云南大学软件学院 实验报告

课程： 软件工程概论 学期： 2024 - 2025学年 第一学期 任课教师： 莫启

专业： 软件工程 学号： 20221120044 姓名： 唐嘉骏 成绩：

**实验3**

（1）利用熟悉的编程语言（如Python）实现多个标号迁移系统LTS的同步+异步的并行组合。**注意：下面给出了用Python定义的一个LTS**；

# 1.定义LTS----------------------------------------------

**class** LTS(object):

**def** \_\_init\_\_(self, start, ends, states, trans):

self.start = start #一个开始状态

self.ends = ends #多个终止状态

self.states = states #所有状态集合

self.trans = trans #状态到状态的迁移，形如(s,a,s')

# 2.定义迁移---------------------------------------------

**class** Tran(object):

**def** \_\_init\_\_(self, state\_from, label, state\_to):

self.state\_from = state\_from

self.label = label

self.state\_to = state\_to

使用顾客，银行，ATM机之间的交互作为例子，顾客和银行是同步的，顾客和ATM机是异步的。

代码：

1. **class** Tran:
2. **def** \_\_init\_\_(self, state\_from, label, state\_to):
3. self.state\_from = state\_from
4. self.label = label
5. self.state\_to = state\_to
7. **def** \_\_repr\_\_(self):
8. **return** f"{self.state\_from} --{self.label}--> {self.state\_to}"

11. **class** LTS:
12. **def** \_\_init\_\_(self, start, ends, states, trans):
13. self.start = start
14. self.ends = ends
15. self.states = states
16. self.trans = trans
18. **def** get\_transitions(self, state):
19. **return** [tran **for** tran **in** self.trans **if** tran.state\_from == state]
21. **def** \_\_repr\_\_(self):
22. states\_str = "\n  ".join(str(state) **for** state **in** self.states)
23. transitions\_str = "\n  ".join(str(tran) **for** tran **in** self.trans)
25. **return** f"Start State: {self.start}\nEnd States: {self.ends}\nStates:\n  {states\_str}\nTransitions:\n  {transitions\_str}"

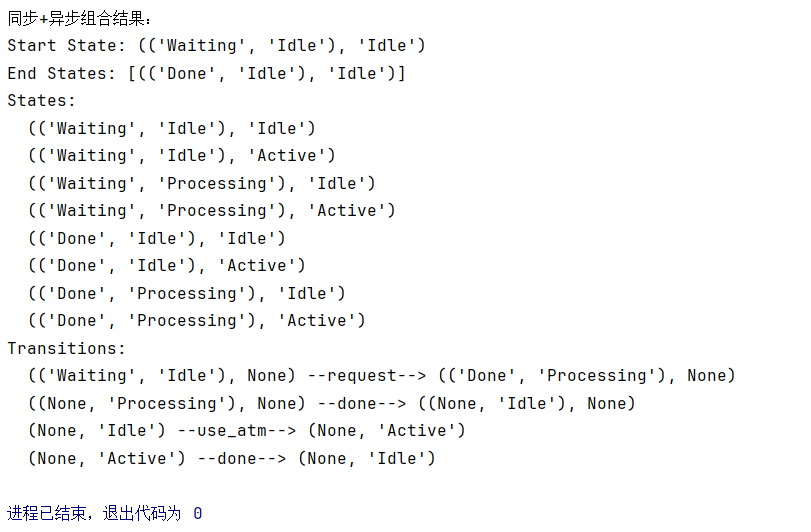
28. # 同步组合
29. **def** sync\_combine(lts1, lts2):
30. sync\_start = (lts1.start, lts2.start)
31. sync\_ends = [(s1, s2) **for** s1 **in** lts1.ends **for** s2 **in** lts2.ends]
32. sync\_states = [(s1, s2) **for** s1 **in** lts1.states **for** s2 **in** lts2.states]
34. sync\_trans = []
35. label = set()
36. **for** tran1 **in** lts1.trans:
37. **for** tran2 **in** lts2.trans:
38. **if** tran1.label == tran2.label:
39. sync\_trans.append(
40. Tran((tran1.state\_from, tran2.state\_from), tran1.label, (tran1.state\_to, tran2.state\_to)))
41. label.add(tran1.label)
43. # 处理其他标签的迁移
44. **for** tran1 **in** lts1.trans:
45. **if** tran1.label **not** **in** label:
46. sync\_trans.append(Tran((tran1.state\_from, None), tran1.label, (tran1.state\_to, None)))
47. **for** tran2 **in** lts2.trans:
48. **if** tran2.label **not** **in** label:
49. sync\_trans.append(Tran((None, tran2.state\_from), tran2.label, (None, tran2.state\_to)))
51. **return** LTS(sync\_start, sync\_ends, sync\_states, sync\_trans)

