# CPT302 W5

# Reactive and Hybrid Agents

## The Subsumption (Reactive) Architecture

Symbolic/logical approaches 中的问题 (transduction, computational complexity)，导致了 reactive paradigm (反应范式) 的出现。

The best-known reactive agent architecture (又名 subsumption architecture) based on the following theses:

- 不需要 explicit representations 即可生成智能行为 (Intelligent behavior)
- 不需要 explicit abstract reasoning 即可生成智能行为
- Intelligence 是某些复杂系统的新兴属性 (emergent property)

#### Characteristics of the Subsumption Architecture

代理的决策是通过一组 task-accomplishing behaviors (完成任务的行为) 来实现的:

- 每个行为都可以看作是一个单独的 selection function，它不断地将 perceptual input (感知输入) 映射到要执行的操作
- 行为 (behavior modules) 作为 finite-state machines/rules 实现，不包含复杂的符号表示 (situation -> action rules)

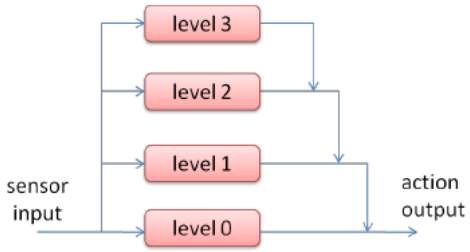
许多行为可以同时激活:

- modules 排列为一个 subsumption hierarchy，behaviors 在其中排列为 layer
- 层次结构中的较低层可以 inhibit (抑制) 较高层（层越低，其优先级越高）

### Action Selection in Layered Architectures

传感器的原始输入没有经过太多的处理或转换。

Action selection 是通过一组行为以及抑制关系来实现的。我们将 $'b1\ inhibits\ b2'$ 表示为 $b1 \prec b2$，其中 b1 在层次结构中低于 b2，因此 b1 的优先级高于 b2。




## Example - Mars Explorer

Mars Explorer 的目标是探索一个遥远的星球，特别是收集珍贵岩石的样本。样品的位置事先尚不清楚，但已知它们往往是聚集的。有许多自动驾驶汽车可以在星球上行驶，收集样本，然后重新进入母舰航天器返回地球。没有详细的行星地图，尽管已知地形充满了障碍物，这些障碍物阻止了车辆交换任何通信。

A gradient field (梯度领域) - 母舰产生无线电信号，以便代理可以知道母舰位于哪个方向。An agent needs to travel 'up the gradient' of signal strength (一个代理需要沿着信号强度的 "梯度" 移动，即向母舰移动)。

### Individual Behaviors

Reactive agent 中是一系列的 behaviors，而不是一系列的 deduction rules (symbolic agent 中的推演规则)。

- $b_0$: **if** detect an obstacle **then** change direction
- $b_1$: **if** carrying a sample **and** at the base **then** drop sample
- $b_2$: **if** carrying a sample **and not** at the base **then** travel up gradient
- $b_3$: **if** detect a sample **and not** at the base **then** pick up sample
- $b_4$: **if** true **then** move randomly

The above behaviors are arranged into the hierarchy:
$b_0 \prec b_1 \prec b_2 \prec b_3 \prec b_4$

注：sample 即为代理要采集的样本；at the base 意为代理当前是否处于 base station (基站，即回到母舰)；travel up 意为代理向母舰移动。

在代理发现 sample 之前，它们执行 b0 和 b4 (寻找 sample)；而在发现 sample 之后，它们执行 b0，b2 和 b3 (把 sample 带回基站)；当到达基站后，代理执行 b1 (把 sample 放在基站)。

行为之间的相互抑制也可以看出来，比如前面有障碍 (避障) 和随机行动的条件同时被满足，避障是第一位的。同时，代理可以同时进行多个操作 (放下 sample 的同时随机行动)。

### Limitations of Reactive Agents

- 如果代理未使用当前的环境模型，则它们必须在 local environment 中得到可靠的信息，以确定可使用的操作
- 很难看出决策如何能够考虑到 non-local 信息 (a 'shot-term' view)
- 单个行为、环境和整体行为之间的关系是无法理解的。为特定任务设计代理是困难的，而且没有建立这种代理的方法
- 由于不同行为之间交互的动态性和复杂性，构建包含许多层**(>10)** 的代理是很困难的 (主要问题)

# Hybrid Agents

有人声称，完全使用 deliberative (pro-active) approach 或完全使用 reactive approach 都不适合构建代理。

一种显而易见的方法是从两个（或多个）子系统构建代理：

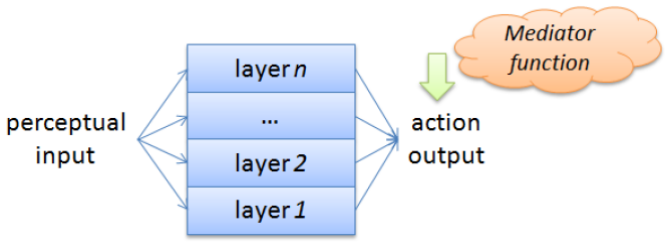
- a **deliberative** one，包含一个 symbolic world model，该模型以 symbolic AI 提出的方式制定 plan 和做出决策
- a **reactive** one，它能够在没有复杂推理的情况下对事件做出反应

Subsystems are arranged into hierarchy of interacting layers (由 layer 构成的层次结构，layered architecture)。

