Chapter 11
现实世界的规划与行动

1 时间、调度、资源
2 分层规划
3 非确定性问题的规划

时间、调度、资源

Review： 经典规划

$Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK)$
$\land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2)$
$\land Airport(JFK) \land Airport(SFO))$
$Goal(At(C_1, JFK) \land At(C_2, SFO))$
$Action(Load(c, p, a),$
PRECOND: $At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $\neg At(c, a) \land In(c, p))$
$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $At(c, a) \land \neg In(c, p))$
$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$
EFFECT: $\neg At(p, from) \land At(p, to))$

- solution: [*Load*(C_1, P_1, *SFO*), *Fly*(P_1, *SFO*, *JFK*), *Unload*(C_1,P_1,*JFK*), *Load*(C_2, P_2, *JFK*), *Fly*(P_2, *JFK*, *SFO*), *Unload*(C_2,P_2,*SFO*)]

经典规划不能做什么？{11.1}

- 经典规划能做什么？
  - 做什么动作 YES
  - 按什么顺序做 YES
  - 动作何时发生 NO
  - 持续多久 NO
  - 考虑资源约束 NO

经典规划不能做什么？{11.1}

- 经典规划表示不能讨论**时间**：动作*何时*发生、持续*多久*。经典规划器(ch10)能为航班制定调度，哪架飞机分配到哪次航班，但实际上需要同时知道出发时间和到达时间。这是**调度**的主要问题。
- 经典规划不能讨论**资源约束**。例如一个航班有有限数量的乘务员，一个航班的乘务员不能同时在另一个航班上。

如何解决？{11.1}

- “先规划，后调度”：把整个问题分解为一个规划阶段和一个接下来的调度阶段，在规划阶段选择动作，考虑次序约束，满足问题的目标，在调度阶段时间信息加入到规划中以满足资源和期限约束。

如何解决？{11.1}

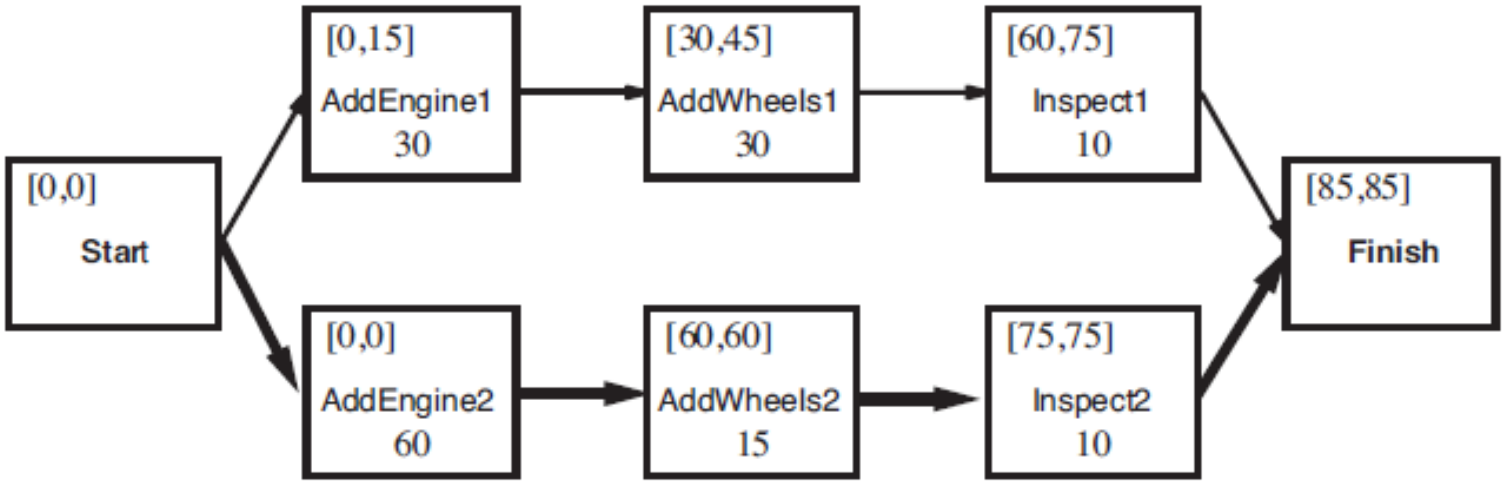
- “先规划，后调度”
  - 做什么动作　　规划阶段
  - 按什么顺序做　　规划阶段
  - 动作何时发生　　调度阶段
  - 持续多久　　调度阶段
  - 考虑资源约束　　调度阶段

时间和资源{11.1.1}

- 车间调度问题
  - 每个动作有一个持续时间（duration）和一组动作所要求的资源约束。
  - **解**必须规定每个动作的开始时间，而且必须满足时间次序约束和资源约束
  - 假设代价函数是**完工时间**(从开始到结束时间长)
  - 用数量表示资源——例如*Inspectors*(2)
    - 当一个调度有10个并发的*Inspect*动作但只有9个检查员时，如果检查员表示为个体，算法会回溯以尝试将检查员分配到动作的10种方法

