# 习题答案

## 第一章习题答案

**习题 1.1**

(a) 规则是：

- 顺序：一个接一个；
- 选择：仅有一个任务会被执行，依赖于某个条件；
- 并行：任务可能同时执行或以任何顺序执行；
- 迭代：一个或多个任务会被执行多次。

(b) 迭代不是一个基本构造：它可以用“选择”来表达。

**习题 1.2**

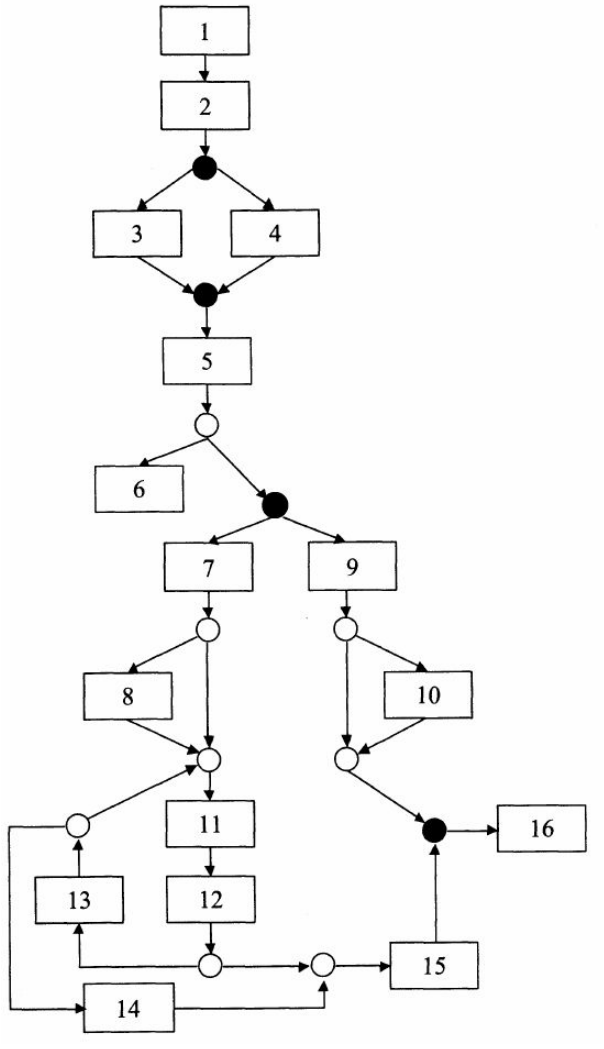


图 S1.1 保险过程

**习题 1.3**

一个过程中的一个任务是由一个资源执行的一个逻辑工作单元。从一个（子）承包人的角度看一个任务是要被执行的一个订单，然而，执行可能需要包含几个任务的过程。

**习题 1.4**

我们可以把人员分成能力小组、功能部门和过程小组。能力小组的优势是具有相同技能的人处于同一个单位，在资源计划方面提供了灵活性。一个缺点是这些单位不直接为一个过程或案例处理负责。过程小组的一个优点是它们集中于过程的执行和有效的案例处理。一个缺点是过程小组间雇员之间的交流比较困难。一个功能部分组织是二者的混合：没有完整的案例处理的职责，但是有责任处理可能超过一个过程的一组任务，这些任务需要类似的技能。

# 第二章习题答案

## 传统 Petri 网

### 习题 2.1 德国交通灯

(a) 可能的状态和变迁系统如图 S2.1 所示：

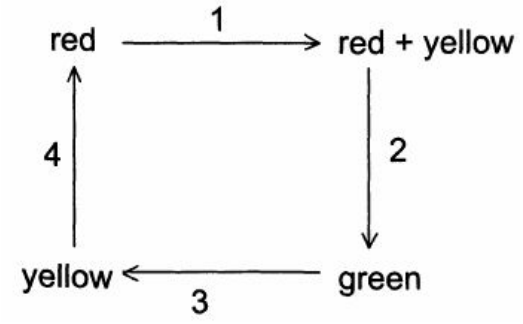


图 S2.1 状态与变迁

(b) 用实线构造的模型能够像一个德国交通灯一样运转，即忽略库所 *c1* 和 *c2*，如图 S2.2 所示。

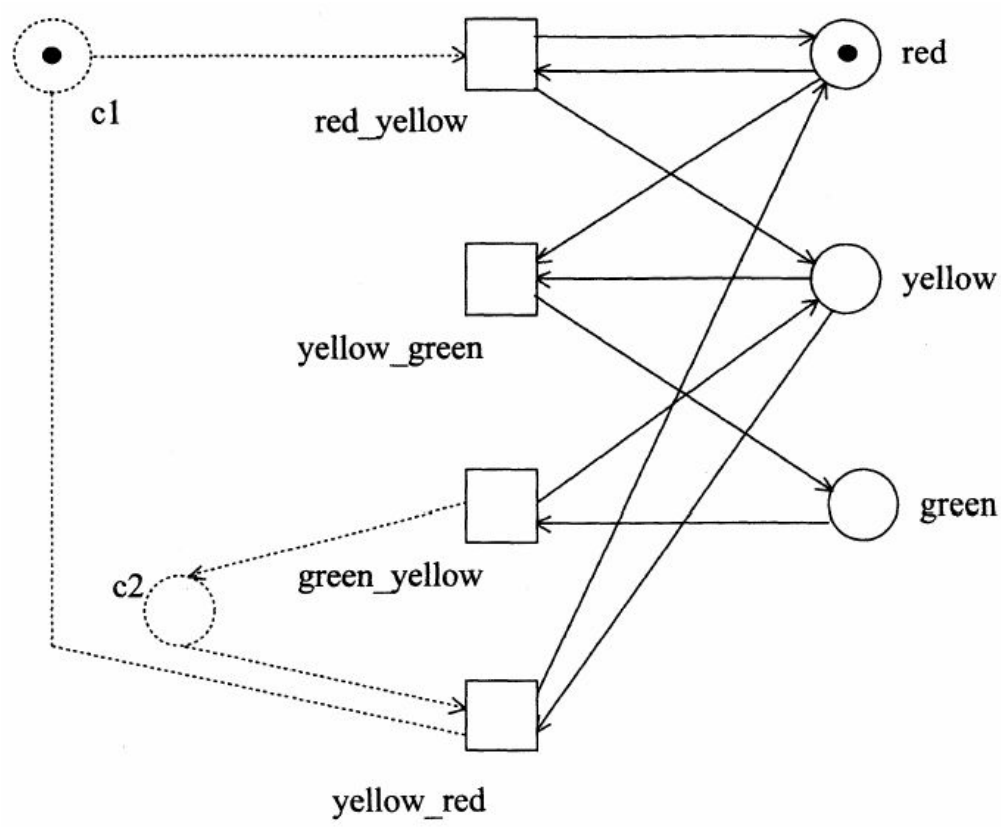


图 S2.2 德国交通灯模型

(c) 虚线库所和弧的添加是必须的，这样才能使得模型像德国交通灯一样运转。否则红绿灯能够正常运转，但是也存在潜在的异常，如：

- 变迁 *red_yellow* 重复实施，而不用切换到 *yellow* 或 *green*，于是导致 *yellow* 中标记的堆积。
- *yellow_red* 有可能在 *green_yellow* 之前实施。

**习题 2.2 项目 X**

(a) 如图 S2.3 所示。

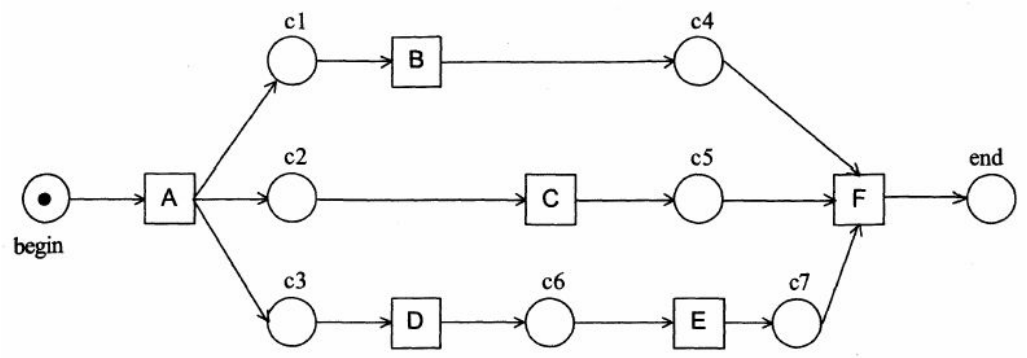


图 S2.3 工程 X

(b) 为了使 *E* 可选择，需要为该变迁增加一个旁路，如图 S2.4。

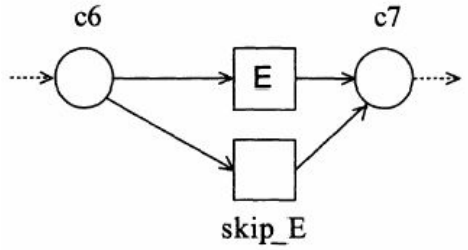


图 S2.4 旁路 E

(c) 引入库所 *c8* 来确保如果变迁 *D* 开始，*B* 和 *C* 不能够被执行，因为它们也需要 *c8* 中的一个标记。当变迁 *E* 结束时，为 *c8* 产生一个标记来使新的变迁成为可能。

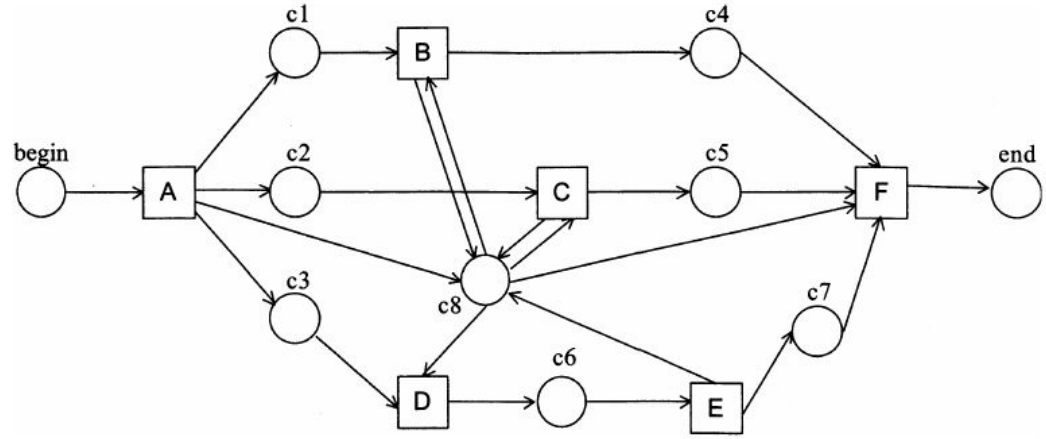


图 S2.5 扩展 c8

**习题 2.3 铁路网**

(a) 一条轨道可以建模为图 S2.6 所示。

一条轨道由三个库所（*b*=繁忙、*c*=申请、*f*=空闲）和它们之间的变迁组成。为了架设支持两辆火车的四条轨道，我们将该轨道拷贝 4 次并在一个 *b* 库所放置两个标记、*f* 库所放置两个标记。

然后我们安排一些额外条件，只有当一列火车成功申请了一条轨道之后才可以转移到该轨道上，在此之前需要查看该轨道是否空闲。需要通过 *b* 库所和 *use_track* 变迁之间的弧来进行判断。

