King's College London

Degree Programmes	BSc, MSci,BEng,MEng
Module Code	6CCS3AIP
Module Title	Artificial Intelligence Planning
Examination Period	January 2019 (Period 1)

Time Allowed	Two hours
Rubric	ANSWER THREE OF FOUR QUESTIONS. All questions carry equal marks. If more than three questions are answered, the three answers with highest marks will count.
Calculators	Calculators may be used. The following models are permitted: Casio fx83 / Casio fx85.
Notes	Books, notes or other written material may not be brought into this examination

PLEASE DO NOT REMOVE THIS PAPER FROM THE EXAMINATION ROOM

Cleanup Domain and Problem for Question 1

```
(define (domain CleanupDomain)
    (:requirements :typing :fluents :durative-actions :duration-inequalities)
    (:types robot door location - object)
    (:predicates
         (exposure_available ?r - robot)
         (can_act ?r - robot)
         (door_at ?d - door ?l - location)
         (can_open ?d - door)
         (door_open ?d - door)
         (have_photo_inside ?l - location)
         (at ?r - robot ?l - location)
         (free ?r - robot)
   )

    (:durative-action exposure_window
         :parameters (?r - robot)
         :duration (and (<= ?duration 22) (>= ?duration 0.01))
         :condition (and
              (at start (exposure_available ?r)))
         :effect (and
              (at start (not (exposure_available ?r)))
              (at start (can_act ?r))
              (at end (not (can_act ?r)))
               )
   )

   (:durative-action move_robot
         :parameters (?r - robot ?from ?to - location)
         :duration (= ?duration 10)
         :condition (and
              (at start (at ?r ?from))
              (over all (can_act ?r))
              (over all (free ?r))
              )
         :effect (and
              (at start (not (at ?r ?from)))
              (at end (at ?r ?to))
```

```
            )
    )

     (:durative-action take_photo
          :parameters (?r - robot ?d - door ?react - location)
          :duration (= ?duration 1)
          :condition (and
               (over all (can_act ?r))
               (over all (door_open ?d))
               (over all (door_at ?d ?react))
               (over all (at ?r ?react))
               (at start (free ?r))
               )
          :effect (and
               (at start (not (free ?r)))
               (at end (free ?r))
               (at end (have_photo_inside ?react))
            )
    )

    (:durative-action open_door
          :parameters (?d - door ?l - location ?r - robot)
          :duration (= ?duration 2)
          :condition (and
               (at start (can_open ?d))
               (over all (door_at ?d ?l))
               (over all (can_act ?r))
               (at end (at ?r ?l))
               )
          :effect (and
                (at start (not (can_open ?d)))
                (at start (door_open ?d))
                (at end (not (door_open ?d)))
               )
    )
)
```

```
(define (problem CleaupProblem)
  (:domain CleanupDomain)
  (:objects
    reactor1door - door
    entrance reactor1  - location
    robby - robot
   )
  (:init
    (exposure_available robby)
    (can_open reactor1door)
    (at robby entrance)
    (free robby)
    (door_at reactor1door reactor1)
  )
  (:goal (and
    (have_photo_inside reactor1)
    (at robby entrance)
  ))
)
```

1. This question refers to the cleanup domain and problem on the previous pages, which models the activities of a robot assisting in nuclear cleanup. Specifically the robot's task is to take a photo of the inside of the reactor, which is itself inside a larger reactor containment building. The `exposure_window` action represents the maximum amount of time that the robot can be in the room without damage to its hardware (taking into account additional exposure at a higher level for 3 time units when the reactor door is open). The reactor door is opened via remote control by operators outside the reactor building, it can be opened at any time (without the robot being at the door) and will automatically close after 3 time units. The robot must be at the door when it closes (i.e. at the end of the `open_door` action) as operators use a camera mounted on the robot to verify that the door has closed properly.

a. This part of the question concerns the `exposure_available` predicate, which is a precondition and start delete effect of the `exposure_window` action.

i. What unintended behaviour could happen if this was not a precondition of `exposure_window` (or was not deleted at all by the `exposure_window` action)?