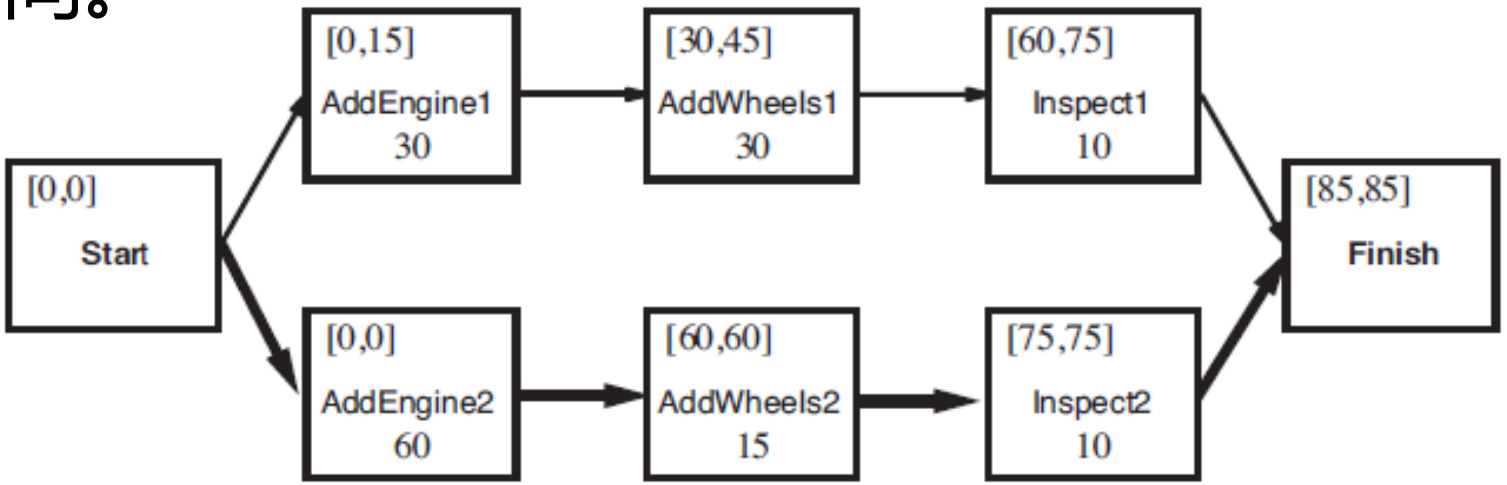
忽略资源约束的时间调度问题：关键路径方法{11.1.2}

- 穿过偏序规划图的一条路径从Start开始，以Finish结束。
- 最短完工时间



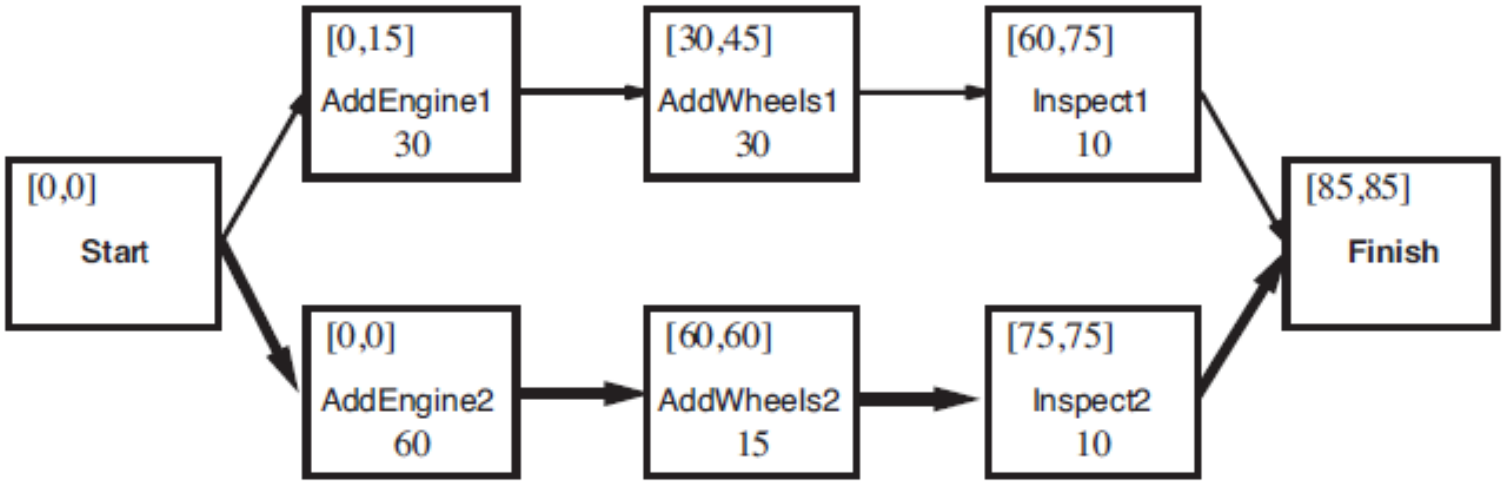
忽略资源约束的时间调度问题：关键路径方法{11.1.2}

- 穿过偏序规划图的一条路径从Start开始，以Finish结束。
- 关键路径是时间跨度最长的路径。延误关键路径上任一动作的开始时间就会延长规划时间。



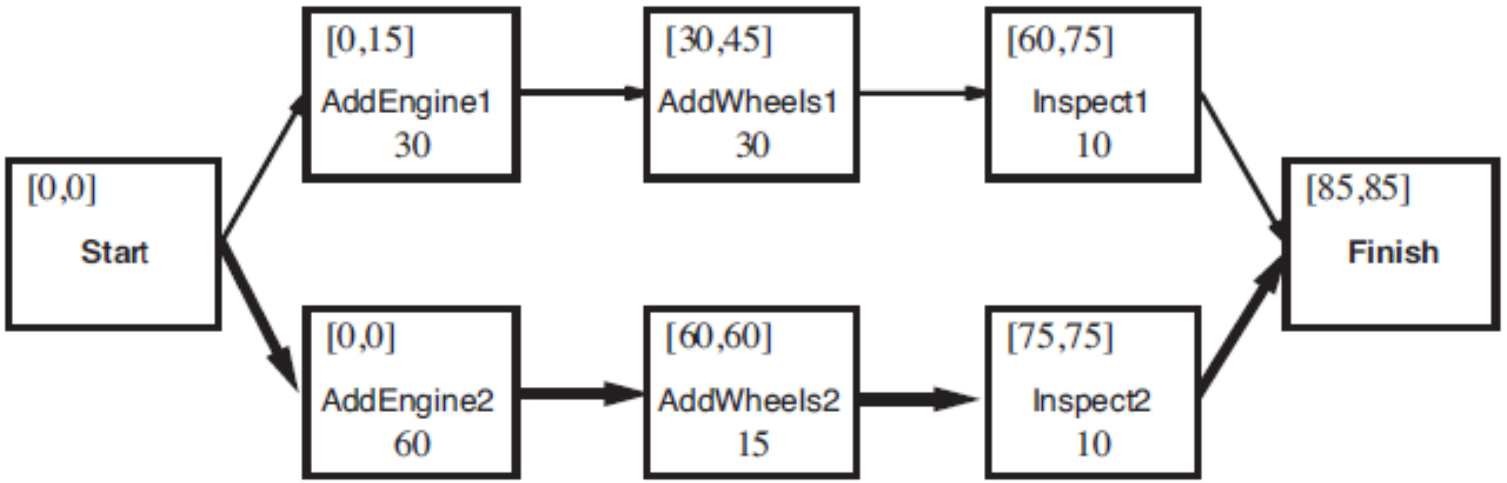
忽略资源约束的时间调度问题：关键路径方法{11.1.2}

- 不在关键路径上的每个动作有个时间窗口，给出该动作最早可能开始时间ES和最晚可能开始时间LS。
