Faculteit der Exacte Wetenschappen	Tentamen Ontwerp van Multi-agentsystemen
Vrije Universiteit Amsterdam	2 April 2002

Question	**1**	**2**	**3**	**4**	**bonus**
Points	25	25	20	20	10

Norm:
Let Pi stand for the points for question i. Then the **examination grade**

T = (P1 + P2 + P3 + P4 + 10) / 10.

The **end grade** for Design of Multi-Agent Systems is calculated as follows:

End grade = (T + H + P) / 3,

where
T = examination grade,
H = homework grade,
P = grade for 1 study point practical work.

You are kindly requested to use the special **answer sheet** for your answer where indicated.

Question 1 (25 punten)

Read Appendix 1 and answer the following questions.

a) (5 points) Provide a graphical representation of the top-level of process abstraction for the MOBIE system. Include the human customers as agents in that picture. Motivate each link between processes, and explain the type of information exchanged.

b) (10 points) The generic agent model of Chapter 6 consists of 7 components: agent_interaction_management, world_interaction_management, maintenance_of_agent_information, maintenance_of_world_information, own_process_control, and agent_specific_task. Which of these components do you need and which do you not need to model a personal assistant agent of the MOBIE system? Motivate your answer and make explicit references to the text of Appendix 1.

c) (10 points) The process of interaction that the personal agent needs is rather complicated. The type of communication that it needs with the human customers is rather different from that with the other software agents. Furthermore, for communication with the human user it has different channels. Suppose that component comp_c of the agent is responsible for all this. Then component comp_c should be composed. Provide a process composition of comp_c and the links needed within comp_c to model these processes. Motivate your answers in a rationale.

Question 2 (25 points)

This question builds on your understanding of the generic model for Reasoning with and about Assumptions (Chapter 11). For your convenience a rather detailed partial specification of that model is given in Appendix 3. Be careful to focus directly on the parts of the specification that you need, so that you don't waste time. This generic model will be used in this exercise to diagnose the fiber problems of a fiber production plant. Read Appendix 2 "Fiber Problem".

a) (10 points) Give a knowledge base for component assumption_determination that reflects the knowledge in Appendix 2. Motivate your answer in a rationale.
b) (10 points) Give a knowledge base of component observation_result_prediction that reflects the knowledge in Appendix 2. Motivate your answer in a rationale.
c) (5 points) Design the information types causes and symptoms for this domain. You can do this in one information type, but you are also allowed to make more levels of abstraction. Motivate your answers, refer back to your answers to questions a) and b) as well.

Question 3 (20 points)

For this question the partial specification of Appendix 3 is used again. Suppose that the information types causes and symptoms are specified as follows:

```
information type causes
relations          a;
end information type

information type symptoms
relations          b;
end information type
```

a) (10 points) Consider information link predictions of the specification. Consider the following information states:

   output information state of component observation_result_prediction is

   [predicted_for(b, pos, a, pos)].

   input information state of component assumption_evaluation is

   [assumed(a, pos), predicted_for(b, neg, a, pos)]

   Give the input information state of component assumption_evaluation after execution of link predictions on the basis of the above information states.
b) (10 points) Give a trace of the behaviour of component assumption_evaluation given that the subsequent **input information states** of that component are as is presented in Table 1. Use the answer sheet and fill in your answer in Table 1.

Question 4 (20 points)

This question is about information states and reasoning. Study the partial specification of Appendix 4. This is the object level public information state S of component mouse_a.

S = [observation_result(at_position(self, p0), pos),
observation_result(at_position(food, p1), pos),
observation_result(at_position(screen, p0), neg)]

a) Provide an information state S', that refines S and is also closed and consistent with respect to the knowledge base of component mouse_a. (8 points)
b) Motivate that S' is a refinement of S (4 points).
c) Motivate that S' is closed with respect to the knowledge base of component mouse_a (4 points). Motivate that S' is consistent with respect to the knowledge base of component mouse_a (4 points).

Appendix 1 The MOBIE system

Prepay usage as a percentage of overall mobile (also called cell) phone access has increased sharply over the past several years. However, the recharging process is still largely manual with personalization provided by the user. A system is needed capable of automatically recharging the prepaid account of a mobile phone in a personalized manner. This visionary system is called MOBIE. The MOBIE multi-agent system consists of personal assistant agents for the consumers and business agents for the mobile telecommunication service providers. The MOBIE system has to take care of the personalization of the agents, security, and human agent interaction modalities.