54. # 异步组合
55. **def** async\_combine(lts1, lts2):
56. async\_start = (lts1.start, lts2.start)
57. async\_ends = [(s1, s2) **for** s1 **in** lts1.ends **for** s2 **in** lts2.ends]
58. async\_states = [(s1, s2) **for** s1 **in** lts1.states **for** s2 **in** lts2.states]
60. async\_trans = []
61. **for** tran1 **in** lts1.trans:
62. async\_trans.append(Tran((tran1.state\_from, None), tran1.label, (tran1.state\_to, None)))
63. **for** tran2 **in** lts2.trans:
64. async\_trans.append(Tran((None, tran2.state\_from), tran2.label, (None, tran2.state\_to)))
66. **return** LTS(async\_start, async\_ends, async\_states, async\_trans)

69. # 同步+异步组合
70. **def** sync\_async\_combine(lts\_list, sync\_indices=None):
71. **if** sync\_indices **is** None:
72. sync\_indices = []
74. combined\_lts = lts\_list[0]
76. **for** i **in** range(1, len(lts\_list)):
77. **if** i **in** sync\_indices:
78. combined\_lts = sync\_combine(combined\_lts, lts\_list[i])
79. **else**:
80. combined\_lts = async\_combine(combined\_lts, lts\_list[i])
82. **return** combined\_lts

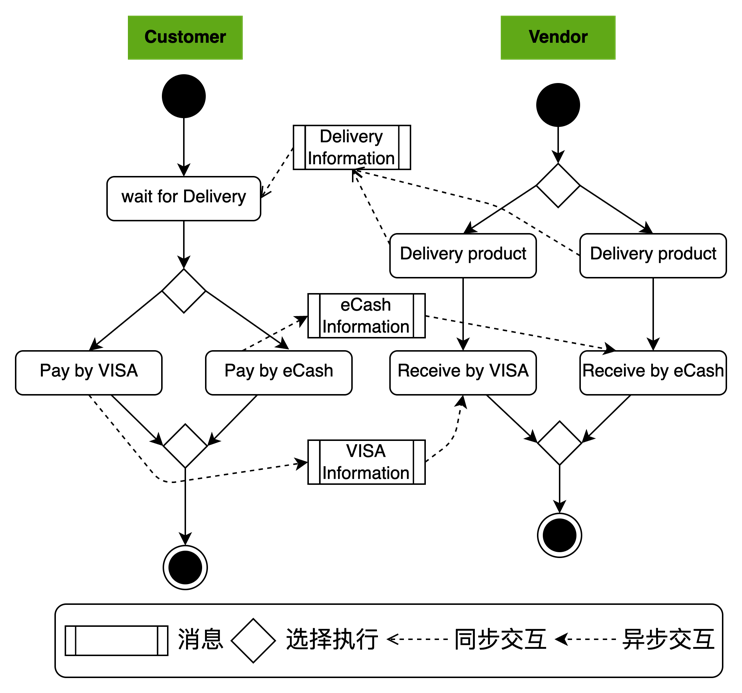
85. **if** \_\_name\_\_ == "\_\_main\_\_":
86. # LTS1: Customer（顾客）
87. trans1 = [Tran('Waiting', 'request', 'Done')]
88. lts1 = LTS('Waiting', ['Done'], ['Waiting', 'Done'], trans1)
90. # LTS2: Bank（银行）
91. trans2 = [Tran('Idle', 'request', 'Processing'), Tran('Processing', 'done', 'Idle')]
92. lts2 = LTS('Idle', ['Idle'], ['Idle', 'Processing'], trans2)
94. # LTS3: ATM（自动取款机）
95. trans3 = [Tran('Idle', 'use\_atm', 'Active'), Tran('Active', 'done', 'Idle')]
96. lts3 = LTS('Idle', ['Idle'], ['Idle', 'Active'], trans3)
98. parallel\_lts\_sync\_async = sync\_async\_combine([lts1, lts2, lts3], sync\_indices=[0, 1])
99. **print**("同步+异步组合结果：")
100. **print**(parallel\_lts\_sync\_async)