- 所有动作的ES和LS一起构成问题的调度。

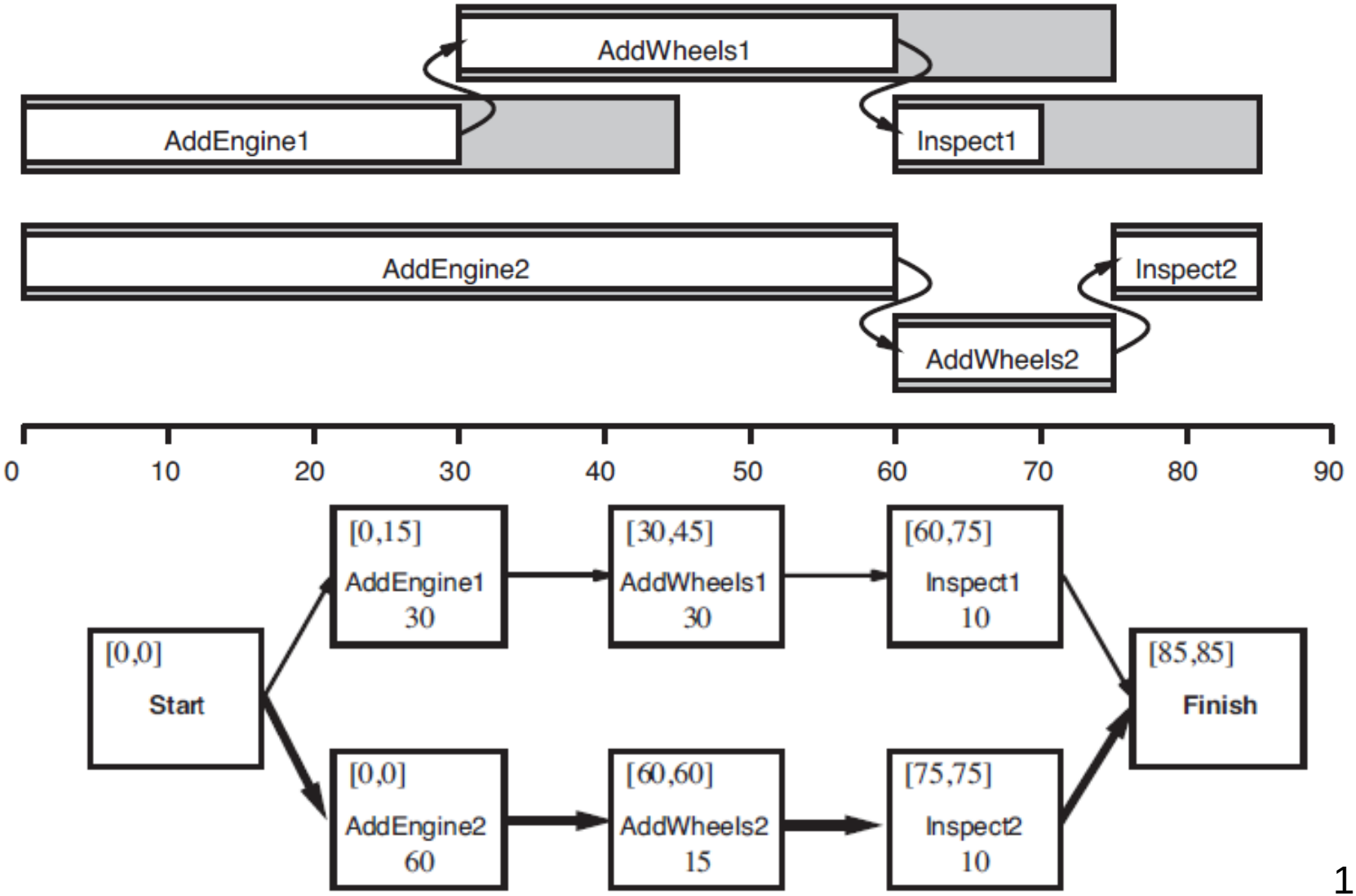


忽略资源约束的时间调度问题：关键路径方法{11.1.2}

- $ES(Start) = 0$
- $ES(B) = \max_{A|A \prec B}\{ES(A) + Duration(A)\}$
- $LS(Finish) = ES(Finish)$
- $LS(A) = \min_{B|A \prec B}\{LS(B) - Duration(A)\}$



忽略资源约束的时间调度问题：关键路径方法{11.1.2}

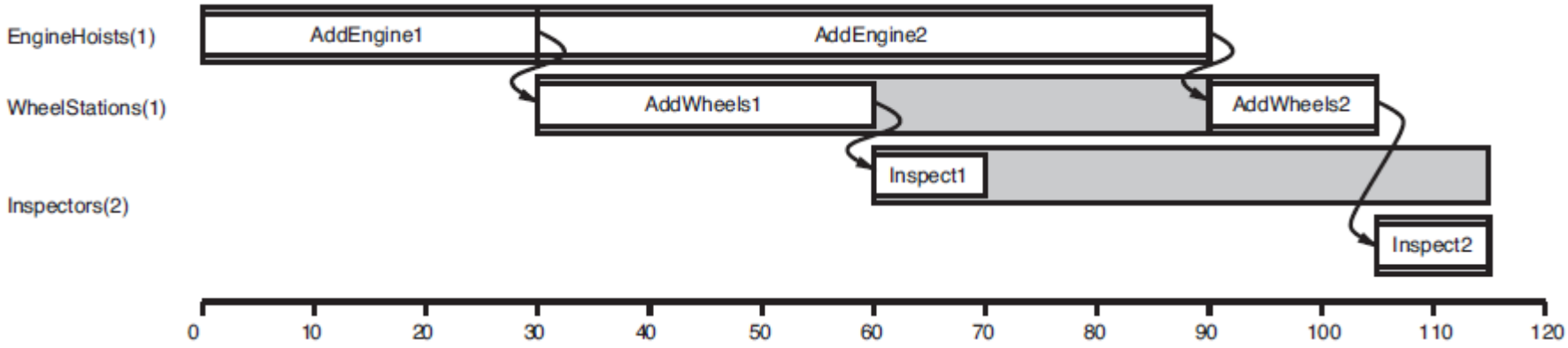


考虑资源约束

有资源约束的时间调度问题 {11.1.2}

- 当引入资源约束。例如，*AddEngine*动作需要相同的*EngineHoist*，因此不能重叠。

“不能重叠”约束是两个线性不等式的析取，每个可能的次序一个



在更高抽象层次上规划

分层规划

层次化分解——为什么需要在更高抽象层次上规划?{11.2}

- 前面的问题求解与规划方法都有一组固定的原子动作；最新的算法可以生成含有几千个动作的解。

- 对于人脑执行的规划，原子动作是肌肉活动。我们大约有10^3块肌肉需要触发，每秒可控制它们的活动10次。“两个星期假期”的详细规划包含大约10^{10}个动作。

层次化分解——为什么需要在更高抽象层次上规划?{11.2}

- 将一个任务分解为大量的单个行动；对于大规模问题，这完全是不切实际的
- 通过层次化分解解决这个问题。

去檀香山度两个星期假：
去旧金山机场；搭乘夏威夷11号航班去檀香山；度假两周；搭乘夏威夷12号航班回到旧金山；回家

去旧金山机场：
开车去长期停车场；停车；搭巴士去航站楼

....