[3 marks]

ii. What could happen if `exposure_available` is a precondition of exposure window, but is deleted at the end of `exposure_window` rather than the start? How would this impact planner performance (time to find a solution)?

[2 marks]

QUESTION 1 CONTINUES ON NEXT PAGE

SEE NEXT PAGE

b. Could a decision epoch planner find a solution to this problem? Explain your answer.

[4 marks]

c. i. Draw the STN that CRIKEY3 would generate for the following partial plan (you may abbreviate action names if you wish, so long as it is clear which is which).

[8 marks]

```
(exposure_window robby) start
(move_robot robby entrance reactor1) start
(move_robot robby entrance reactor1) end
(open_door reactor1door reactor1 robby) start
(take_photo robby reactor1door reactor1) start
(take_photo robby reactor1door reactor1) end
(open_door reactor1door reactor1 robby) end
(move_robot robby reactor1 entrance) start
(move_robot robby reactor1 entrance) end
```

ii. Is this a temporally valid plan? Explain how you worked this out from your STN.

[2 marks]

d. i. How does the STN that POPF would build for this plan differ from the one you drew above for Crikey?

[4 marks]

ii. Can POPF find a solution from this plan? Explain your answer.

[2 marks]

2. This is a question about SMTPlan, and encoding PDDL+ as SAT Modulo theories. Here is a fragment of a *AUVdocking* domain:

Actions and Processes:

```
(:process hover
 :parameters (?u - auv)
 :precondition (>= (energy ?u) 0)
 :effect (and
   (increase (pos ?u) (* #t (vel ?u))))
   (decrease (energy ?u) (* #t 5.6))
 )
)


(:action dock
 :parameters (?u - auv)
 :precondition (= (pos ?u) 1000.0)
 :effect (and (docked ?u))
)


(:durative-action brake
 :parameters (?u - auv)
 :duration (>= ?duration 0)
 :condition (over all (>= (energy ?u) 0))
 :effect (and (decrease (vel ?u) (* #t 2.4)))
)
```

QUESTION 2 CONTINUES ON NEXT PAGE SEE NEXT PAGE

The auv must reach the docking station (which is in position 1000) at velocity 0, so that it can then dock and recharge. As it is moving towards the docking station, the auv is able to brake in order to decrease its speed.

a. SMTPlan solves PDDL+ problems by encoding them as SMT formulae, and solving the formulae using an SMT solver. SMTPlan uses iterative deepening on the number of happenings.

i. What is meant by a happening in this context?

[1 marks]

ii. Explain how iterative deepening works in the context of SMTPlan. What effect does this approach have on finding optimal solutions to PDDL+ problems?

[3 marks]

iii. The zero-crossing problem occurs when dealing with a domain containing continuous non-linear change. Explain what is the zero-crossing problem, using this domain as an example.

[3 marks]

iv. Explain how the gradient of the continuous change can be used to alleviate the zero-crossing problem. In SMTPlan, what impact does this have on the kinds of dynamics that can be soundly solved?

[3 marks]

QUESTION 2 CONTINUES ON NEXT PAGE

SEE NEXT PAGE

Here is a fragment of an SMT encoding of the lander problem:

```
(declare-fun (dock auv01)0 () Bool)
(declare-fun (docked auv01)0_0 () Bool)
(declare-fun (docked auv01)0_1 () Bool)

(declare-fun (dock auv01)1 () Bool)
(declare-fun (docked auv01)1_1 () Bool)
(declare-fun (docked auv01)1_0 () Bool)

(not (docked auv01)0_0))

(=> (docked auv01)0_1
    (or (docked auv01)0_0 (dock auv01)0)))
(=> (not (docked auv01)0_1)
    (not (docked auv01)0_0)))

(=> (docked auv01)1_1
    (or (docked auv01)1_0 (dock auv01)1)))
(=> (not (docked auv01)1_1)
    (not (docked auv01)1_0)))

(=> (dock auv01)0 (docked auv01)0_1))
(=> (dock auv01)1 (docked auv1)1_1))

(docked auv01)1_1)
```