运行结果：

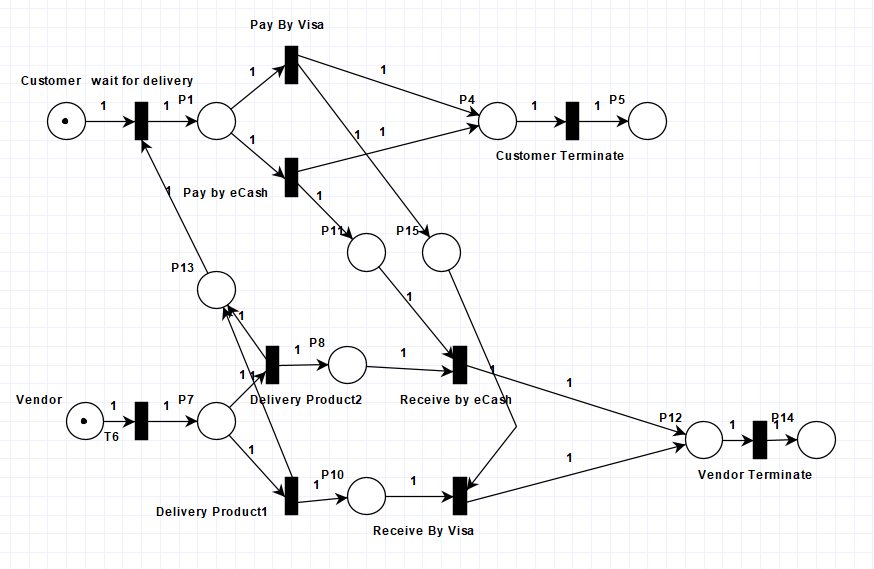


通过输出我们可以看到，顾客和银行通过request进行同步，并变为Done和Processing，然后银行的Processing状态通过内部活动转为Idle。ATM通过异步转为active状态，处理完成业务后变回Idle状态，因为不受顾客同步的限制，所以是None。

（2）在（1）的基础上对如下所示的分布式系统进行建模并进行行为分析以确定其是否在运行的过程中存在异常行为，例如死锁等。



通过对系统进行petri网建模，可以发现，如果Vendor使用的是eCash的Delivery，但是Customer却使用的是Pay By Visa，那么就会导致死锁，同样的，如果Vendor使用的是visa的Delivery，Customer使用Pay By eCash，那么也会导致死锁。



代码：

1. **class** Tran:
2. **def** \_\_init\_\_(self, state\_from, label, state\_to):
3. self.state\_from = state\_from
4. self.label = label
5. self.state\_to = state\_to
7. **def** \_\_repr\_\_(self):
8. **return** f"{self.state\_from} --{self.label}--> {self.state\_to}"

11. **class** LTS:
12. **def** \_\_init\_\_(self, start, ends, states, trans):
13. self.start = start
14. self.ends = ends
15. self.states = states
16. self.trans = trans
18. **def** get\_transitions(self, state):
19. **return** [tran **for** tran **in** self.trans **if** tran.state\_from == state]
21. **def** \_\_repr\_\_(self):
22. states\_str = "\n  ".join(str(state) **for** state **in** self.states)
23. transitions\_str = "\n  ".join(str(tran) **for** tran **in** self.trans)
25. **return** f"Start State: {self.start}\nEnd States: {self.ends}\nStates:\n  {states\_str}\nTransitions:\n  {transitions\_str}"

28. # 同步组合
29. **def** sync\_combine(lts1, lts2):
30. sync\_start = (lts1.start, lts2.start)
31. sync\_ends = [(s1, s2) **for** s1 **in** lts1.ends **for** s2 **in** lts2.ends]
32. sync\_states = [(s1, s2) **for** s1 **in** lts1.states **for** s2 **in** lts2.states]
34. sync\_trans = []
35. label = set()
36. **for** tran1 **in** lts1.trans:
37. **for** tran2 **in** lts2.trans:
38. **if** tran1.label == tran2.label:
39. sync\_trans.append(
40. Tran((tran1.state\_from, tran2.state\_from), tran1.label, (tran1.state\_to, tran2.state\_to)))
41. label.add(tran1.label)
43. # 处理其他标签的迁移
44. **for** tran1 **in** lts1.trans:
45. **if** tran1.label **not** **in** label:
46. sync\_trans.append(Tran((tran1.state\_from, None), tran1.label, (tran1.state\_to, None)))
47. **for** tran2 **in** lts2.trans:
48. **if** tran2.label **not** **in** label:
49. sync\_trans.append(Tran((None, tran2.state\_from), tran2.label, (None, tran2.state\_to)))
51. **return** LTS(sync\_start, sync\_ends, sync\_states, sync\_trans)