继续分解，直到无需生成马达控制序列就可执行的动作层次

层次化分解——为什么需要在更高抽象层次上规划?{11.2}

- AI系统不得不在更高抽象层次上规划。夏威夷假期的合理规划可能是“去旧金山机场；搭乘夏威夷11号航班去檀香山；度假两周；搭乘夏威夷12号航班回到旧金山；回家。”
- “去旧金山机场”本身可看作是一个规划任务，其解像“开车去长期停车场；停车；搭巴士去航站楼。”
- 其中每个动作又可以继续分解，直到无需生成马达控制序列就可执行的动作层次。

高层动作及其细化{11.2.1}

- **原语动作**(primitive action)：不能再细化，具有标准的前提-效果模式
- **高层动作**(HLA, high level action)：每个HLA有一个或多个可能的**细化动作序列**，其中每个动作可以是一个HLA或一个原语动作

$$
\begin{aligned}
&Refinement(Go(Home, SFO),\\
&\quad \text{STEPS: } [Drive(Home, SFOLongTermParking),\\
&\qquad\qquad Shuttle(SFOLongTermParking, SFO)]\ )\\
&Refinement(Go(Home, SFO),\\
&\quad \text{STEPS: } [Taxi(Home, SFO)]\ )
\end{aligned}
$$

高层动作及其细化{11.2.1}

- 一个只包含原语动作的HLA的细化被称为该HLA的**实现**
- 高层规划是HLA序列，高层规划的**实现**是该HLA序列中每个HLA的实现的拼接。

HLA可能恰好只有一个实现；也可能有多个可能的实现

高层动作及其细化{11.2.1}

- 一个HLA恰好有一个实现时，我们能够从该实现的前提和效果中计算出这个HLA的前提和效果（习题11.3），然后这个HLA本身就可看作是一个原语动作。

11.3 假设一个高层动作刚好可用一个原语动作序列实现。给定完整的细化层次和原语动作模式，给出一个计算它的前提和效果的算法。

高层动作及其细化{11.2.1}

11.3 假设一个高层动作刚好可用一个原语动作序列实现。给定完整的细化层次和原语动作模式，给出一个计算它的前提和效果的算法。

```
net_preconditions <- {}
net_effects <- {}
remaining <- [h]
while remaining not empty:
   a <- pop remaining
   if a is primitive:
      add to net_preconditions any precondition of a not in effects
      add to net_effects the effects of action a, first removing any
      complementary literals
   else:
      r <- the unique refinement whose preconditions do not include
      literals negated in net_effect or net_preconditions
      add to net_preconditions any preconditions of r not in effect
      prepend to remaining the sequence of actions in r
```

高层动作及其细化{11.2.1}

- 当HLA有多种可能的实现时，有两种选项：(1)搜索原语解：搜索一个可以有效工作的实现；(2)搜索抽象解:直接对HLA推理——，无需搜索出它们的实现。

如何搜索原语解

搜索原语解——搜索一个可以有效工作的实现{12.2.2}

- 反复在当前规划中选择一个HLA，用它的细化替换它，直到规划达到目标。

搜索原语解——搜索一个可以有效工作的实现{12.2.2}

- 对于一个非层次化的、每个状态有b个可用动作的前向状态空间规划器，代价是$O(b^d)$

搜索原语解——搜索一个可以有效工作的实现{12.2.2}

- 对于一个非层次化的、每个状态有b个可用动作的前向状态空间规划器，代价是$O(b^d)$
- 对于一个HTN (层次任务网络)规划器，假设一个一般的细化结构：每个非原语动作有r个可能的细化，每个细化有k个动作。如果在原语层有d个动作，那么在根下方的层数是$\log_k d$，因此内部细化结点数是$1+k+k^2+...+k^{\log_k d-1}=(d-1)/(k-1)$。每个内部结点有$r$个可能的细化，因此可构建出$r^{(d-1)/(k-1)}$可能的分解树。小$r$和大$k$可以导致大量的节省。

搜索抽象解

搜索抽象解{11.2.3}

- 我们确信含有两个HLA的高层规划：

[*Drive*(*Home*, *SFOLongTermParking*), *Shuttle*(*SFOLongTermParking*, *SFO*)]