There are two Boolean variables representing the application of the *(dock auv01)* action. There are four Boolean variables describing the *(docked auv01)* fact, just before and after

QUESTION 2 CONTINUES ON NEXT PAGE **SEE NEXT PAGE**

two distinct happenings. The constraints above show that *auv01* is not docked in the first happening, and that it should be in the second. There are also several implications.

b. Assuming that there are no other constraints containing the *(docked auv01)* variables, explain how the constraints above are not sufficient. Prove this by finding a valid assignment that does not correspond to the PDDL+ semantics.

[4 marks]

c. Extend the constraints with one or more new constraints to fix this issue.

[4 marks]

d. The following real variables are used to model the absolute time of the two happenings, the position of the auv, and the durative actions and processes during the interval between the happenings.

```
(declare-fun t0 () Real)
(declare-fun t1 () Real)

(declare-fun (pos auv01)0_0 () Real)
(declare-fun (pos auv01)0_1 () Real)
(declare-fun (pos auv01)1_1 () Real)
(declare-fun (pos auv01)1_0 () Real)

(declare-fun (hover auv01)0_run () Bool)
(declare-fun (brake auv01)0_run () Bool)
```

Using these variables, extend the constraints above in order to:

i. Ensure that the initial happening occurs at time zero.

[1 marks]

QUESTION 2 CONTINUES ON NEXT PAGE **SEE NEXT PAGE**

ii. Ensure that the position of the auv descent does not change instantaneously within a happening.

[2 marks]

iii. Ensure a valid continuous change in *(pos auv01)* between the happenings.

[4 marks]

Continuous Version of Cleanup Domain and Problem for Question 3

```
(define (domain CleanupDomainContinuous)
     (:requirements :typing :fluents :durative-actions :duration-inequalities)
     (:types robot door location - object)
     (:predicates
          (can_act ?r - robot)
          (door_at ?d - door ?l - location)
          (can_open ?d - door)
          (door_open ?d - door)
          (have_photo_inside ?l - location)
          (at ?r - robot ?l - location)
          (free ?r - robot)
    )
(:functions (exposure ?r - robot) )

    (:durative-action move_robot
          :parameters (?r - robot ?from ?to - location)
          :duration (= ?duration 10)
          :condition (and
               (at start (at ?r ?from))
               (over all (can_act ?r))
               (over all (free ?r))
               (over all (<= (exposure ?r) 27)))
          :effect (and
               (at start (not (at ?r ?from)))
               (at end (at ?r ?to))
               (increase (exposure ?r) (*#t 1))))
```

```
 (:durative-action take_photo
      :parameters (?r - robot ?d - door ?react - location)
      :duration (= ?duration 1)
      :condition (and
           (over all (can_act ?r))
           (over all (door_open ?d))
           (over all (door_at ?d ?react))
           (over all (at ?r ?react))
           (at start (free ?r)))
      :effect (and
           (at start (not (free ?r)))
           (at end (free ?r))
           (at end (have_photo_inside ?react))))

(:durative-action open_door
      :parameters (?d - door ?l - location ?r - robot)
      :duration (= ?duration 2)
      :condition (and
           (at start (can_open ?d))
           (over all (door_at ?d ?l))
           (over all (can_act ?r))
           (at end (at ?r ?l)))
      :effect (and
            (at start (not (can_open ?d)))
            (at start (door_open ?d))
            (at end (not (door_open ?d)))
            (increase (exposure ?r) (*#t 3)))))
```

```
(define (problem CleaupProblem)
  (:domain CleanupDomainContinuous)
  (:objects
    reactor1door - door
    entrance reactor1  - location
    robby - robot
  )
  (:init
    (= (exposure robby) 0)
    (can_open reactor1door)
    (at robby entrance)
    (free robby)
    (door_at reactor1door reactor1)
    (can_act robby)
    (free robby)
  )
  (:goal (and
    (have_photo_inside reactor1)
    (at robby entrance)
  ))
)
```

3. This question refers to an alternative model of the robot cleanup domain (see previous pages) that has been proposed, which makes use of continuous effects. These model the total exposure of the robot, and an overall constraint on the move action ensures that the robot can only return to the entrance, and thus complete the plan, if the total exposure is below a specified threshold.

a. i. The fact (door_at reactor1door reactor1) is a static fact. What is a static fact?

