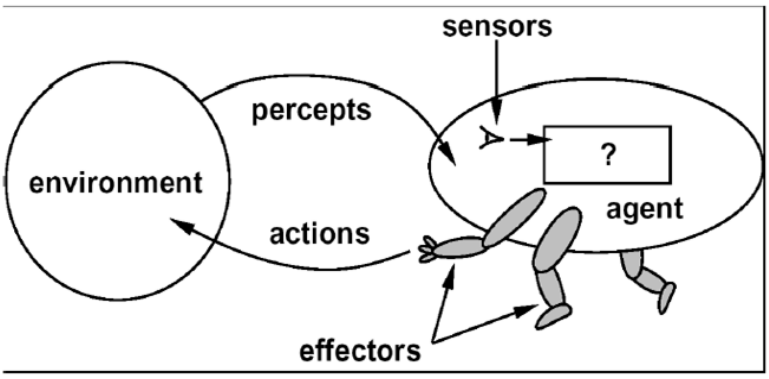
Лабораторийн ажил 2

Агент бол хүрээлэн буй орчныг мэдрэгчээр дамжуулан ойлгож, хариу үйлдэлийн эффекторуудаар дамжуулан тухайн орчинд ажилладаг. Хүний мэдрэгч нь мэдрэх чадвартай нүд, чих, бусад эрхтэнүүдтэй мөн эффект хийх зориулалттай гар, хөл, ам, биеийн бусад хэсгүүдтэй байдаг. Робот агент нь мэдрэгч ба эффектийн янз бүрийн ажиллагааг камер, хэт улаан туяаны хайгч төхөөрөмжүүдээр орлуулдаг. Ухаалаг агент бол зөв зүйлийг хийдэг зүйл юм. Өөрөөр хэлбэл оролтуудын цуглуулгыг гаралт уруу буулгагч функц юм $f : P^* \rightarrow A$. Зураг 1-д үзүүлэв. Энэ нь буруу зүйл хийснээс дээр боловч энэ нь яг юу гэсэн утгатай вэ? Эхний байдлаар зөв үйлдэл нь агентийг хамгийн их амжилтанд хүргэх шалтгаан болно. Энэ нь агентийн амжилтыг хэрхэн, хэзээ үнэлэхийг шийдэх асуудал бидэнд тулгардаг.



Зураг 1. Агентууд мэдрэгч ба эффектороор дамжуулан хүрээлэн буй орчинтой харьцдаг
Хэрэглээнд суурилсан агентууд (Utility-Based Agents) Олон боломжит хувилбар байгаа тохиолдолд аль нь хамгийн сайн болохыг шийдэхийн тулд хэрэглээнд суурилсан агентууд ашиглагдах бөгөөд муж тус бүрийн давуу эрх (хэрэгсэл) дээр үндэслэн үйлдлүүдийг сонгоно. Agent program AI-ийн нь агент функцийг хэрэгжүүлж агент програмыг боловсруулах явдал юм. Агент програм нь агент функцийн хэрэгжилт юм. А агент програм нь физик архитектур дээр f функцийг бий болгодог. Оновчтой агент Ухаалаг (оновчтой) агент бол гүйцэтгэлийн хэмжүүр ашиглан амжилтанд хүрэх хамгийн боломжит үйлдлийг сонгодог агент юм. Автономит систем нь өөрийн туршлагаар автономит зан авирийг тодорхойлдог. Өөрөөр автономит дасан зохицоход хангалттай хугацаа өгч, олон янзын орчинд амжилттай ажиллаж чаддаг болдог. Reflex agent буюу энгийн рефлексийн агентууд нь зөвхөн одоогийн ойлголтын үндсэн дээр ажилладаг. Бусад ойлголтыг үл хамаардаг. Зарим рефлексийн агентууд нь одоогийн төлөв байдлын талаархи мэдээллийг агуулдаг бөгөөд энэ нь идэвхжүүлэгч аль хэдийн өдөөгдсөн нөхцлийг үл тоомсорлох боломжийг олгодог.

