过协议来定义这种关系，前面介绍的联合持续目标是一个可能的协议

通过互相建模的协调

- 回顾一下先前给出的简单协调的例子：

  你和我一起向门走去，并且门容不下我们两个一起通过

  出现冲突时，我们应该做什么？

- 一种做法：都停下来，这样可以保证不会产生冲突，但门作为一个资源就没有被利用，浪费了
- 另一种做法：我们将自己置身于对方的立场上
  - 你可能相信我热心于取悦你，因此我可能会让你先通过门，然后基于这一点，你继续向门走去

MACE系统的熟人模型

- 熟人模型是对其他Agent的表示
  - 其他Agent的技能、兴趣、能力、喜好等

- Agent对熟人的如下信息进行维护：
  - 类：用来组织Agent，并通过类名识别类
  - 名字：每个Agent分配一个名字
  - 角色：Agent在类中扮演的成分
  - 技能：Agent知道的、被建模的Agent的能力
  - 目标：Agent知道的、被建模的Agent打算实现的目标
  - 规划：该Agent观点下，被建模的Agent实现目标的方法

相互建模的前沿进展

- 推荐读物

Stefano V. Albrecht, Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence, 258: 66-95, 2018

Stefano V. Albrecht

Peter Stone

通过规范和社会法律的协调

- 社会通常是由规范和社会法律所调控的
  - 规范：建立的、期望的行为模式
  - 社会法律：也表达相同的含义，但通常带有强迫性

- 例：

  在英国，当等候公交车时排队，并让先来的人先上公交车，这是规范

- 规范提供了一种模式来控制大家的行为
  - 违背规范通常会引起公交车上其他人的冷眼

协议及其制订方法

- 在社会过程中，协议扮演关键的角色
  - 协议为Agent提供了模式，用它Agent可以组织行为过程
  - 协议表示一种行为的约束
  - 协议简化了Agent做出决策的过程
  - 几乎社会的每个方面都自然依赖于协议
- 在Agent社会中，制订协议的方法
  - 离线设计
  - 系统内产生

离线设计

- 假设Agent模型是一个把运行映射到动作的函数：

$$Ag : \mathcal{R}^E \rightarrow Ac$$

- 那么约束是一个二元组：

$$(E', \alpha)$$

其中 $E' \subseteq E$ 是环境状态的一个集合，$\alpha \in Ac$是一个动作

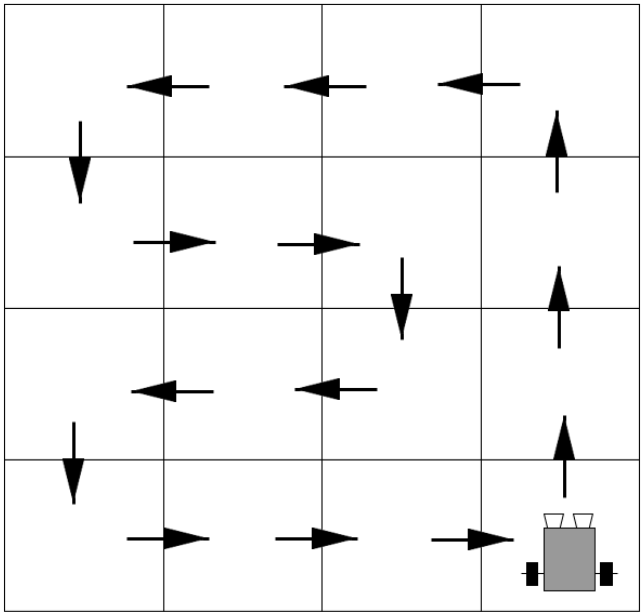
- 这个约束是说，如果Agent处在某种 E' 中的某个状态，那么动作α是被禁止的
- 社会法律就是这些约束的集合sl
  - 如果Agent不试图去执行sl中某约束所禁止的动作，则可以说Agent是遵守社会法律sl的

如何定义有用的社会法律？

- 定义一个焦点状态集合 $F \subseteq E$
  - 这些状态是合法的，Agent总是能访问它们
- 一个有用的社会法律不会阻止Agent从一个焦点状态转移到另一个焦点状态
  - 只要当环境位于某焦点状态$e \in F$时，Agent能保证通过可能的动作到达任何其他状态 $e' \in F$
- 有用的社会法律问题（NP-完全问题）
  - 给定一个环境$Env =< E, \tau, e_0 >$和焦点状态集合$F \subseteq E$，如果存在就寻求有用的社会法律，否则通告不存在这样的法律