还需要注意两个并发任务的变迁 *use_track* 和 *clear_track* 被同时执行，于是，我们将它们融合在一个变迁 *transfer* 中。

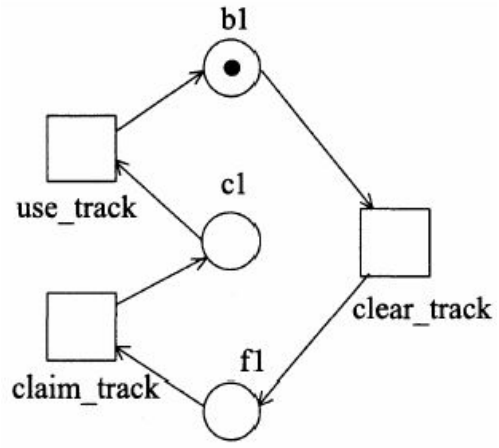


图 S2.6 一条轨道

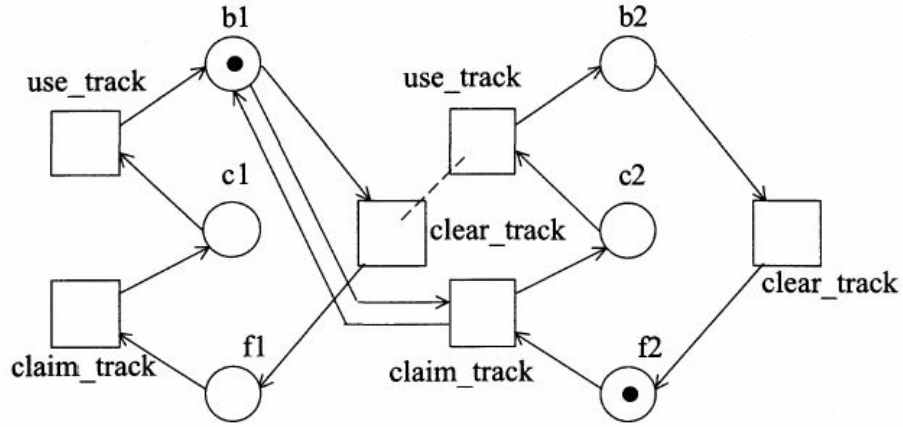


图 S2.7 两条轨道

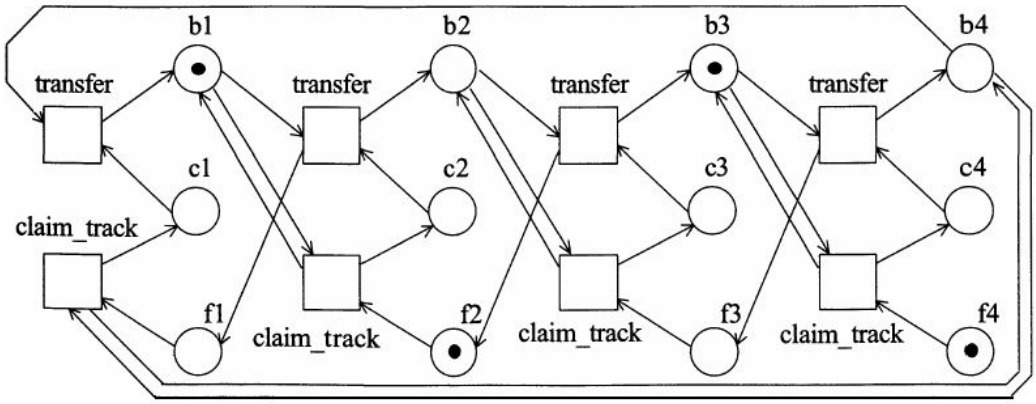


图 S2.8 由四条轨道和两列火车组成的完整系统

(b) 只要添加新的轨道。当状态的总数迅速增长时，Petri 网的大小与轨道的数目保持线性关系。注意状态的数目通过下列公式表达：

$$\frac{n*(n-1)}{2}+(n*(n-2))+\frac{n*(n-3)}{2}=2n(n-2)$$

**习题 2.4 二进制计数器**

很明显不同的状态如下：

| a | b | c | | | a | b | c | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | = | 0 | 1 | 0 | 0 | = | 4 |
| 0 | 0 | 1 | = | 1 | 1 | 0 | 1 | = | 5 |
| 0 | 1 | 0 | = | 2 | 1 | 1 | 0 | = | 6 |
| 0 | 1 | 1 | = | 3 | 1 | 1 | 1 | = | 7 |

得到了图 S2.9 显示的下列模型。

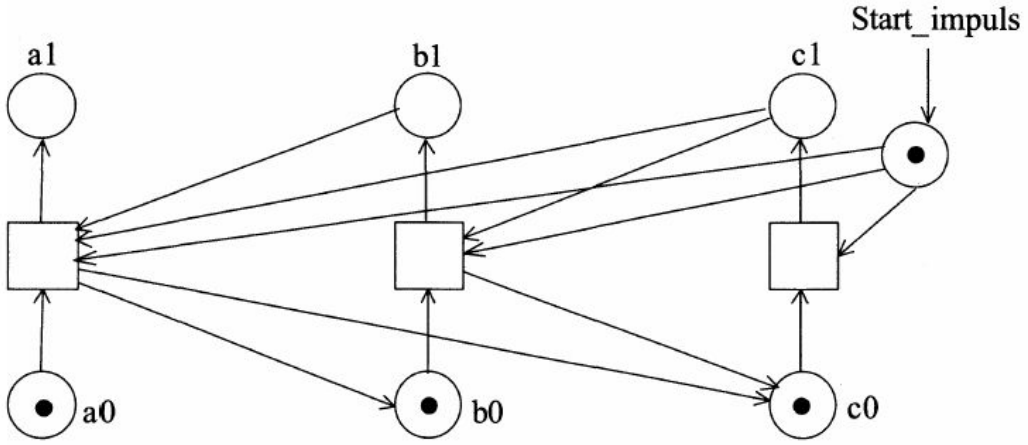


图 S2.9 二进制计数器

库所 *a1* 和 *a0* 表示第一个数字的状态，*b1* 和 *b0* 表示第二个数字的状态，*c1* 和 *c0* 表示第三个数字的状态。

## 高级 Petri 网

**习题 2.5 驾驶学校**

(a) 见图 S2.10。

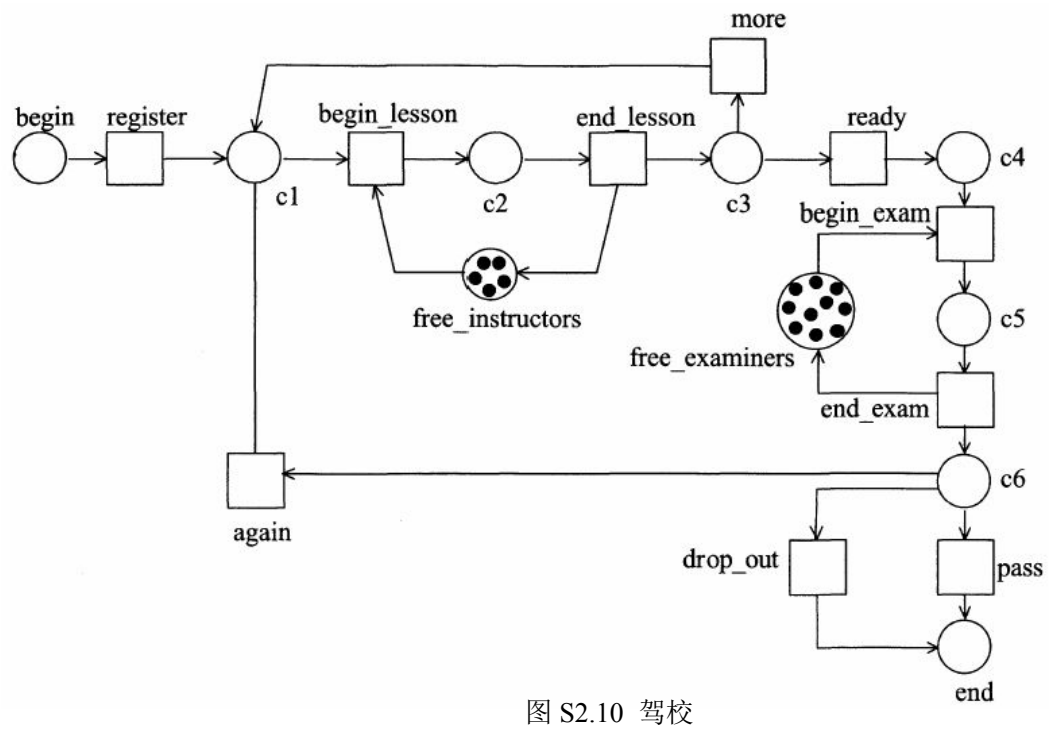


图 S2.10 驾校

(b) 库所 *begin*、*c1*、*c2*、*c3*、*c4*、*c5*、*c6*、*end* 中的每个标记现在都有一个值。例如：一个名字是 J.Walker、18 岁的人，没有上任何课，也没有考试过，表示如下：

[标识：'X07'；名字：'J.Waker'；年龄：'18'；性别：'男'；课数：'0'；考试数：'0']

最后两个属性对于训练很重要，因为我们希望知道一个人已经上了多少次课且通过了几次考试。变迁定义如下：

*register*：nof_lessons:=0

nof_exams:=0

到 *more* 和 *ready* 的变迁可以融合为具有如下行为的一个变迁 *more?*

如果

nof_lessons<10

那么

为 *c1* 产生一个标记

否则

为 *c4* 产生一个标记

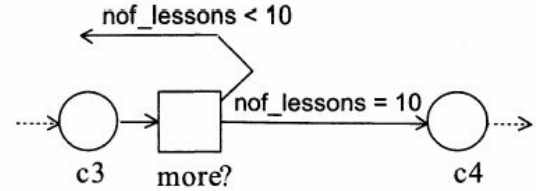


图 S2.11 变迁 more 与 ready 混合为一个变迁 more?