无需确定精确路径、停车点等等就能让我们到达机场

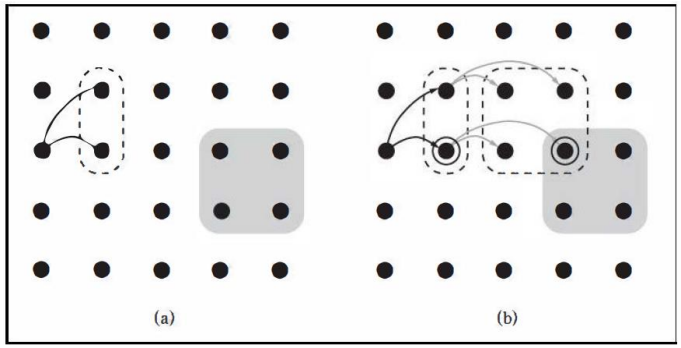
搜索抽象解{11.2.3}

- 如果得到的高层规划可证明能达到目标（那么这个高层规划就是抽象解），那么我们可以致力于这个规划，对这个规划的每一步进行细化。这样，搜索得到指数量级的缩减。

- 每个声明能达到目标的高层规划是在“它至少有一个实现能达到目标”的意义上能达到目标。这个特性被称为是HLA描述的**向下细化特性**。

可到达集{11.2.3}

- 给定一个状态*s*，一个HLA *h*的**到达集**记为REACH(*s*,*h*)，是这个HLA的任一实现可到达的状态集合
- REACH(s,h1)，　REACH(s,[h1,h2])



h1和**h2**都有两种可能的实现

- 在高层规划中搜索，找出一个高层规划其可到达集与目标相交。

HLA的效果{11.2.3}

- 当HLA只有恰好一个实现时，我们能够从该实现的前提和效果中计算出这个HLA的(前提和)效果（习题11.3）

- 当HLA有多个可能的实现时，如何描述效果？

HLA的效果的(精确)描述{11.2.3}

- 用符号~表示“可能，如果Agent这样选择”

$$Action(h_1, \text{PRECOND: } \neg A, \text{ EFFECT: } A \wedge -B)$$

$$Action(h_2, \text{PRECOND: } \neg B, \text{ EFFECT: } +A \wedge \pm C)$$

- h1有个实现的效果是A,另一实现效果是A∧¬B
- 如果初始状态只有*B*为真，而且目标是$A \wedge C$，那么序列[h_1,h_2]达到目标：我们选择一个使*B*为假的h_1的实现，然后选择一个保持*A*为真使*C*为真的和h_2的实现。

HLA的效果的近似描述{11.2.3}

- 很多情况下一个HLA可能有无限多的实现，我们只能对效果进行近似，从而可能产生任意摇摆不定的可到达集
- 使用两种近似：一个HLA动作*h*的**乐观描述**（optimistic description）$\mathrm{REACH}^{+}(s,h)$可能夸大可到达集，**悲观描述**（pessimistic description）$\mathrm{REACH}^{-}(s,h)$可能低估可到达集。

$$\mathrm{REACH}^{-}(s,h) \subseteq \mathrm{REACH}(s,h) \subseteq \mathrm{REACH}^{+}(s,h)$$

搜索抽象解
精确描述下如何进行目标测试？
(判断一个规划能否能够达到目标)