To accommodate the automated recharging process for the user the mobile phone service providers need to be able to interact with the personal assistant agents in a reliable and secure manner. Because of the expected high frequency of such interactions the service providers need to automate these customer interactions. The option chosen in this paper is to introduce business agents that are capable of the required interactions with the personal agents of the users. The personal assistant agent that represents the customer is capable of the following main tasks.

1. The personal agent creates and maintains a profile of the customer, containing:
    a. The criteria that tell the agent when to recharge the account.
    b. The information needed to execute recharging, like the amounts it can use, and payment information.
2. The personal agent matches the criteria against the actual balance of the prepaid account.
3. The personal agent requests the necessary information from the business such as:
    a. The balance of the prepaid account.
    b. The actual usage pattern of the phone for a specified period of time.
4. The personal agent is capable of recharging the prepaid account.
5. The personal agent can ask the telecom companies (through the business agents that represent them) to recharge the prepaid account with amount x.
6. The personal agent is responsible for keeping the customer informed in accordance to the customer profile.
7. The personal agent is able to interact with the customer through different channels:
    a. web-based,
    b. WAP (for those customers that have a WAP enabled mobile phone)
    c. voice. Due to inherent restrictions of current WAP implementations and of mobile devices in general, we think that a voice-enabled interface has high potential.

The personal assistant agents function within MOBIE in an environment consisting of business agents that represent the different telecom companies, and financial institutions (like banks, with whom the actual payment is to be arranged). The personal assistant agents do not contact the financial institutions themselves. They can ask telecom company to recharge the prepaid account, the telecom company will then contact the appropriate financial institution.

Appendix 2   Fiber Problem

Consider the following situation, which involves two agents, the manager of a nylon fiber production plant (M) and a nylon fiber specialist (F). M, who is always interested in producing fibers of the best quality, suddenly is confronted with unsatisfied customers. A problem that needs fixing right away. As he is not able to find out himself what the cause of this problem is, he decides to ask F for help.

A specific responsibility of F is to make a diagnosis of fiber production problems of his clients communicated to him by phone. F has no possibility to observe the differnent parts of the process, therefore, he asks M to make certain observations and communicate them back to F.

To determine the nature of the problem, F uses a line of reasoning modelled by the generic model for reasoning with and about assumptions (see Appendix 3). That model proceeds along the following lines: making assumptions (in some kind of order), predicting observation results for that assumption, and then evaluating the assumption by making the appropriate observations and comparing them to the assumption. If necessary, the old assumptions are rejected, and new ones are made. F uses the hierarchy (taxonomy) depicted in Figure 1 of the subproblems of fiber problems, that he uses to efficiently order the assumptions he can make:

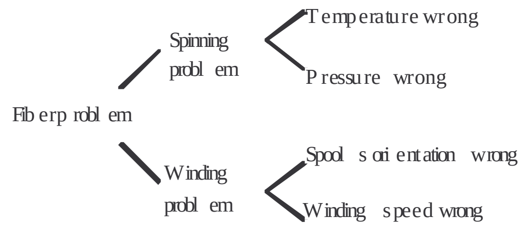

Figure 1

F can instruct M to make the following observations:

- is the fiber fluffy or smooth?
- is the fiber brittle or flexible?
- is the thickness of the fiber uneven or even?

The causal relations between diseases and observations known to F are depicted in Figure 2. If the temperature is too high, then the fiber becomes brittle instead of flexible. A high temperature can also lead to fluffy fibers, because the fibers tend to stick together a bit. If the pressure is too high, this can cause an uneven fiber because the cooling down of the fiber is not controlled enough. If the orientation of the spools is not exactly vertical, then the fibers are pulled in an uneven way, which may lead to an uneven fiber. Also the friction of the fiber against the guiders of the spool might cause a bit of fluffiness. If the winding speed is wrong, the fiber that is still cooling down will become too thin and thus brittle.