1. Агент ба түүний орчин дасгал

```
import random, copy
class Object:
    #Энэ нь хүрээлэн буй орчинд гарч болох аливаа биет объектыг илэрхийлнэ.
    #Хүссэн обьектуудаа авахын тулд та Object-ийн дэд ангиллыг үүсгэж ашиглана.
    #Объект бүр нь __ name__slot (зөвхөн гаралтанд ашигладаг)-тай байж болно .
    def __repr__(self):
        return '<%s>' % getattr(self, '__name__', self.__class__.__name__)
```

```
    def is_alive(self):
        #Амьд объектууд үнэн утгыг буцаана
        return hasattr(self, 'alive') and self.alive

    def display(self, canvas, x, y, width, height):
        #Объектын дүрслэлийг харуулах
        pass

class Agent(Object):
    #Агент бол Обьектийн дэд анги бөгөөд .програм гэсэн нэг үүрийг заавал
    #шаарддаг. Энэ нь нэг аргумент, орчны буулгалтыг авч хариу үйлдлийг буцаах
    #функц байх ёстой. (Аливаа зүйл орчны буулгалт эсвэл хариу үйлдэл нь тухайн
    #агент оршин буй орчноос хамаарна.). Програм' нь арга биш харин слот гэдгийг
    #анхаарна уу. Хэрэв энэ арга байсан бол дараа нь програм нь өөрийгөө
    #засварлаж моделоо үүсгэх боно. Нэмэлт үүрүүд байдаг. Жишээ нь гүйцэтгэл нь
    #агентын орчин дахь агентийн гүйцэтгэлийн хэмжүүр утгыг тооцоолж болно.
    def __init__(self):
        def program(percept):
            return raw_input('Percept=%s; action? ' % percept)
        self.program = program
        self.alive = True

    def TraceAgent(agent):
        #Агентын програмыг оролт, гаралтыг багцалж хэвлэнэ. Энэ нь агент тухайн
        #орчинд юу хийж байгааг харуулна.

        old_program = agent.program
        def new_program(percept):
            action = old_program(percept)
            print('%s perceives %s and does %s' % (agent, percept, action))
            return action
        agent.program = new_program
        return agent
```

2. Хүснэгтэд суурилсан агент дасгал

```
class TableDrivenAgent(Agent):
    #Энэ агент нь орны буулгалтын дараалалд үндэслэн тохирох хариу үйлдлийг
    #сонгоно. Энэ нь зөвхөн жижиг домэйнуудад л ашиглагддаг. Үүнийг өөрчлөхийн
    #тулд та байгуулагчид хүснэгт өгнө.
    def __init__(self, table):
        #Бүх {percept_sequence: action} хосуудын толь бичгийг хүснэгт
        #болгон оруулна."
        #Агент програм нь зарчмын хувьд функц байж болох юм, гэхдээ class-ийн
        #дуудаж болох instance хэлбэрээр төлвүүдийг хадгалах хэрэгтэй.
        Agent.__init__(self)
        percepts = []
```

```
        def program(percept):
            percepts.append(percept)
            action = table.get(tuple(percepts))
            return action
        self.program = program
```

Тоос сорогчийн жишээний хувьд нь дээрх хүснэгтэд суурилсан агентыг үүсгэхдээ дараах орчны буулгалтыг тохирох хариу үйлдэлүүдтэй харгалзуулсан хүснэгтийг оролт болгон TableDrivenAgent агентыг дуудаж болно.

```
loc_A, loc_B = (0, 0), (1, 0) # Вакуум ертөнцийн хоёр байршил
def TableDrivenVacuumAgent():
    table = {((loc_A, 'Clean'),): 'Right',
             ((loc_A, 'Dirty'),): 'Suck',
             ((loc_B, 'Clean'),): 'Left',
             ((loc_B, 'Dirty'),): 'Suck',
             ((loc_A, 'Clean'), (loc_A, 'Clean')): 'Right',
             ((loc_A, 'Clean'), (loc_A, 'Dirty')): 'Suck',
             # ...
             ((loc_A, 'Clean'), (loc_A, 'Clean'), (loc_A, 'Clean')): 'Right',
             ((loc_A, 'Clean'), (loc_A, 'Clean'), (loc_A, 'Dirty')): 'Suck',
             # ...
             }
    return TableDrivenAgent(table)
```