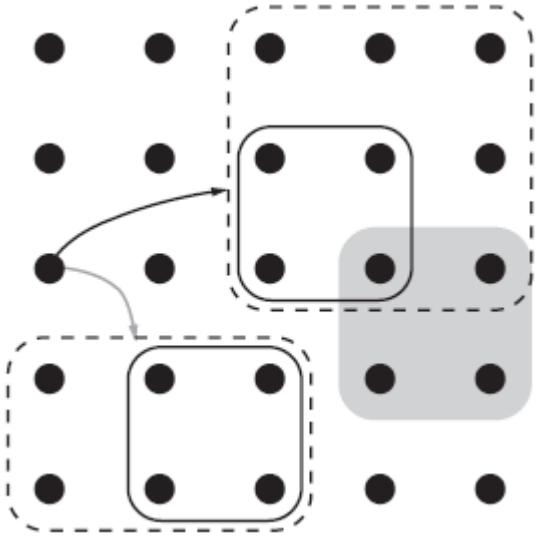
如何进行目标测试{11.2.3}

- 在精确描述下，一个规划或者工作(其可到达集与目标相交)或者不工作(其可到达集与目标不相交)。

搜索抽象解
近似描述下如何进行目标测试？

如何进行目标测试{11.2.3}

- 近似描述下：
- 如果乐观可到达集与目标不相交，那么这个规划是不能工作的；
- 如果悲观可到达集与目标相交，那么这个规划是能够工作的。

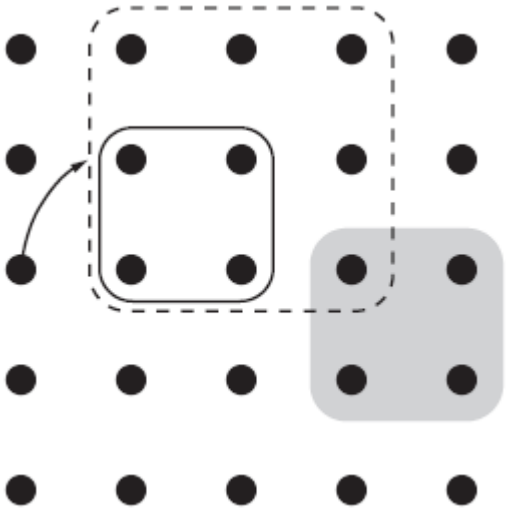


黑色箭头指示的规划肯定达到目标，而灰色箭头指示的规划肯定不会达到目标。

目标状态集加上了阴影。实线：悲观可到达集。虚线：乐观可到达集。

如何进行目标测试{11.2.3}

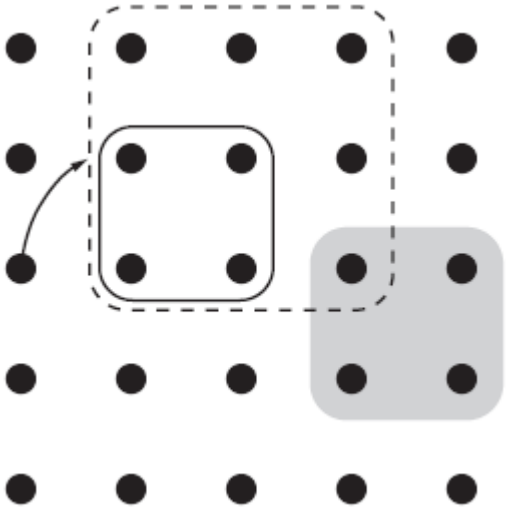
- 在近似描述下有中间状态：如果乐观集与目标相交，但悲观集与目标不相交，那么我们不能确定规划是否工作。当出现这种情况，怎么办？



目标状态集加上了阴影。实线：悲观可到达集。虚线：乐观可到达集。

如何进行目标测试{11.2.3}

- 但在近似描述下有中间状态：如果乐观集与目标相交，但悲观集与目标不相交，那么我们不能确定规划是否工作。当出现这种情况，通过进一步细化规划可以消除这种不确定性。



需要进一步细化以确定是否确实达到目标

目标状态集加上了阴影。实线：悲观可到达集。虚线：乐观可到达集。

非确定性问题的规划

- 用于无观察的环境中的无传感器规划（sensorless planning）；
- 用于部分可观察的、非确定性环境中的应急规划（contingency planning）；
- [用于未知环境中的在线规划（online planning）和重新规划（replanning）。]

涂色问题

- 考虑这个问题：给定一把椅子和一张桌子，目标是对其进行匹配——有相同颜色。初始状态我们有两罐颜料，但颜料和家具的颜色未知。只有桌子开始时在Agent的视眼内：
- $Init(Object(Table) \wedge Object(Chair) \wedge Can(C_1) \wedge Can(C_2) \wedge InView(Table))$
- $Goal(Color(Chair,c) \wedge Color(Table,c))$

涂色问题

- 有两个动作模式：从颜料罐去掉盖子，使用打开的罐子中的颜料涂抹对象。允许前提和效果包含不属于动作变量列表中的变量（因为这是在部分可观察的情况下；在完全可观察的情形下，这是不允许的）
  - *Action*(*RemoveLid*(*can*),
    PRECOND: *Can*(*can*)
    EFFECT: *Open*(*can*))
  - *Action*(*Paint*(*x*,*can*),
    PRECOND: *Object*(*x*) $\wedge$*Can*(*can*)
    $\wedge$*Color*(*can*,*c*) $\wedge$*Open*(*can*)
    EFFECT: *Color*(*x*,*c*))

涂色问题

- Agent需要一个使对象（每次一个）进入到视线里的动作模式：
- *Action*(*LookAt*(*x*),

  PRECOND: *InView*(*y*) $\wedge$ ($x \neq y$)

  EFFECT: *InView*(*x*) $\wedge \neg$*InView*(*y*))

涂色问题

- 为了求解部分可观察问题，Agent实际行动时，它的传感器将提供感知信息，但当它进行规划时，它将需要它的传感器模型。第4章中，这个模型是由一个函数给定的，PERCEPT(*s*)。对于规划，我们用感知模式（percept schema）来扩展PDDL：
  - *Percept*(*Color*(*x*,*c*),
    PRECOND: *Object*(*x*) ∧*InView*(*x*))
  - *Percept*(*Color*(*can*,*c*),
    PRECOND: *Can*(*can*) ∧*InView*(*can*) ∧*Open*(*can*))

涂色问题

- 对于一个完全可观察的环境，对于每个流（fluent）我们将有一个没有前提的感知模式。

- 另一方面，一个无传感器的Agent根本没有感知模式。

- 无传感器+确定性

无传感器的规划{11.3.1}

- 无传感器+确定性
  - 信念状态规划

- 信念状态规划与信念状态空间搜索的差异
  - 主要差异是，潜在的物理转移模型由一组动作模式表示，信念状态可以用一个逻辑公式而不是一组显式枚举的状态表示。
  - 为了简化，我们假设潜在的规划问题是确定性的。