*end_lesson*：nof_lessons:=nof_lessons+1

*end_exams*：nof_exams:=nof_exams+1

*again* 有一个前条件：nof_exams<3

设置属性 nof_lessons:=0，因为一个人在下一次考试前不得不再上 10 次课。

(c) 除了图 S2.12 指出的，所有的延迟都等于 0。

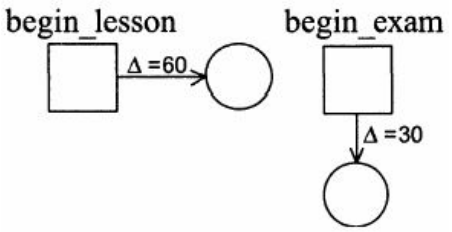


图 S2.12 正的延时的加入

**习题 2.6 自行车制造厂**

(a) 见图 S2.13。

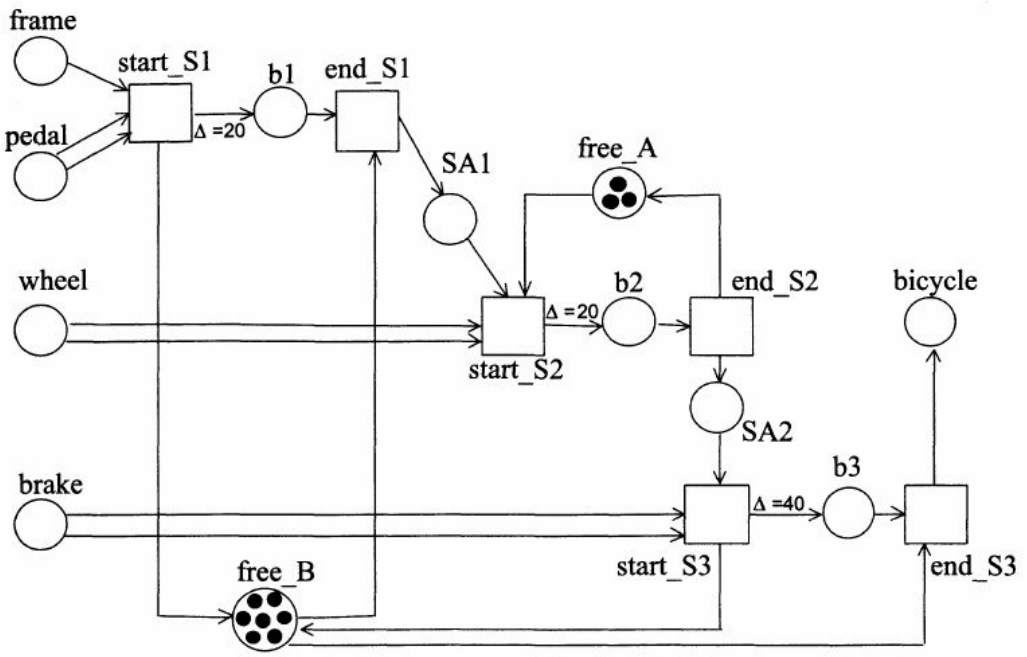


图 S2.13 自行车制造厂

(b) 能力 A：3*(60 分钟/20 分钟 of 行动 SA2)=9 p/h

能力 B：7*(60 分钟/20+40 分钟 of 行动 SA1 和 SA3)=7 p/h

我们识别出机器 B 的能力是瓶颈，而且因此该工厂有能力每小时生产 7 辆自行车。

## 工作流练习

**习题 2.7 保险公司**

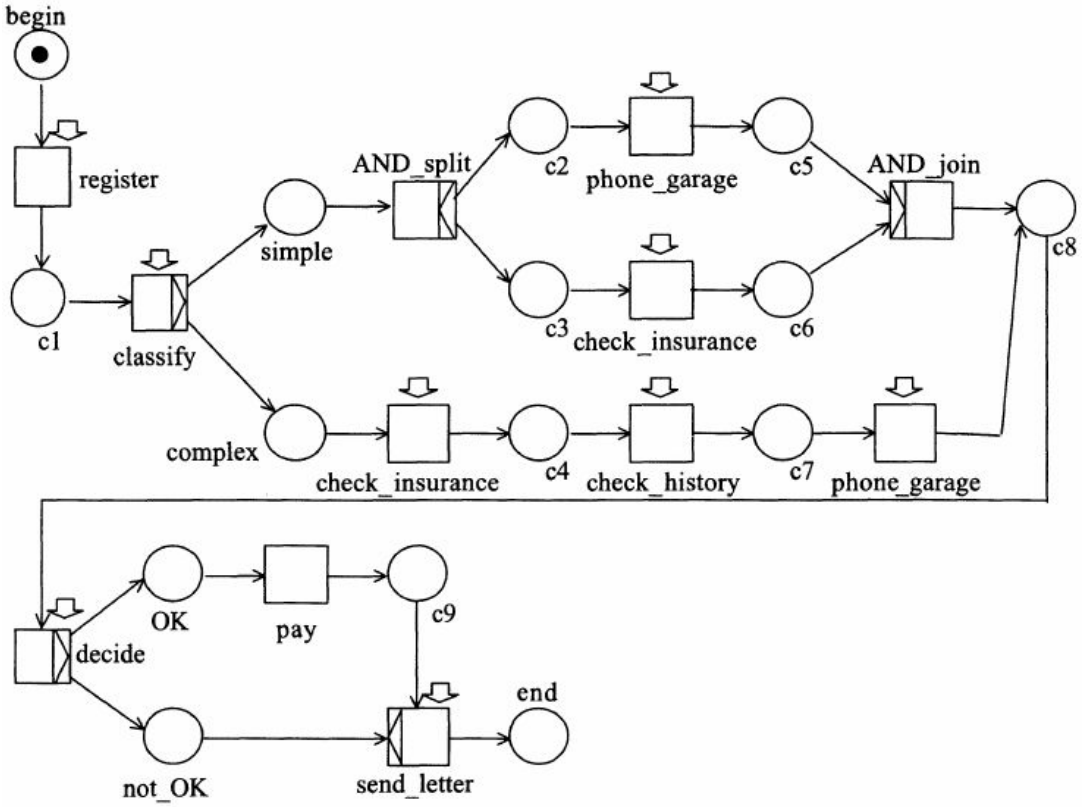


图 S2.14 保险公司

在 *OK* 与 *not_OK* 之间的选择也可以通过把 *not_OK* 与 *c9* 合并为一个库所来实现，在这种情况下，*send_letter* 只有一个输入库所且不需要 OR-join 符号。

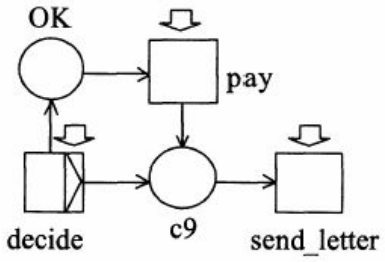


图 S2.15 通过合并库所 c9 与 not_OK 来移除 OR 合并

**习题 2.8 投诉处理**

建模最困难的部分是表单和实际的处理之间的关系：任务处理必须等待直到表单的处理完成而且可能被执行任意多次。在图 S2.16 中，这个问题通过实际处理的两个任务得到解决，即：*process* 和 *process_again*。在图 S2.17 中，只有一个叫做 *process* 的任务，这里 *process* 从 *c9* 取走一个标记，但是立即放回一个。结果是 *process* 可以被执行任意多次而不需要从 *c9* 移走标记。