[2 marks]

ii. How can we exploit the fact that some facts are static to reduce the memory overheads of storing states during search?

[3 marks]

b. i. Write down the LP that COLIN would generate to check the temporal and numeric constraints for the following partial plan:

[10 marks]

```
(move_robot robby entrance reactor1) start
(open_door reactor1door reactor1 robby) start
(move_robot robby entrance reactor1) end
(open_door reactor1door reactor1 robby) end
```

ii. Can COLIN find a feasible solution to this LP or would the state resulting from this partial plan be pruned from search?

[3 marks]

QUESTION 3 CONTINUES ON NEXT PAGE

SEE NEXT PAGE

c. Suppose there is another reactor, `reactor2`, within the same containment building. The operators would also like the robot to photograph inside `reactor2`, and if it does it should do the photograph of `reactor1` first. They wish to express this as a preference in order to allow the planner to satisfy this if it is able to, but to still get a plan if this is not possible.

   i. How would you represent this preference in PDDL?

   [3 marks]

   ii. Draw the automaton for the preference
   `(at-most-once (door_open reactor1door))`.

   [4 marks]

4. This question is about PDDL+. For reference, here is *a fragment* of a variant of the *generator domain*:

Actions:

```
(:durative-action generate
 :parameters (?g - generator)
 :duration (>= ?duration 0)
 :condition ( and (over all (>= (fuelLevel ?g) 0))
                            (over all (safe ?g)))
 :effect (and (decrease (fuelLevel ?g) (* #t 1))
  (at start (increase (power-supplied) 10))
         (at end (decrease (power-supplied) 10))
)


(:action refuel
 :parameters (?g - generator ?t - tank)
 :precondition (and (not (using ?t ?g)) (available ?t))
 :effect (and (using ?t ?g) (not (available ?t)))
)


(:process refuelling
 :parameters (?g - generator ?t -tank)
 :precondition (and (using ?t ?g))
 :effect (and (increase (fuelLevel ?g) (* #t 2))
      (decrease (fuelInTank ?t) (* #t 1)))
)
```

QUESTION 4 CONTINUES ON NEXT PAGE SEE NEXT PAGE

The generator(s) must be used to provide enough power to satisfy the demand defined in the problem file. Each generator is initially filled to its 100-unit fuel capacity, and consumes fuel at a rate of 1 unit per second. When active, each generator provides 10 units of power. The generator can be refilled using tanks, the action to do this is named refuel and increases the fuel level in the generator at a rate of 2 units per second.

a. In the fragment above, once the process `refuelling` is triggered, it would run for ever. What is needed to avoid this odd behaviour and to have a correct domain for this problem?

[3 marks]

b. Extend the domain with the corresponding PDDL+ construct.

[4 marks]

c. In the fragment above, each generator can be used arbitrarily often. Instead, we want to change the model so that once a generator has been switched off it cannot be switched on again. Modify the domain to satisfy this new constraint.

[3 marks]

d. The pair of action `refuel` and process `refuelling` can be both replaced by a single durative action `refuel`. Modify the domain by adding this new durative action and discuss possible differences between the two design choices.

[9 marks]

QUESTION 4 CONTINUES ON NEXT PAGE **SEE NEXT PAGE**

e. In the current model, the requirement of producing enough power to satisfy the demand is not modelled. Assuming the function `demand` is used to represent the current demand, describe how the domain needs to be modified so that a valid plan will have to satisfy the demand requirements.

[6 marks]