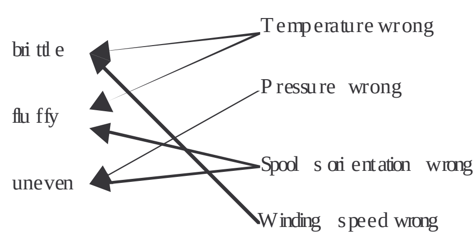

Figure 2

Appendix 3 Reasoning with and about assumptions

information types

information type **truth_indication**
  sorts SIGN
  objects pos, neg: SIGN
end information type

information type **obs_to_be_performed**
  sorts INFO_ELEMENT
  relations to_be_observed: INFO_ELEMENT ;
end information type

information type **observation_results**
  sorts INFO_ELEMENT,
  SIGN
  relations observation_result: INFO_ELEMENT * SIGN ;
end information type

information type **assumptions_hypotheses_and_such**
  sorts INFO_ELEMENT, SIGN
  relations assumed: INFO_ELEMENT * SIGN ;
  rejected: INFO_ELEMENT * SIGN ;
  has_been_considered: INFO_ELEMENT * SIGN ;
  possible_assumption: INFO_ELEMENT * SIGN ;
  predicted: INFO_ELEMENT * SIGN ;
  predicted_for: INFO_ELEMENT * SIGN * INFO_ELEMENT * SIGN ;
  known_to_hold: INFO_ELEMENT * SIGN ;
end information type

information type **causes**
... ...
end information type

information type **symptoms**
... ...
end information type

information type **world_info**
  information types symptoms, causes;
end information type

INFO_ELEMENT
meta-descriptionsworld_info : WORLD_INFO_ELEMENT;
sub sorts WORLD_INFO_ELEMENT: INFO_ELEMENT;
end information type

information type **domain_meta_info**
information types world_meta_info;
end information type

information type **observation_info**
information types obs_to_be_performed, domain_meta_info;
end information type

information type **observation_result_info**
information types observation_results, domain_meta_info, truth_indication;
end information type

information type **assumption_info**
information types domain_meta_info, truth_indication, assumptions_hypotheses_and_such;
end information type

component assumption_determination

input information types assumption_info, observation_result_info;
output information type assumption_info;

initial kernel information level_2
assumption(has_been_considered(HYP: INFO_ELEMENT, S: SIGN), neg);

knowledge base assumption_determination_local_kbs
information types assumption_info, observation_result_info;
contents
/* use as many rules as you like, you may also create additional information types if you like. */

... ...

end knowledge base

component assumption_evaluation

input information types observation_result_info, assumption_info;
output information type observation_info, assumption_info;

information types observation_result_info, assumption_info, observation_info;

```
contents
if      predicted_for(OBS: INFO_ELEMENT, S1: SIGN, HYP: INFO_ELEMENT, S2: SIGN)
then    to_be_observed(OBS: INFO_ELEMENT);

if      assumed(HYP: INFO_ELEMENT, S: SIGN)
  and   predicted_for(OBS: INFO_ELEMENT, pos, HYP: INFO_ELEMENT, S: SIGN)
  and   observation_result(OBS: INFO_ELEMENT, neg)
then    rejected(HYP: INFO_ELEMENT, S: SIGN)
  and   has_been_considered(HYP: INFO_ELEMENT, S: SIGN);

if      assumed(HYP: INFO_ELEMENT, S: SIGN)
  and   predicted_for(OBS: INFO_ELEMENT, neg, HYP: INFO_ELEMENT, S: SIGN)
  and   observation_result(OBS: INFO_ELEMENT, pos)
then    rejected(HYP: INFO_ELEMENT, S: SIGN)
  and   has_been_considered(HYP: INFO_ELEMENT, S: SIGN);

end knowledge base
```

component observation_result_prediction

input information types assumption_info;
output information type assumption_info;

knowledge base observation_result_prediction_local_kbs
information types assumption_info;

contents

/* use as many rules as you like */

end knowledge base

component Manager (External World)

task control foci observations;
input information type target_observation_result_info; /* standard type */
output information type symptoms;
alternative specification user

information links

private link **hypotheses** : object - object
domain assumption_determination
information type assumption_info;
co-domain assumption_evaluation

sort links identity

object links identity

term links identity

atom links

(possible_assumption(HYP: INFO_ELEMENT, S: SIGN),

assumed(HYP: INFO_ELEMENT, S: SIGN)): <<true,true>, <false,false>,<unknown, unknown>>;

end link

private link **assessments** : object - object

domain assumption_evaluation

information type assumption_info;

co-domain assumption_determination

information type assumption_info;

sort links identity

object links identity

term links identity

atom links

(rejected(HYP: INFO_ELEMENT, S: SIGN),

rejected(HYP: INFO_ELEMENT, S: SIGN)): <<true, true>, <false, false>>;