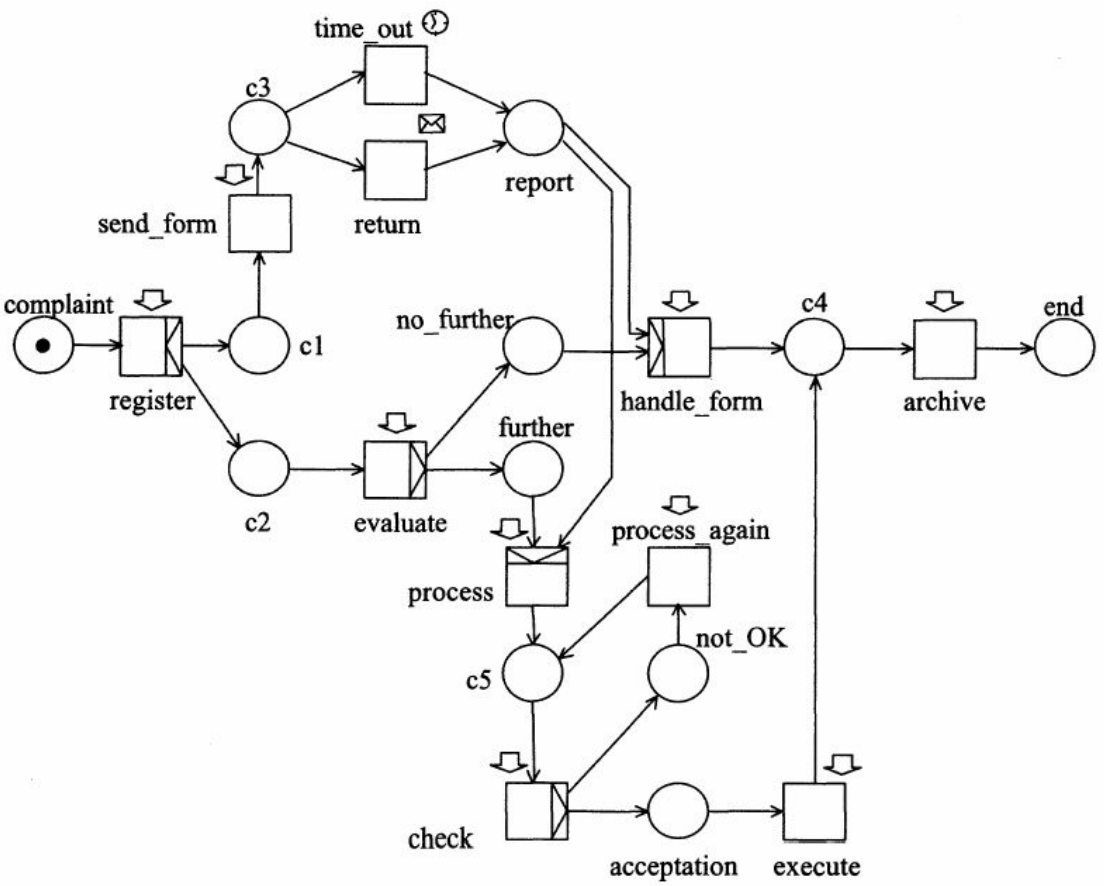


图 S2.16 投诉处理

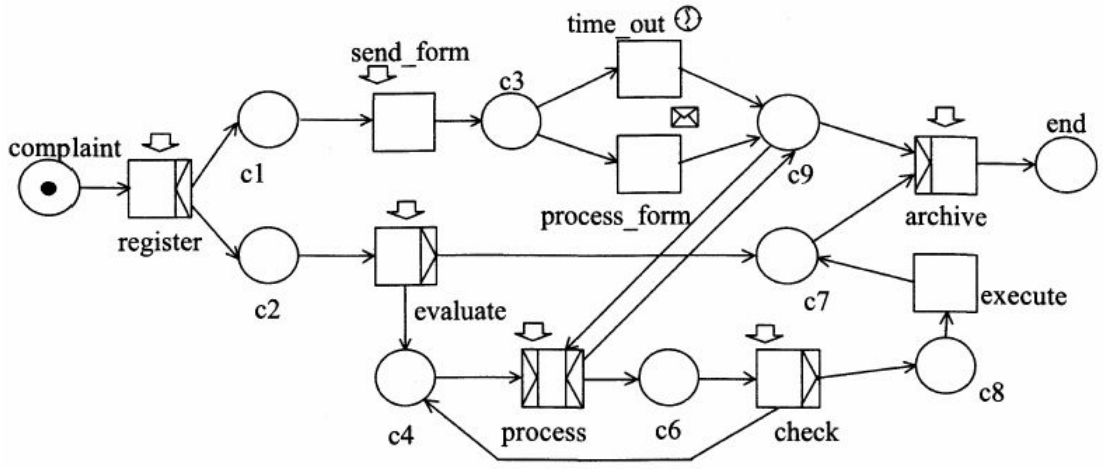


图 S2.17 投诉处理

**习题 2.9 举行聚会**

(a) 可以识别过程的三部分：

- 组织场所
- 组织音乐
- 最终安排（列表、食物、饮料和参观）

前两个部分并行执行，后面跟着第三部分。第二部分（乐曲）是过程中最复杂的部分，需要两个隐式的 OR-split 来处理超时。

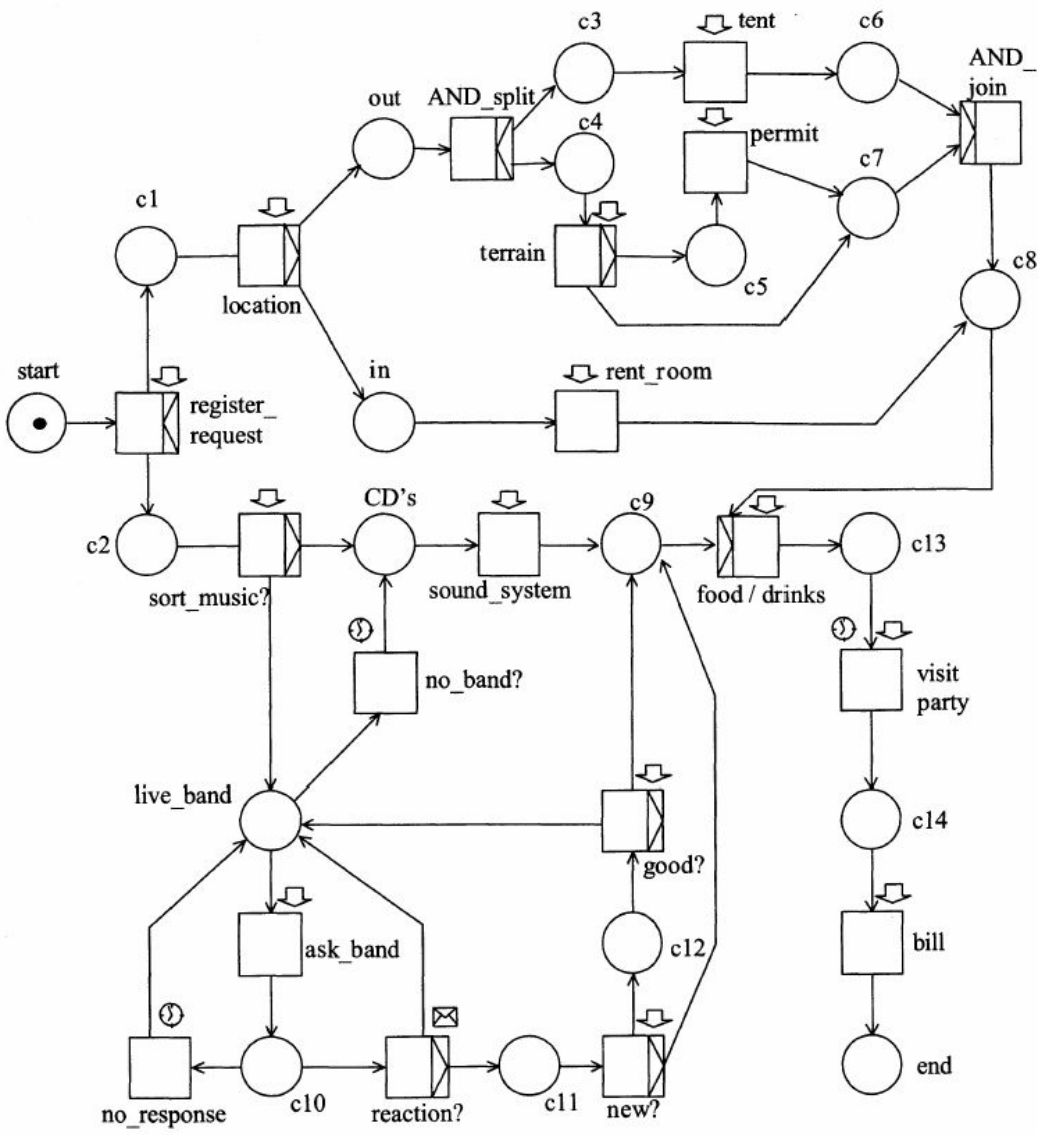


图 S2.18 聚会

过程中最重要的瓶颈是一个乐队的选择，过程的这个部分可能花费很长时间，特别是当一个或两个乐队拒绝或者当一个乐队的演奏太差时。于是，当过程被分为两个分离的过程时，可以获得最大的改进：一个处理聚会的请求，另一个评估乐队。结果是乐队可以独立于特定的聚会请求被评估。

## 第三章习题答案

### 习题 3.1 保险公司

可以识别出下列角色：

雇员 (E)

索赔处理者 (CH)

索赔处理者 A (CHA)

索赔处理者 B (CHB)

可以识别下列组织单元：

轿车损伤部门 (CD)

金融部门　　(FN)

结果模型显示于图 S3.1。

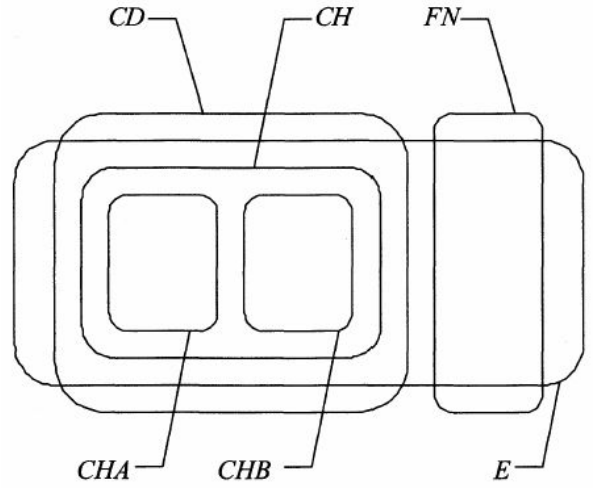


图 S3.1 资源分类保险公司

我们假设所有的索赔处理者也是雇员，这意味着当车辆损伤部门的一个雇员被请求执行一个任务时，并不在乎他或她是否为一名索赔处理者。如果我们假设一个认领处理者不能执行一个普通雇员的任务，那么图 S3.1 需要被修改（CH，CHA 和 CHB 将在 *E* 外面）。

如果我们把资源类别与过程模型结合，我们可以获得图 S3.2 所示的模型：

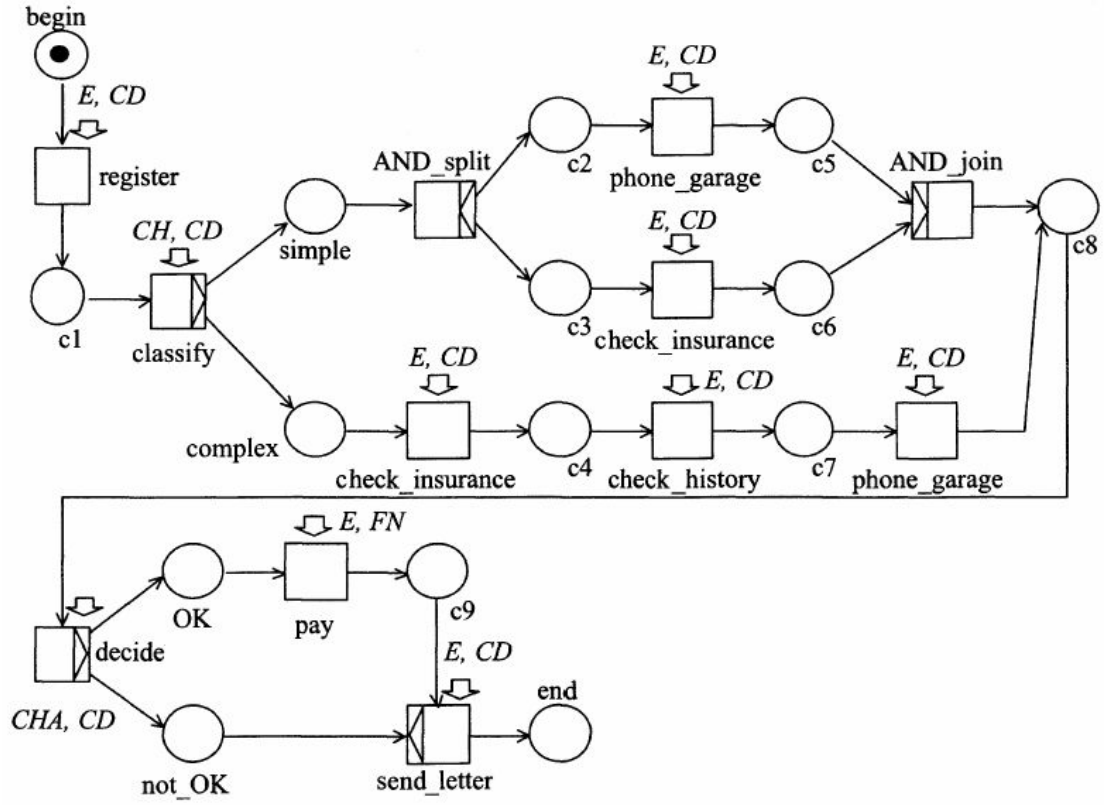


图 S3.2 模型保险公司中的资源分类

**习题 3.2 投诉处理**

可以识别下列角色：

雇员　　（E）
投诉管理员　（CM）
可以识别下列组织单元：
部门 C　　（DC）
后勤部门　　（LD）
结果的模型如图 S3.3 所示：

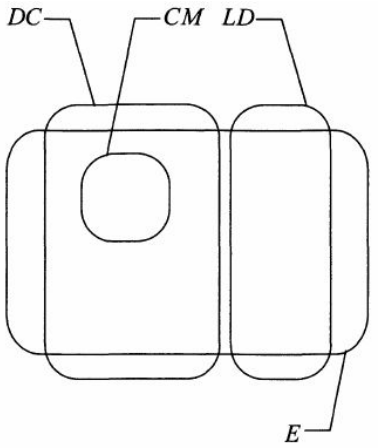


图 S3.3 资源分类投诉处理

这里我们（也）假设投诉管理员是一名雇员，这意味这他也能够处理一个雇员的工作。如果我们把资源类别与过程模型合并，我们可以获得图 S3.4 所示的模型：

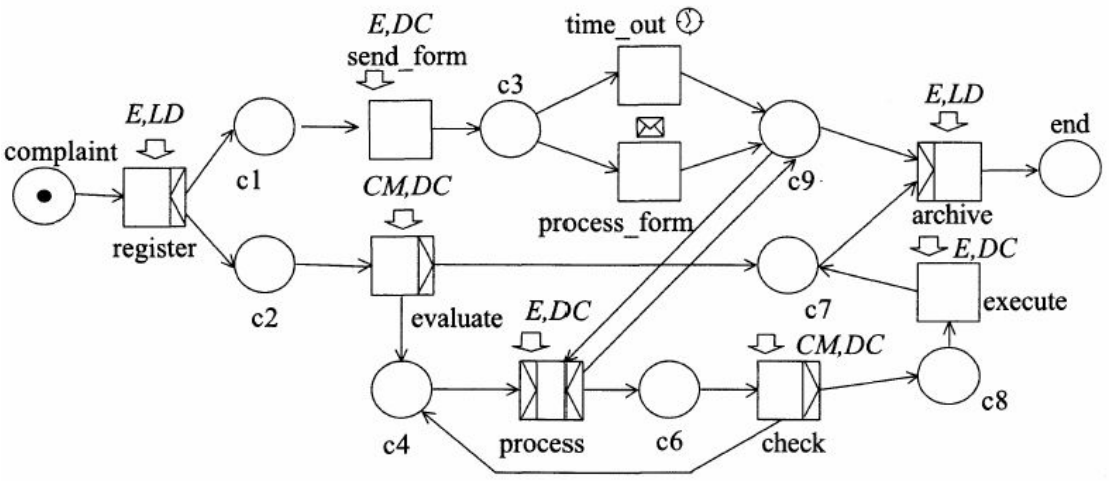


图 S3.4 模型投诉处理中的资源分类

**习题 3.3 职业介绍所**

可以识别下列角色：
公共关系　　(PR)
业务关系　　(BR)
招募　　(RC)
管理员　　(MA)
IT 专家　　(IT)
识别下列组织单元：
工作部　　(JS)

埃因霍温　　(EH)

莱瓦顿　　(LW)

结果的模型如图 S3.5 所示：

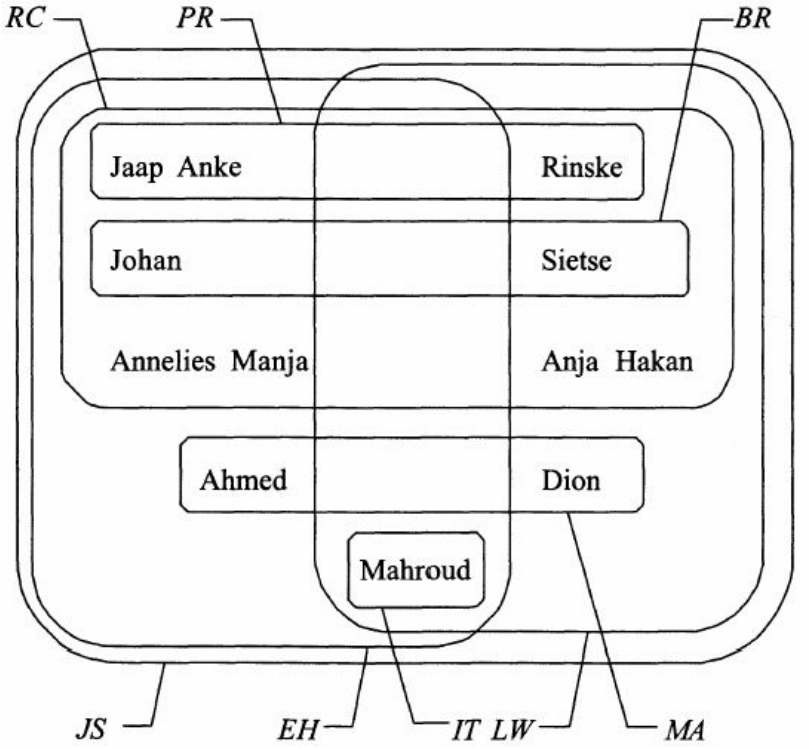


图 S3.5 资源分类职业介绍所

图 S3.6 显示了过程模型，添加正确的触发是重要的，例如为任务 *stop_processing*，添加时间触发对于保持流移动是重要的，防止案例永远呆在库所 *wait* 当中。