确定初始信念状态{11.3.1}

- 涂色问题
- 忽略*InView*流，因为Agent没有传感器
- 将$Object(Table) \wedge Object(Chair) \wedge Can(C_1) \wedge Can(C_2)$作为给定的不发生变化的事实
- Agent不知道罐子里颜料或对象的颜色，也不知道罐子是开着的还是盖上的，但知道对象和罐子有颜色：$\forall x \exists c\ Color(x,c)$。Skolem化以后，获得初始信念状态：$b_0 = Color(x, C(x))$

确定初始信念状态{11.3.1}

- 在经典规划中我们使用封闭世界假设：一个状态中没有提到的任何流（fluent）为假（状态只包含正流，如果一个流不出现，他的值就为假）

- 在无传感器的规划中我们需要切换到开放世界假设，其中状态包含正流和负流，而且如果一个流不出现，它的值就是未知的。

转移模型{11.3.1}

- *ch04确定性环境里更新信念状态：*
  b'=RESULT(b,a) = {s':
  s'=RESULT$_P$(s,a) and $s \in b$}
- *ch10的正向搜索(封闭世界假设下)：*
  b'=RESULT(b,a)=(b−DEL(a))∪ADD(a)

- 开放世界假设下：如何计算RESULT(b,a)？

转移模型{11.3.1}

- *ch04确定性环境里更新信念状态：*
  b'=RESULT(b,a) = {s':
  s'=RESULT$_P$(s,a) and $s \in b$}
- *ch10的正向搜索(封闭世界假设下)：*
  b'=RESULT(b,a)=(b−DEL(a))∪ADD(a)

- 开放世界假设下：RESULT(b,a)的计算从b开始，将DEL(a)中出现的任何原子设置为假，将ADD(a)中出现的任何原子设置为真

当效果依赖状态时的转移模型{11.3.1}

- 当动作的效果依赖于状态时会有问题。

- 例如吸尘器问题，吸尘器在左边，执行吸尘动作，左边变干净，吸尘器在右边，执行吸尘动作，右边变干净。

当效果依赖状态时的转移模型 {11.3.1}

- 当动作的效果依赖于状态时会有问题，引入"条件效果"
- when *condition*: *effect*
- *Action*(*Suck*, EFFECT: **when** *AtL*: *CleanL* ∧**when** *AtR*: *CleanR*)
  - 当初始信念状态为真，结果信念状态就是 (*AtL*∧*CleanL*)∨(*AtR*∧*CleanR*)， 不再是1-CNF
  - 信念状态发生了摇摆
  - 如何处理摇摆？

处理摇摆：分裂动作{11.3.1}

- 将*Suck*分裂为两个具有非条件效果的动作
  - *Action*(*SuckL*, PRECOND: *AtL*, EFFECT: *CleanL*)
  - *Action*(*SuckR*, PRECOND: *AtR*, EFFECT: *CleanR*)
- 现在只有非条件模式，因此信念状态都保持在1-CNF里；不幸的是，不能在初始信念状态里确定*SuckL*和*SuckR*的适用性。

处理摇摆：保守近似{11.3.1}

- 保守近似用于精确信念状态：信念状态保持在1-CNF里，它包含所有真值能确定的文字，将其他文字视为未知。
- 它从不生成不正确的规划，但它是不完备的
- 实例，如果目标是要机器人在一个干净的方格里，那么[Suck]就是一个解，但一个坚持1-CNF信念状态的无传感器Agent将不能找到这个解。

- 考虑部分可观察和非确定性

应急规划{11.3.2}

- **应急规划**（contingent planning）——带有基于感知的条件分支的规划生成

- 对于部分可观察或非确定性环境或同时是部分可观察和非确定性的环境是合适的。

应急规划{11.3.2}

涂色问题的应急规划

[*LookAt*(*Table*), *LootAt*(*Chair*),

if *Color*(*Table*, *c*) ∧*Color*(*Chair*,*c*) **then** *NoOp*

else [*RemoveLid*(Can_1), *LootAt*(Can_1), *RemoveLid*(Can_2), *LookAt*(Can_2),

if *Color*(*Table*,*c*) ∧ *Color*(*can*,*c*) **then** *Paint*(*Chair*, *can*)

else if *Color*(*Chair*,*c*) ∧ *Color*(*can*,*c*) **then** *Paint*(*Table*,*can*)

else [*Paint*(*Chair*,Can_1), *Paint*(*Table*,Can_1)]]]

变量应该考虑用存在量词量化

评估每个分支条件的方式是通过确定信念状态是否蕴涵这个条件公式或它的否定

新信念状态的确定{11.3.2}

- 第一个阶段在动作之后预测信念状态

$$\hat{b} = (b\text{-DEL}(a) \cup \text{ADD}(a)$$

- 第二个阶段，感知之后更新信念状态
- 假设接收到了感知文字$p_1,\ldots,p_k$。
- 如果一个感知p恰好有一个感知公理 *Percept*(p,PRECOND:c)，其中c是文字的合取，那么这些文字可以和p一起丢进信念状态。另一方面，如果p有不止一个感知公理，其前提根据预测的信念状态 可能成立，那么我们不得不加进前提的析取。

summary

1 时间、调度、资源
2 分层规划
3 非确定性领域的规划