### Types of Layered Architectures

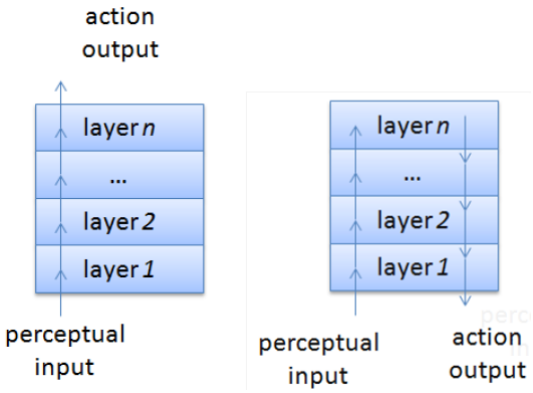
- Horizontal layering

  每个层都直接将感官输入和动作输出相连接。实际上，每个层本身都像一个代理，就要执行的操作提出建议 (每层的输入都是一样的，然后将得到的所有输出进行综合考虑 - mediator function，调解函数)




- Vertical layering

  感官输入和动作输出最多各由一层处理 (下图的左边的 one pass，其中 layer n 的输入是 layer n-1 的输出；右边是 two pass，可能得到更好的结果，但会花费更多时间)

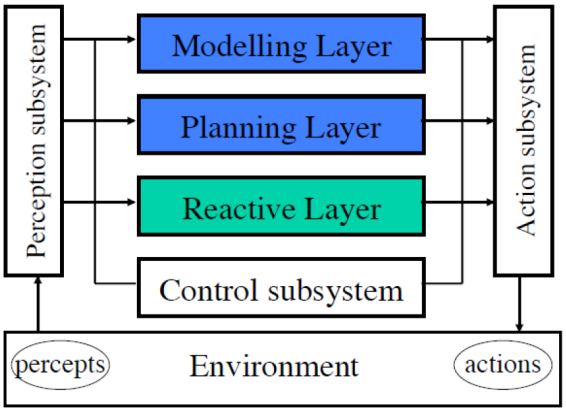



## TouringMachines

演示场景 - TouringMachines 是在 TouringWorld 的街道上行驶的自动驾驶汽车。环境包含障碍物，交通信号灯，雨（影响制动距离）和雾（改变 TouringMachines 视野和范围）。

TouringMachines 的目标是在规定时间内到达指定地点，避免与障碍物和其他 TouringMachines 发生碰撞，并遵守交通规则。

TouringMachines 由三个 horizontal activity-producing layers 组成，每一层不断生成有关代理应执行哪些操作的建议。




注：Hybrid Agents 要干两件事：deliberative 和 reactive。Modelling layer 和 Planning layer 是 deliberative；Reactive layer 是 reactive。

Reactive layer：

- 负责对环境变化做出即时响应 (比如避障)
- 作为一组 condition action rules 实现，这些规则将感知直接映射到操作
- 规则只能反应代理的当前状态，它们不能对世界进行任何显式的推理

Planning layer：

- 负责实现简单的目标，例如从一个地方移动到另一个地方
- 作为预定义的计划架构库 (library of predefined plan schemas) 实现，运行时会详细说明，以便决定要遵循的计划
- 为了实现目标，planning layer 会尝试在计划库中查找与该目标匹配的方案 (schemas)

- 如果方案包含子目标，则 planning layer 将尝试在计划库中查找与每个子目标匹配的方案

Modelling layer：

- 负责表示世界上的其他实体（或代理，包括代理本身）
- 预测代理之间的冲突，并生成新目标以解决这些冲突
- 这些目标被传递给 planning layer，计划以正常方式实现它们

Control subsystem：

- 负责决定在任何给定的时间应控制哪些 layer（负责确定这些 layer 如何一起工作）
- 作为一组 control rules 实现，这些规则可以
  - 抑制 perceptual subsystem 输出的感知
  - 或者，审查 control layers 生成的操作
- 例如，控制规则可以防止 reactive layer 知道已经感知到的特定的障碍物，如果另一个层更适合处理这种类型的障碍物

### Integration

TouringMachines 架构可以被视为分层的 (hierarchical)，因为存在一个子系统 (layer) 可以有效地做出所有控制决策。

在这种架构中，设计人员必须考虑各层之间所有可能的交互作用。如果有 n 个层，并且每个层都可以建议 k 个操作，则意味着需要考虑 $k^n$ 个交互。

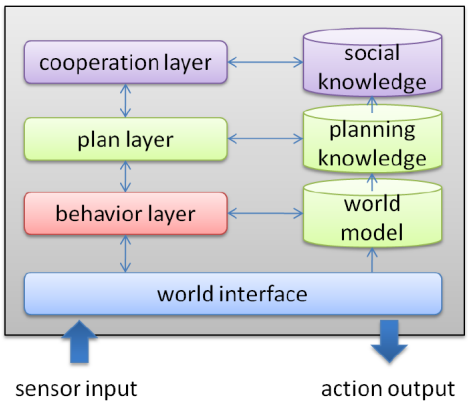
## InteRRaP

InterRRaP uses a vertically layered two-pass architecture。

它包含三层：

- ***behavior-based layer*** deals with reactive behavior
- ***local planning layer*** 处理日常计划以实现代理的目标
- ***cooperative planning layer*** deals with social interactions

每一层都有一个相关的 knowledge base（即，适合于此层的世界的表示）- 从“原始”信息到复杂的模型。




### Layer Interactions in InterRRaP

- Bottom-up activation

  较低层将控制权传递给较高层，因为它无法胜任处理当前情况

- Top-down execution

  较高层利用较低层提供的设施来实现其目标，因为较高层无法直接行动

- 基本的控制权流动 (flow of control) 开始于 perceptual input 到达最低层时，之后控制权流动到最高层，再流回最底层