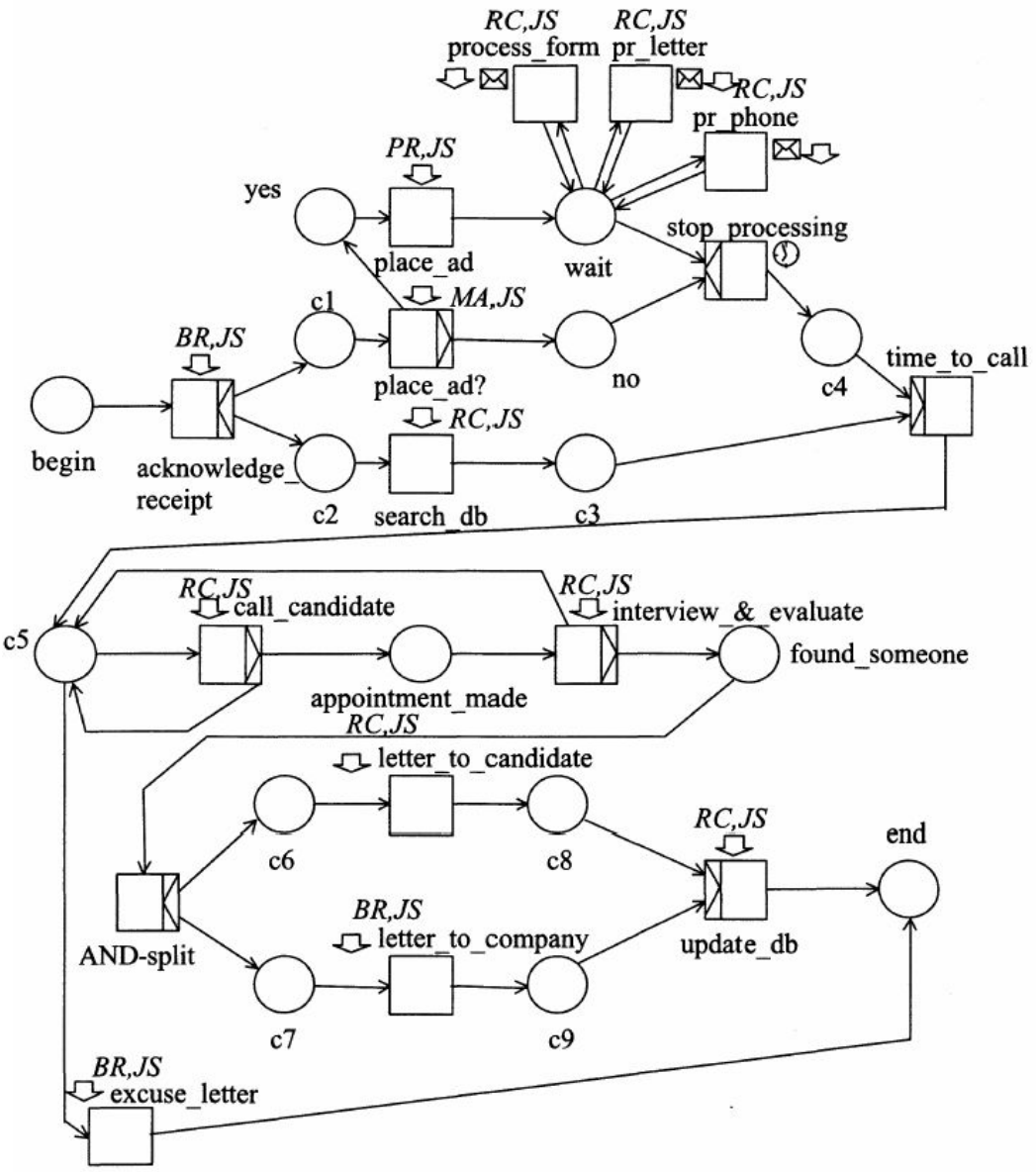


图 S3.6 职业介绍所过程

**习题 3.4 同 CRASH 的一次愉快飞行**

(a) 识别如下角色：

| | |
|---|---|
| 装卸长 | (LM) |
| 航海家 | (NV) |
| 船长 | (CP) |
| Meteo | (MT) |
| 主管 | (DR) |
| 后勤 | (LG) |
| 秘书 | (SE) |
| 从仆 | (CO) |

识别如下组织单元：

| | |
|---|---|
| AIR | (AR) |
| KLM | (KL) |

Support (SP)
CRASH (CR)
结果模型显示在图 S3.7 中。

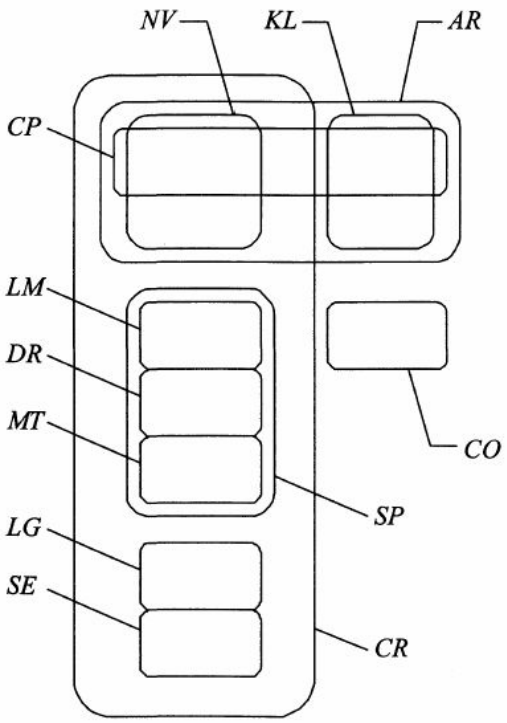


图 S3.7 资源 CRASH

(b) 过程是直接的；简单的应用基本的路由构造。任务 *discuss* 需要两个资源：一个航海家和一个装卸长。于是两个角色附在该任务上：NV 和 LM（见图 S3.8）。由于它们都是 CR 的成员，我们使用 NV/LM、CR 符号。将它们看作一个不同组织单元的独立成员，并且使用符号 NV、AR/LM、CR。这个概念在其它任务中也被使用，这些任务同样需要两个资源。注意当前这一代工作流管理系统不支持多个资源在同一个工作项上工作。因此，我们要尽可能地避免多个资源的任务。

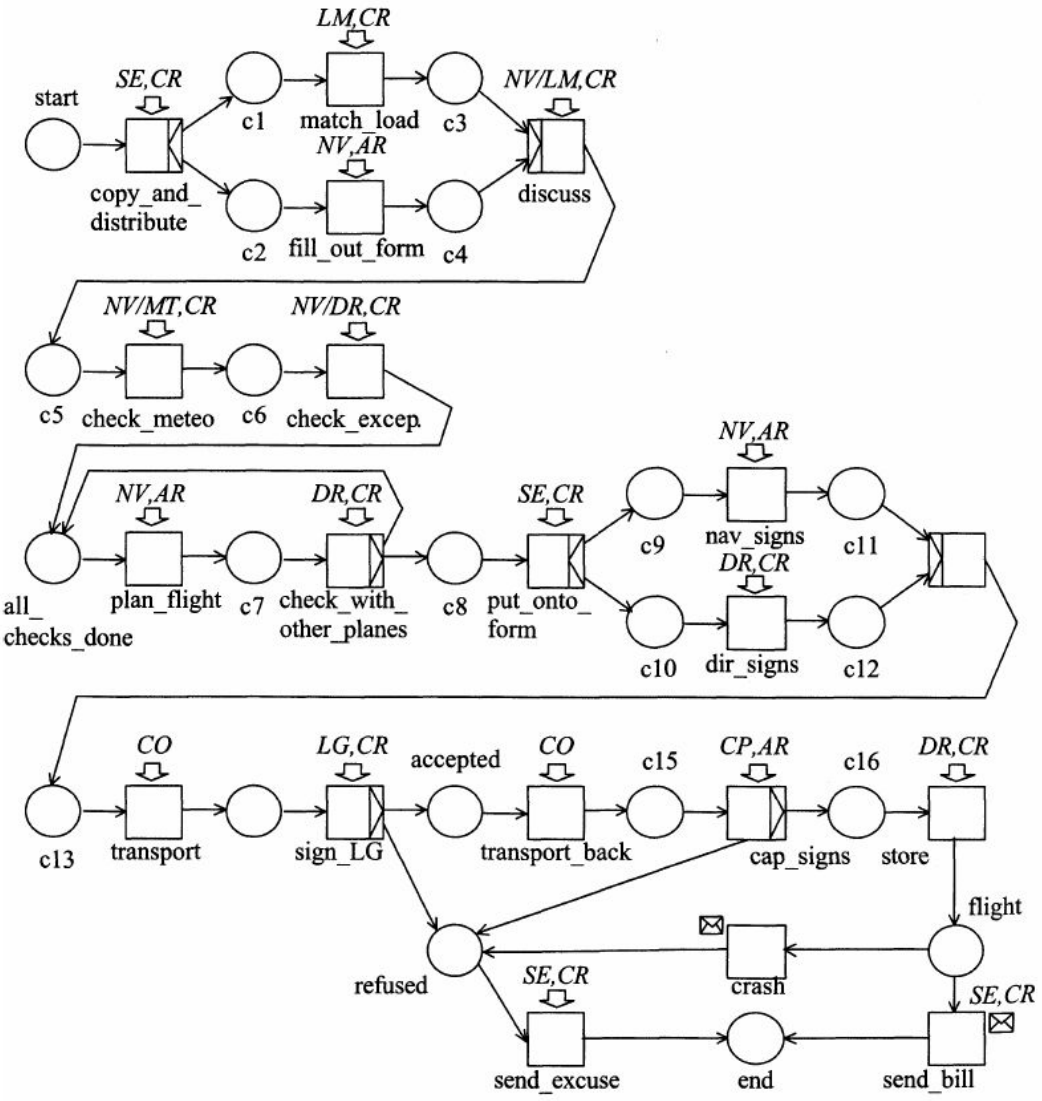


图 S3.8 过程 CRASH

(c) 可能的改进：电子文档（工作流系统）的引入能够改进生产时间。几个任务变得多余（如 *copy_and_distribute* 和 *put_onto_form*）而且并行的数量可以提高。此外，任务 *sign_LG* 和 *cap_signs* 应该尽可能早地执行，以便避免不必要的工作。

## 第四章习题答案

### 习题 4.1 优化数据用法

(a) 见图 S4.1。

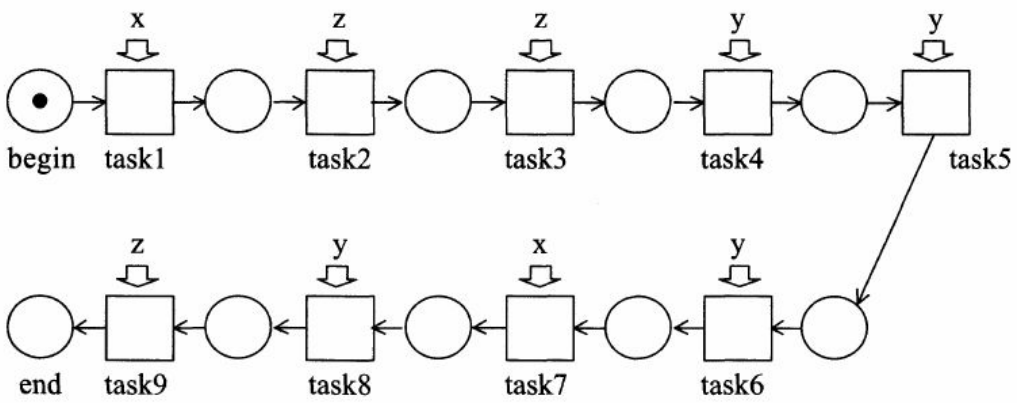


图 S4.1 顺序过程

(b) 不，表示不同形式的路由（如选择和并行路由）是不可能的。

(c) 在图 S4.2 中，我们看到所有的优先关系。在图 S4.3 中我们跳过可以被派生的关系，即如果 *task1* 必须在 *task2* 和 *task7* 之前执行并且 *task2* 也必须在 *task7* 之前执行，*task1* 和 *task7* 之间的关系可以被派生，因此可以被省略。这将得到图 S4.4 所示的 Petri 网。