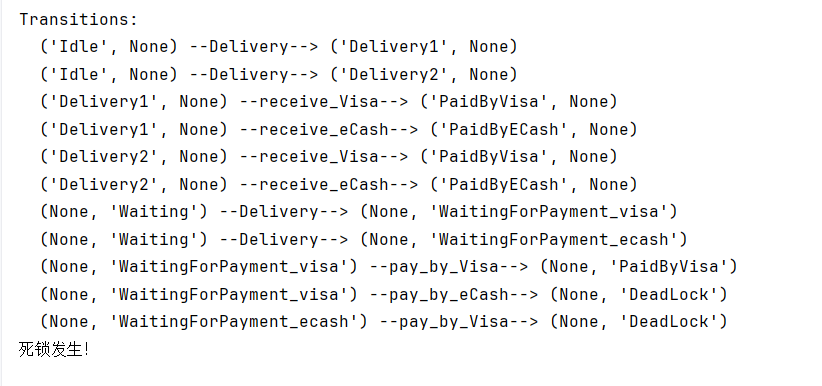
54. # 异步组合
55. **def** async\_combine(lts1, lts2):
56. async\_start = (lts1.start, lts2.start)
57. async\_ends = [(s1, s2) **for** s1 **in** lts1.ends **for** s2 **in** lts2.ends]
58. async\_states = [(s1, s2) **for** s1 **in** lts1.states **for** s2 **in** lts2.states]
60. async\_trans = []
61. **for** tran1 **in** lts1.trans:
62. async\_trans.append(Tran((tran1.state\_from, None), tran1.label, (tran1.state\_to, None)))
63. **for** tran2 **in** lts2.trans:
64. async\_trans.append(Tran((None, tran2.state\_from), tran2.label, (None, tran2.state\_to)))
66. **return** LTS(async\_start, async\_ends, async\_states, async\_trans)

69. # 同步+异步组合
70. **def** sync\_async\_combine(lts\_list, sync\_indices=None):
71. **if** sync\_indices **is** None:
72. sync\_indices = []
74. combined\_lts = lts\_list[0]
76. **for** i **in** range(1, len(lts\_list)):
77. **if** i **in** sync\_indices:
78. combined\_lts = sync\_combine(combined\_lts, lts\_list[i])
79. **else**:
80. combined\_lts = async\_combine(combined\_lts, lts\_list[i])
82. **return** combined\_lts

85. # 死锁检测
86. **def** detect\_deadlock(lts):
87. **for** tran **in** lts.trans:
88. **if** tran.state\_to == (None,'DeadLock'):
89. **return** True
90. **return** False

93. **if** \_\_name\_\_ == "\_\_main\_\_":
94. # Vendor
95. trans1 = [
96. Tran('Idle', 'Delivery', 'Delivery1'),
97. Tran('Idle', 'Delivery', 'Delivery2'),
98. Tran('Delivery1', 'receive\_Visa', 'PaidByVisa'),
99. Tran('Delivery1', 'receive\_eCash', 'PaidByECash'),
100. Tran('Delivery2', 'receive\_Visa', 'PaidByVisa'),
101. Tran('Delivery2', 'receive\_eCash', 'PaidByECash')
102. ]
103. vendor\_lts = LTS('Idle', ['PaidByVisa', 'PaidByECash'],
104. ['Idle', 'Delivery1', 'Delivery2', 'PaidByVisa', 'PaidByECash'], trans1)
106. # Customer
107. trans2 = [
108. Tran('Waiting', 'Delivery', 'WaitingForPayment\_visa'),
109. Tran('Waiting', 'Delivery', 'WaitingForPayment\_ecash'),
110. Tran('WaitingForPayment\_visa', 'pay\_by\_Visa', 'PaidByVisa'),
111. Tran('WaitingForPayment\_visa', 'pay\_by\_eCash', 'DeadLock'),
112. Tran('WaitingForPayment\_ecash', 'pay\_by\_Visa', 'DeadLock')
113. ]
114. customer\_lts = LTS('Waiting', ['PaidByVisa', 'PaidByECash'],
115. ['Waiting', 'WaitingForPayment\_visa', 'WaitingForPayment\_ecash', 'PaidByVisa', 'PaidByECash'], trans2)
117. parallel\_lts = sync\_async\_combine([vendor\_lts, customer\_lts], sync\_indices=[0])
119. **print**("同步+异步组合结果：")
120. **print**(parallel\_lts)
122. **if** detect\_deadlock(parallel\_lts):
123. **print**("死锁发生！")
124. **else**:
125. **print**("没有发生死锁。")

输出结果：



检测到死锁发生。