社会法律的例子

- 在以下多个Agent在网格中行走的场景中，一种可能的避免冲突的社会法律为：



  - 在偶数行时向左移动，而在奇数行时向右移动
  - 当位于最右边的列时，向上移动
  - 当位于偶数行的最左边列或奇数行的右边倒数第二列时，向下移动

- 不是非常有效率
  - 到达某个指定的方格的时间复杂度为$O(n^2)$

扩展阅读

- 效率更高的社会法律：时间复杂度为$O(n)$

Yoav Shoham, Moshe Tennenholtz. On social laws for artificial agent societies: Off-line design. Artificial Intelligence, 73: 231-252, 1995

Yoav Shoham

Moshe Tennenholtz

产生的规范和社会法律

- 我们也可以设计社会法律在其中自行产生的系统
- T恤衫游戏：

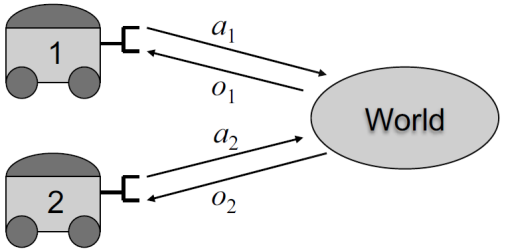
> 每个Agent有两件T恤衫：一件红色，一件蓝色。游戏的目标是所有的Agent最后穿上相同颜色的T恤衫。每一轮，每个Agent和其他的一个Agent随机组成一队，每队Agent查看对方的颜色并决定是否更换衣服。这一轮中Agent只能看到对方的T恤颜色，没有其他信息。

- Agent只能使用所存储的前几轮的经验来做出决策
  - 是换一件T恤衫还是继续穿现在的？

- 策略更新函数
  - 表示从Agent至今所观察的历史信息到颜色的映射
  - 通过使用该函数，能使Agent社会的每个Agent尽可能有效地达成全局的一致
- 不同的策略更新函数
  - 简单从众
    - 总是选择出现次数最多的那种颜色的T恤衫
  - 根据Agent类型的简单从众
    - 将Agent分为两类，如果Agent和自己同类型的Agent相遇，则通过通信来交换各自的记忆历史，否则简单从众
  - 最高的累积奖赏
    - Agent根据所看到的其他Agent（所有Agent的某一子集）成功配色的频率，选择最多配对的颜色

通过多Agent规划的协调

- 多Agent规划
  - 必须考虑Agent的活动之间存在相互作用的事实，直接规划所有Agent要做些什么
  - 集中式规划、分布式规划
- 分布式规划
  - 输入问题的模型表示，通过算法输出Agent所能执行的动作序列
  - 规划过程不一定是分布式的，但要求Agent能够分布式地执行规划的结果
  - Agent之间是相互独立的，它们获得的信息也是相互独立的
    - 每个Agent只能观察到世界状态的某个局部



由两个Agent构成的分布式规划问题

小结

- 合作分布式问题求解（CDPS）
  - 仁慈性、自利性、一致性、协调性
  - 与并行问题求解的区别，研究的主要问题
- 任务共享和结果共享
  - CDPS的三个阶段：问题分解、子问题求解、解综合
  - 任务共享：合同网
  - 结果共享：黑板系统、订阅/通知模式
- 不一致性
- 协调
  - 部分全局规划、联合意图、互相建模、规范和社会法律等

课后作业3-2

- 利用课后作业3-1中设计的本体，使用KQML语言描述一段对话：
  - A向B查询（evaluate）《多智能体系统》课程的选课人数情况。
  - B告知A《多智能体系统》选课人数为30人。
- 需要描述清楚每条消息的语用词和参数，你可以使用自然语言表达消息内容。

课后作业3-3

- 利用课后作业3-1中设计的本体，使用KQML语言描述一段对话：
  - A向B查询《机器学习导论》课程的所有信息。
  - B告知A《机器学习导论》的课程编号为30000150。
  - B告知A《机器学习导论》的授课老师为周志华老师。
  - B告知A《机器学习导论》选课人数为300人。
  - B告知A当前通信流结束。

- 需要描述清楚每条消息的语用词和参数，你可以使用自然语言表达消息内容。

课后作业3-4

- 使用自然语言，描述下面两段KQML消息的含义

(a)
```
(ask-if
     :sender      A
     :receiver    B
     :language    OWL
     :ontology    pizza
     :reply-with  q1
     :content  ( (margherita isa Pizza)
                 (margherita hasTopping mozzarella) )
  )
```

(b)
```
(tell
     :sender      A
     :receiver    B
     :language    OWL
     :ontology    pizza
     :reply-with  q1
     :content (not (hawaiian isa ItalianPizza))
  )
```

课后作业3-5

- 使用KQML语言，描述如何实现合同网协议。

交第一次课后作业（第1~3部分）的截止时间为：

2023年4月18日