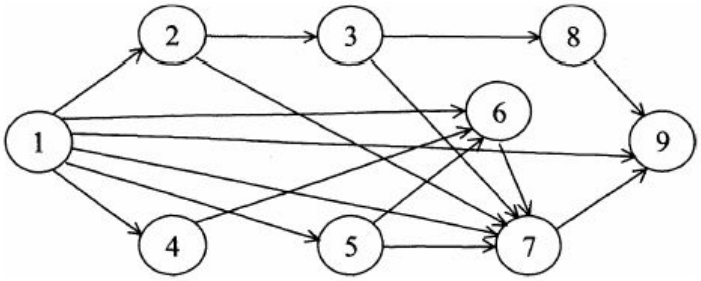


图 S4.2 完整过程

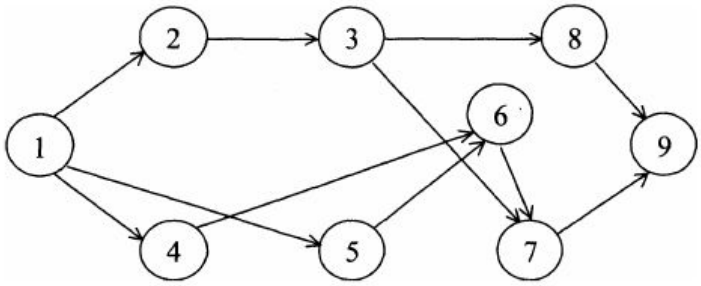


图 S4.3 清除后的过程

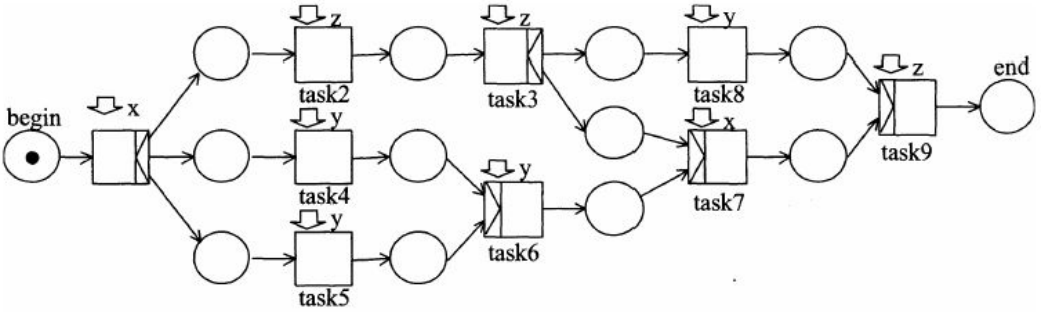


图 S4.4 Petri 网

(d) 是，任务 2 和 3 以及任务 4、5 和 6 被一种类型的资源执行，而且可以被聚类。因此它们可以合并成一个任务。

**习题 4.2 不变量**

(i) 第一个 Petri 网（图 4.38）

(a) *w_rest+type_mail* (=1)
*r_rest+read_mail* (=1)

(b) *begin+send_mail+receive_mail+read*

(c) 不，在库所 mailbox 中可能有任意多的标记

(d) 是

(e) 是

(f) {*w_rest*，*type_mail*，*begin*，*send_mail*}，{*read_mail*，*r_rest*，*receive_mail*，*read*}

(ii) 第二个 Petri 网（图 4.39）

(a) *c1+c2* (=1)
*c3+c4* (=1)

(b) *a+b+c+d*

(c) 是

(d) 是

(e) 否

(f) {*c1*，*c2*，*a*，*b*，*c*，*d*}，{*c3*，*c4*，*a*，*b*，*c*，*d*}

(iii) 第三个 Petri 网（图 4.40）

(a) *c1+c4* (=1)
*c2+c3* (=1)
*c3+c6* (=1)

(b) *g*
*a+b*
*c+d*
*e+f*

(c) 是

(d) 是

(e) 否

(f) {*c1*，*c4*，*a*，*b*，*g*，*e*}，{*c2*，*c5*，*a*，*c*，*d*，*g*}，{*c3*，*c6*，*e*，*f*，*c*，*g*}

(iv) 第四个 Petri 网（图 4.41）

(a) *start+c1+c2+c3+c4+end* (=1)
*start+order_a+c5+c7+c9+c11+c13+invoice+c4+end* (=1)
*start+order_a+c6+c8+notification+c2+c3+c4+end* (=1)
*c5+c7-c6-c8* (=0)
*c9+c11-c10-c12* (=0)
等等

(b) *produce_b+check_b+NOK_b*
*produce_c+check_c+NOK_c*

(c) 是

(d) 否

(e) 是

(f) 无

**习题 4.3 验证过程定义**

(a) 见图 S4.5。

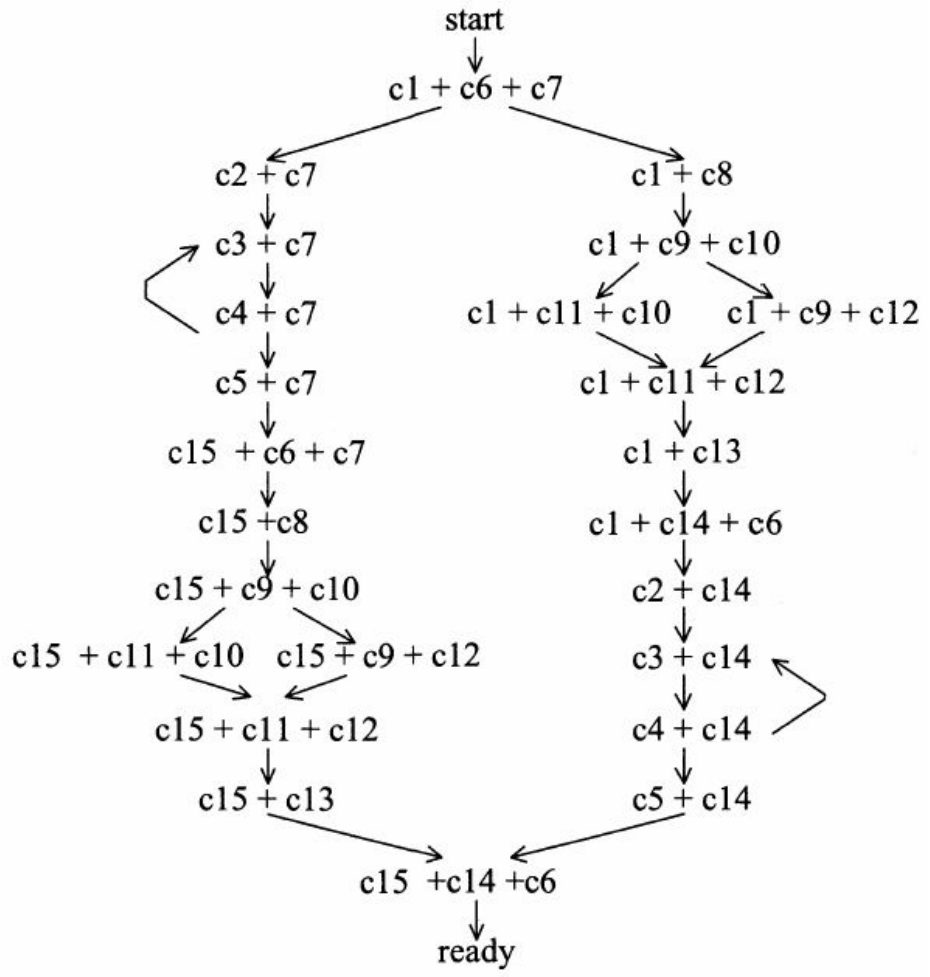


图 S4.5 可达图

(b) 1+(6*8)+1=50。注意原始过程只有 26 个状态。

(c) *start+c6+c2+c3+c4+c5+c8*+1/2(*c9+c10+c11+c12*)+*c13+ready*

(d) *start+c1+c2+c3+c4+c5+c15+ready*

*start+c7+c8+c9+c11+c13+c14+ready*

*start+c7+c8+c9+c10+c12+c14+ready*

*start*+1/2(*c1+c2+c3+c4+c5+c15*)+1/2(*c7+c8*+1/2(*c9+c10+c11+c12*)+*c13+c14*)+*ready*

**习题 4.4 查错**

(i) 如果一个表单被处理而且 *evaluate* 为 *c7* 产生一个标记，一个标记会逗留在 *c9* 中。当一个 *time_out* 发生而且 *evaluate* 为 *c4* 产生一个标记时，过程在标记 *c8* 和 *c4* 的状态处发生死锁。

(ii) 因为 *c9* 以空库所开始并且一直为空，当把标记放入 *c1* 和 *c2* 时过程不能继续。

(iii) 如果过程的左上角部分在过程下面部分的一个标记到达 *c4* 之前到达 *c8*，*process* 不能够实施而且过程在标记 *c8* 和 *c4* 的状态处发生死锁。

**习题 4.5 性能分析 I**

我们使用下列公式：

$$L=\frac{\rho}{1-\rho},\quad S=\frac{1}{\mu-\lambda},\quad W=\frac{\rho}{\mu-\lambda}$$

(a) 任务 1：　　任务 2：

λ=20　　λ=20

μ=60/2=30　　μ=60/2.5=24

ρ ＝0.67　　ρ ＝0.83

L=2　　L=5

S=0.1(6 分钟)　　S=.025（15 分钟）

W=0.066(4 分钟)　　W=0.208（12.5 分钟）

总和：

$L^T=7W^T=0.274$（16.5 分钟）

$S^T=0.35$(21 分钟)

(b) 任务 1：　　任务 2：

λ=20　　λ=5

μ=60/2=30　　μ=10

ρ ＝0.67　　ρ ＝0.5

L=2　　L=1

S=0.1(6 分钟)　　S=.0.2（12 分钟）

W=0.066(4 分钟)　　W=0.1（6 分钟）

总和：

$L^T=3$

$S^T=0.1+1/4*0.2=0.15$（ 9 分钟）△=-12 分钟，即比以前少 12 分钟。

**习题 4.6 性能分析Ⅱ**

(a) 任务 1a：

λ=10　ρ=0.833　S=0.5

μ=12　L=5　W=0.04167

任务 1b：

λ=10　ρ=0.33　S=0.05

μ=30　L=0.5　W=0.0166

任务 2：

λ=20　ρ=0.66　S=0.1

μ=30　L=2　W=0.066

总共：

$L^T=5+0.5+2=7.5$

$S^T=1/2*0.5+1/2*0.05+0.100=0.375$（22.5 分钟）

(b) 任务 1：

λ=20　ρ=0.66　S=0.100

μ=30　L=2　W=0.066

任务 2：

λ=20　ρ=0.66　S=0.100

μ=30　L=2　W=0.066

总和：

$L^T=2+2=4$

$S^T=0.1+0.1=0.2$(12 分钟) △=-10.5 分钟

**习题 4.7 性能分析Ⅲ**

(a) ct1：(%=1.0)

λ=10　　ρ=0.8333　　S=0.5

μ=12　　L=5　　W=0.04167

ct2：(%=0.8)