3. Санамсаргүй агент

```
class RandomAgent(Agent):
    #Бүх орчны буулгалтыг үл тоомсорлож, санамсаргүй байдлаар
    #үйлдлийг сонгодог төлөөлөгч.
    def __init__(self, actions):
        Agent.__init__(self)
        self.program = lambda percept: random.choice(actions)
```

Тоос сорогчийн жишээний хувьд нь санамсаргүй агентыг үүсгэхдээ орчны буулгалтаас үл хамааран хариу үйлдэлүүдээс санамсаргүйгээр сонгож болох хариу үйлдэлүүдийг оролт болгон RandomAgent-г дуудна.

```
def RandomVacuumAgent():
    #Вакуум орчноос нэг үйлдлийг санамсаргүй байдлаар сонгоно
    return RandomAgent(['Right', 'Left', 'Suck', 'NoOp'])
```

4. Агентын орчин

```
class Environment:
    #Орчныг харуулсан хийсвэр анги. Үүнээс 'бодит' орчныг удамшуулж үүсгэнэ.
    #Орчин нь ихэвчлэн дараахь зүйлийг хэрэгжүүлэх шаардлагатай болно.
    # орчны тусгал: Агент хардаг ойлголтыг тодорхойл.
    # execute_action: Гүйцэтгэж байгаа үйлдлийн үр дүнг тодорхойлох.
    # Agent.performance slot-ийг шинэчлэх. Хүрээлэн буй орчны .objects болон .agents
    #жагсаалтыг хадгалдаг (энэ нь дэд хэсэг юм). Агент бүр нь 0-ээр эхэлсэн
```

```
#гүйцэтгэлийн үүр байна. Обьект бүр нь .location slot-тэй байдаг, гэхдээ зарим
#орчинд байхгүй байж болно.


def __init__(self,):
    self.objects = []; self.agents = []

object_classes = [] ## Орчинд нэвтрэх боломжтой хичээлүүдийн жагсаалт

def percept(self, agent):
    #Агентын харж буй ойлголтыг буцаана. Засварлаж утга оруулна
    abstract

def execute_action(self, agent, action):
    #Үйлдэл хийхэд хамаарах ертөнцийг өөрчил. Засварлаж утга оруулна
    abstract

def default_location(self, object):
    #Тодорхойгүй байршилтай шинэ объектыг байрлуулах үндсэн байршил
    return None

def exogenous_change(self):
    #Хэрэв дэлхий дээр тогтмол өөрчлөгдөж байгаа бол засварлаж утга оруул
    pass

def is_done(self):
    #Амьд агент олж чадахгүй байх үед дуусна
    for agent in self.agents:
        if agent.is_alive(): return False
    return True

def step(self):
     #Орчныг нэг удаагийн алхамаар ажиллуулна. Хэрэв үйлдлүүд ба гадны
     #өөрчлөлтүүд нь хамааралгүй бол энэ аргыг хийх болно. Хэрэв тэдгээрийн
     #хооронд харилцан хамаарал байгаа бол энэ аргыг засаж бичнэ.
      if not self.is_done():
        actions = [agent.program(self.percept(agent))
                   for agent in self.agents]
        for (agent, action) in zip(self.agents, actions):
                self.execute_action(agent, action)
        self.exogenous_change()

def run(self, steps=1000):
      #Хүрээлэн буй орчныг өгөгдсөн тооны алхамаар ажиллуулна
      for step in range(steps):
        if self.is_done(): return
        self.step()
```

```
def add_object(self, object, location=None):
    #Хүрээлэн буй орчинд объектыг нэмж байршлыг тохируулна. Мөн агентууд
    #төрлийн объектуудыг хянана.
    object.location = location or self.default_location(object)
    self.objects.append(object)
    if isinstance(object, Agent):
     object.performance = 0
     self.agents.append(object)
    return self
```

Даалгавар

Лабораторі 2-оор Simple Reflex Agent, Model-based reflex agent, Goal-based agents, Utility-based agent, Learning agent-ийн аль нэг загварыг өөрсдийн санаанд үндэслэн хөгжүүлнэ үү. Үүнд агент болон орчныг дүрслэнэ, харилцан үйлдэл гэх мэт орно.

Хугацаа 2 долоо хоног