λ=8　　ρ=0.5333　　S=0.143

μ=15　　L=1.14　　W=0.076

bt：(%=0.56)

λ=5.6　　ρ=0.28　　S=0.0694

μ=20　　L=0.389　　W=0.0194

总和：

$L^T$=6.53

$S^T$=1*0.5+0.8*0.143+0.56*0.0694=0.5+0.114+0.0389=0.65(39.2 分钟)

(b) 选择 1：

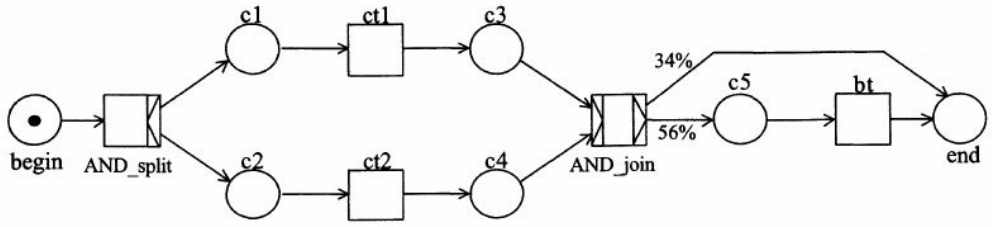


图 S4.6 选择 1

可以通过并行执行事情来减少流动时间。

ct1：(%=1.0)

λ=10　　ρ=0.833　　S=0.5

μ=12　　L=5

ct2：(%=1.0)

λ=10　　ρ=0.67　　S=0.2

μ=15　　L=2

bt：(%=0.56)

λ=5.6　　ρ=0.28　　S=0.07

μ=20　　L=0.389

*ct1* 是并行过程的瓶颈。

总和：

$L^T$>5.39

最大吞吐量= $\lambda$ *(1/ $\rho_{bottleneck}$)=10*(1/0.833)=12

选择 2：

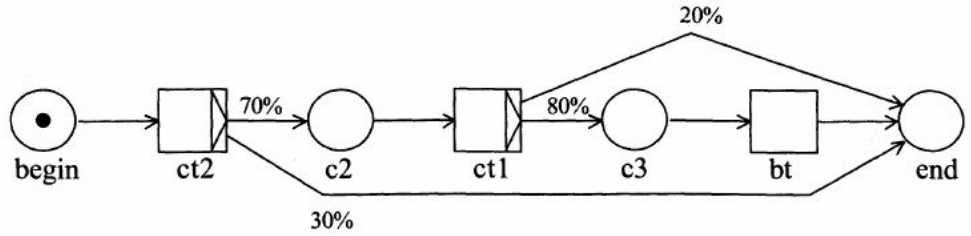


图 S4.7 选择 2

在这种情况下，更多的标记会直接结束以便可以使用较少的资源。

ct1：(%=0.7)

λ=7　　ρ=0.583

μ=12　　L=1.4

ct2：(%=1.0)

λ=10　　ρ=0.67

μ=15　　L=2

bt：(%=0.56)

λ=5.6　　ρ=0.28

μ=20　　L=0.389

*ct2* 现在已经成为瓶颈而且在系统中存在较少的案例。

总和：

$L^T=3.79$

最大的吞吐量= $\lambda*(1/\rho_{bottleneck})=10*(1/0.67)=15$

其它选择：

- 将 *ct1* 和 *ct2* 合并为一个任务来节省预热时间
- 制作一个对所有任务可用的资源池

**习题 4.8 电子商务**

(a)

| 步骤 | 任务集合 | 选中的任务 | 使用的块 | 新任务 |
|---|---|---|---|---|
| 1 | a | a | 顺序 | b |
| 2 | a,b | b | 顺序 | c |
| 3 | a,b,c | b | 迭代 | d |

客户端工作流

| 步骤 | 任务集合 | 选中的任务 | 使用的块 | 新任务 |
|---|---|---|---|---|
| 1 | e | e | 顺序 | f |

服务器工作流

(b)

| 步骤 | 任务集合 | 选中的任务 | 使用的块 | 新任务 |
|---|---|---|---|---|
| 1 | a | a | 顺序 | b |
| 2 | a,b | b | 顺序 | c |
| 3 | a,b,c | b | 迭代 | d |
| 4 | a,b,c,d | b | 与 | e |
| 5 | a,b,c,d,e | e | 顺序 | f |

耦合的工作流

耦合的工作流的步骤 1、2 和 3 与客户工作流相同，步骤 4 是新的而且步骤 5 是服务器工作流的步骤 1。

(c) 不，这样一个派生是不可能的。为了验证这一点，注意(p，q)、(t，v)和(r，s)组成了与分裂和与合并对，于是这些中的每一个必须由一个与块的替换完成。然而，它们会被嵌套（一个被放入另一当中）或脱节。这不是事实，实际上它们有下列序列：p、t、q、r、v、s，于是它们彼此交叉。

(d) 是，它是一个可靠且安全的工作流。为明白这一点，注意没有消息交换，即没有 q、t、r 和 s，我们拥有一个可靠而安全的工作流（见练习 1）。因为 b 和 d 会实施，我们看到 t 和 q 也会实施而且 c 和 e 也会实施。类似的 r 和 v 会实施。于是 f 和 s 会实施。没有标记留下，因此网是可靠的。网是安全的，是没有消息交换的网是安全的这个事实的一个

直接结果。

## 第五章习题答案

### 习题 5.1

图 S5.1 给出了工作流管理联盟的参考模型的一个图形表示。组件和接口的细节描述见第 5 章。

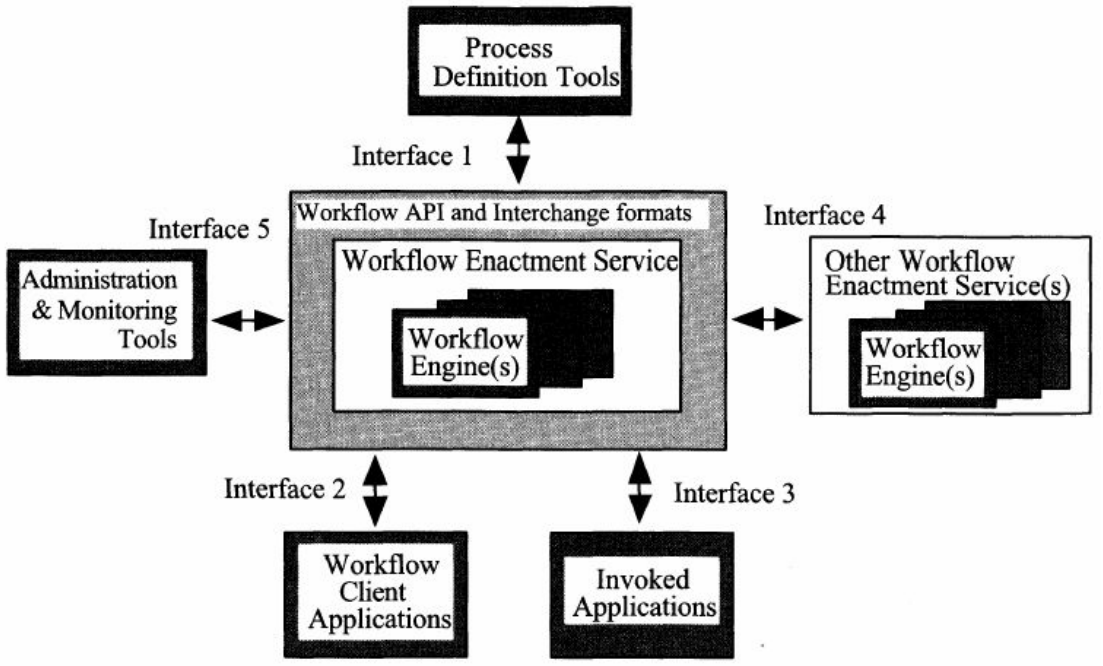


图 S5.1 工作流管理联盟的参考模型(©WfMC)

### 习题 5.2

简短问题的答案：

(a) 原子性、一致性、隔离性和持久性

(b) 接口 3：工作流管理系统和应用是非同步的

(c) 工作流设计师、管理员、过程分析家和雇员

(d) 互操作：WFMC 的定义、SWAP、WF-XML 和 OMG 的 jointFlow

(e) Staffware：市场领导者的定位是产品工作流。COSA：基于 Petri 网的工作流管理系统，属于产品工作流。ActionWorkflow：一个侧重于协作和谈判的系统，而不是路由，非常不同于典型的产品系统。

(f) Woflan：使用非精确的分析技术进行验证，即定性分析。ExSpect：基于 Petri 网的仿真工具。这两个工具可以同几个工作流产品混合使用。

(g) Protos（Pallas Athena BV，Plasmolen，荷兰），ARIS（IDS Sheer AG，Saarbriicken，德国），BusinessSpecs（IvyTeam，Zug，瑞士），Income（Promatis AG，Karlsbad，德国）和 Meta Workflow Analyzer（Meta Software，Cambridge，MA，美国）。

### 习题 5.3

COSA 基于 Petri 网，然而存在一种一对一的转换而且我们没有显示使用 CONE 的过程。过程到 Staffware 的转换更加棘手，图 S5.2 显示了对应的在 Staffware 的 GWD 中的工作流过程定义。在给定了构造块的描述的情况下，模型是直接的。

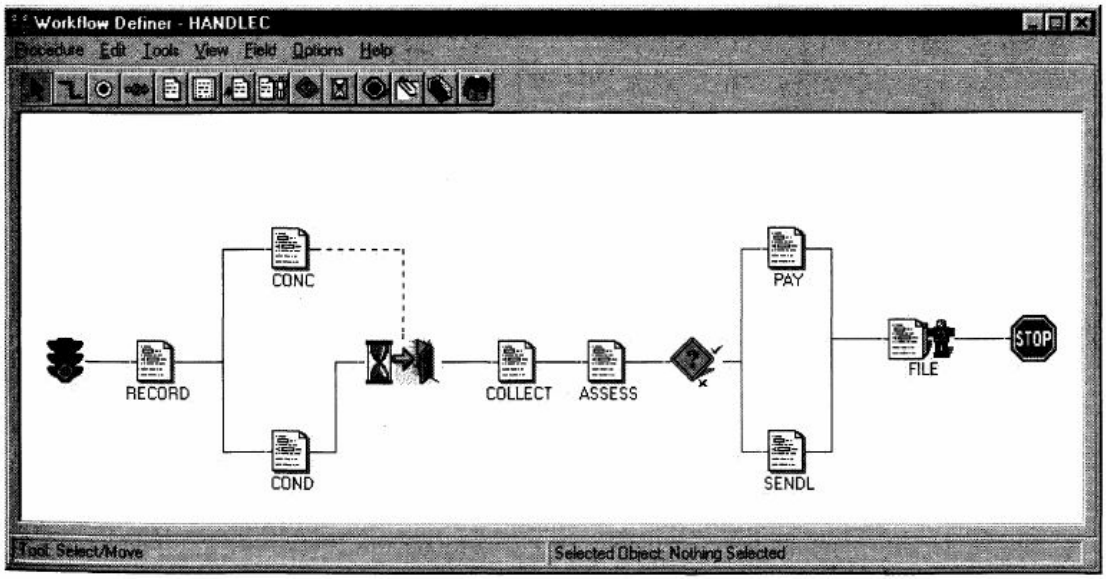


图 S5.2 用 Staffware 的 GWD 建模的过程“处理投诉”

**习题 5.4**

在第 2 章显示的模型与 COSA 之间存在一对一的转换，过程到 Staffware 的转换更加棘手。图 S5.3 显示了 Staffware 的 GWD 的对应工作流过程定义。给定了构造块的描述，模型的第一部分是直接的。对模型来说较容易的唯一的事情是 *cancel* 任务。一般来讲，用 Staffware 建模非自由选择构造是非常难的。这种情况下我们能够使用一个简单的窍门来进行建模：带有一个超时的两个 *cancel* 步骤。为了简化，我们不建模触发并且简化了两种类型保险的选择。

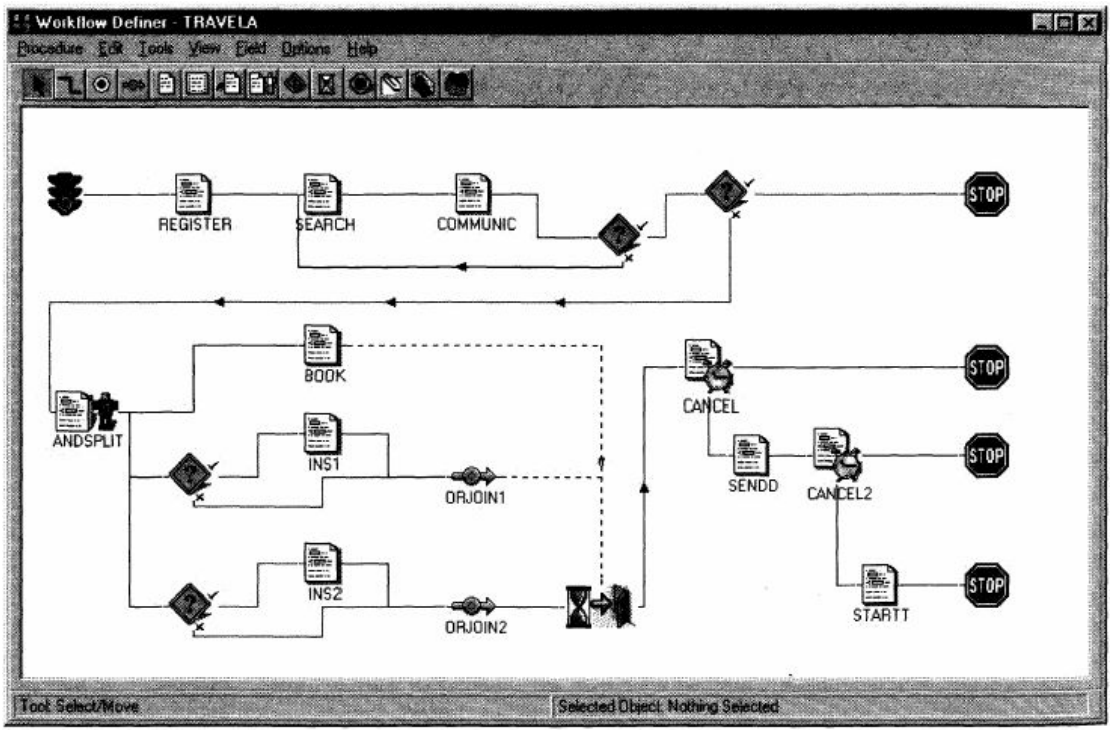


图 S5.3 用 Staffware 的 GWD 建模的过程“旅行社”

## 第六章习题答案

**习题 6.1**

(a) 首先，包含（可能的）用户是非常重要的，因为他们具有现有的过程和系统的大量知识，经常他们有改进的好主意，因此他们的知识和创造力对于重新设计组来说具有很大的价值。

第二，他们的介入对于获得组织中的承诺是重要的。积极参与到新过程和系统的设计的人有它是他们的孩子的感觉，所以他们总是千方百计的保护新过程和系统，特别是他们的同事，因此他们成为改变过程中的关键人物。过于频繁的改变操作会受到与会员工的强烈反抗，改变是一个非常情绪化的过程。

(b) 选择具有以下特性的人员是非常重要的：

- 受同事尊敬
- 过程和系统的知识渊博
- 思想开明，即具有创新思维能力

**习题 6.2**

在诊断阶段，业务案例用来确定实际情形中的关键性能指标（KPI）的值：指零测量。有时用一个例子解释事情比明确地表达例子所属的规则容易。业务案例可以被看作例子而过程则被看作规则，因此对于用户来说根据业务案例思考比更加抽象的过程容易。它们被应用的下一个阶段是过程再设计阶段，即模拟实验和游戏。它们还可以用于需求定义阶段。最后业务案例还用在系统被测试的集成阶段和执行接受测试的交付阶段。因此，仔细地维护一组业务案例是重要的，这样考虑一个改进时，它们还可用于监督和改进阶段。

**习题 6.3**

合并这些阶段的好处如下。与组件结构一起定义概念数据模型和功能模型是有好处的，因为组件之上功能的分布可以用迭代的方式生成。如果需求模型和体系结构被划分为两个阶段，迭代则更加困难。在一个阶段考虑功能和技术细节是有优势的，因为这样避免了技术上不可行的需求。

也存在不足。在需求阶段用户可以做出重要的贡献，然而他们在技术体系的定义中并非那么有用，因此分裂这些阶段是自然的。另一个不足是对“关注分离原则”的违背，该原则说同一时刻关注一个方面比较好，即功能和技术细节应该在单独的阶段考虑。

## 第七章习题答案

**习题 7.1**

我们仅为问题 7.1(b)提供解决方案。图 S7.1 显示了当前情形的过程模型，我们没有建模资源触发：多数资源需要一个资源触发。

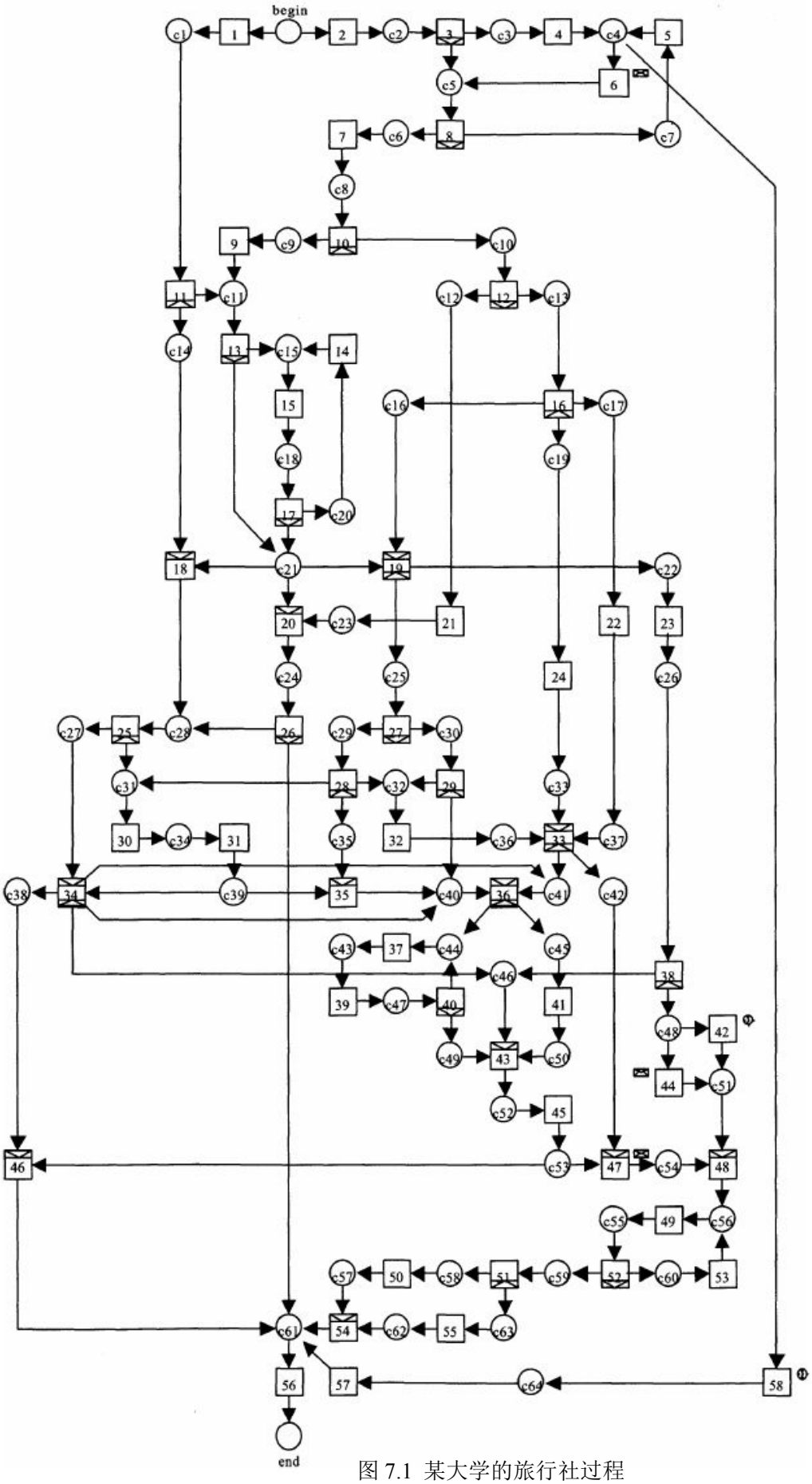


图 7.1 某大学的旅行社过程

**任务**

1. 注册个人顾客
2. 注册公务顾客
3. 检查许可
4. 供给空白许可
5. 返回不正确的许可
6. 接受填好的许可
7. 归档正确的许可
8. 检查正确的许可
9. 开始公务旅行
10. 发送副本给财务科
11. 开始私人旅行
12. 检查容许
13. 准备提议
14. 准备新提议
15. 召唤顾客进行确认
16. 发送肯定便签
17. 检查确认的提议
18. 检查私人旅行
19. 决定航班成本
20. 召唤顾客
21. 发送否定便签
22. 进行预付款
23. 准备现金和支票
24. 支付注册费用
25. AND-split
26. 检查决策
27. 检查航班付款
28. AND-split
29. AND-split
30. 进行会面
31. 顾客付款
32. 发送业务航班费用给财务科
33. 为航班付款
34. 检查私人旅行付款
35. 检查公务旅行付款
36. 检查所有付款
37. 预约旅店
38. 发送现金和支票
39. 打印凭证
40. 检查所有预约
41. 打印机票
42. 未返还
43. 制作方便文件夹

44. 顾客返回现金或支票
45. 捡起
46. 结束私人旅行
47. 接收申报
48. 处理申报
49. 计算余额
50. 扣除预算
51. AND-split
52. 检查确认
53. 纠错
54. 结束公务旅行
55. 整理余额
56. 关闭归档
57. 发送通知
58. 收到未填写的许可

**条件**

c1. 注册的个人旅行
c2. 注册的公务旅行
c3. 没有许可
c4. 顾客填写许可
c5. 填写的许可
c6. 正确的许可
c7. 不正确的许可
c8. 归档的许可
c9. 发送的副本
c10. 发送的归档副本
c11. 开始旅行组织
c12. 不允许
c13. 允许
c14. 正在组织个人旅行
c15. 提议
c16. 肯定的便签
c17. 预付款
c18. 顾客（不）同意
c19. 注册费
c20. 没有计划
c21. 计划
c22. 请求现金和支票
c23. 否定的便签
c24. 确定的顾客
c25. 知道的航班费用
c26. 现金和支票
c27. 正在为个人旅行付款
c28. 正在为个人旅行和要支付的个人航班付款

c29. 一些个人航班
c30. 所有的公务航班
c31. 要付款的个人航班
c32. 要付款的公务航班
c33. 费用完毕
c34. 要支付的顾客
c35. 正在为公务旅行付款
c36. 详细的报价信息
c37. 完成的付款
c38. 支付的个人旅行
c39. 付款的顾客
c40. 付款的个人航班
c41. 付款的公务航班
c42. 等待申报
c43. 预约的旅馆
c44. 预约旅馆
c45. 打印机票
c46. 发送的现金和支票
c47. 打印的凭证
c48. 等待返还
c49. 安排的运输
c50. 打印的机票
c51. 数量信息
c52. 准备好的文件夹
c53. 捡起的
c54. 收到的申报
c55. 余额
c56. 正在处理申报
c57. 扣除了
c58. 扣除的数量
c59. 确认的余额
c60. 未确认的余额
c61. 要关闭的文档
c62. 解决的余额
c63. 确认的余额
c64. 不能保证旅行