(has_been_considered(HYP: INFO_ELEMENT, S: SIGN),

has_been_considered(HYP: INFO_ELEMENT, S: SIGN)): <<true, true>, <false, false>>;

end link

private link **required_observations** : object - target

domain assumption_evaluation

information type observation_info;

co-domain external_world

information type target_observation_result_info; /* standard type */

sort links (WORLD_INFO_ELEMENT, OA)

object links identity

term links identity

atom links

(to_be_observed(OBS: WORLD_INFO_ELEMENT),

target(observations, OBS: OA, determine)) :

<<true, true>, <unknown, false>, <false,false>>;

end link

private link **observation_results** : epistemic - object

domain external_world

information type epistemic_world_info; /* standard meta-level */

co-domain assumption_evaluation

```
sort links (OA, WORLD_INFO_ELEMENT)
object links identity
term links identity
atom links
        (true(OBS: OA), observation_result(OBS: WORLD_INFO_ELEMENT, pos)) :
                        <<true, true>,<false,false>>;
        (false(OBS: OA), observation_result(OBS: WORLD_INFO_ELEMENT, neg)) :
                        <<true, true>,<false,false>>;
end link


private link assumptions : object - object
domain assumption_determination
        information type assumption_info;
co-domain observation_result_prediction
         information type assumption_info;

sort links identity
object links identity
term links identity
atom links
        (possible_assumption(HYP: INFO_ELEMENT, S: SIGN),
                assumed(HYP: INFO_ELEMENT, S: SIGN)) :
                        <<true, true>, <unknown, false>, <false,false>>;
end link


private link predictions : object - object
domain observation_result_prediction
        information type assumption_info;
co-domain assumption_evaluation
        information type assumption_info;

sort links identity
object links identity
term links identity
atom links
        (predicted_for(OBS: INFO_ELEMENT, S1: SIGN, HYP: INFO_ELEMENT, S2: SIGN),
                predicted_for(OBS: INFO_ELEMENT, S1: SIGN, HYP: INFO_ELEMENT, S2: SIGN)) :
                        <<true, true>, <unknown, unknown>, <false,false>>;
end link
```

Appendix 3 Mouse A

3.1 Problem Description

Separated by a transparent screen (a window, at position p0), at each of two positions p1 and p2 a cup (upside down) and/or a piece of food can be placed. At some moment (with variable delay) the screen is raised, and the mouse is free to go to any position. A genuine mouse is known to go to food and eat it.

In sections 3.2 and 3.3 of this Appendix a partial specification can be found of the artificial mouse.

3.2 Information types

The information types used within the system are:

```
information type truth_indication
        sorts                                   SIGN ;
        objects   pos,
                  neg :                         SIGN;
  end information type

  information type observation_results
        sorts                                   INFO_ELEMENT, SIGN ;
        relations observation_result:           INFO_ELEMENT * SIGN;
end information type

information type domain_meta_info
        sorts                                   INFO_ELEMENT ;
        meta-descriptions domain_info :         INFO_ELEMENT;
end information type

information type observation_result_info
        information types truth_indication,
                                                observation_results,
                                                domain_meta_info;
end information type

information type actions_to_be_performed
        sorts                                   ACTION ;
        relations to_be_performed:              ACTION;
end information type
```

domain_actions;

end information type

3.3 Partial specification of the component

The component is primitive and is shortly described here.

The component mouse_a

The interfaces are defined by:

input interface: de informatietypen observation_result_info;

output interface: het informatietype action_info;

The targets associated to task control focus determine_action are:

target(determine_action, to_be_performed(X: ACTION), confirm);

The initial extent is: all-p

The knowledge base is:

```
if      observation_result(at_position(food, P:POSITION), pos)
  and   observation_result(at_position(screen, p0), neg)
  and   observation_result(at_position(self, P:POSITION), neg)
then    to_be_performed(goto(P:POSITION))

if      observation_result(at_position(self, P:POSITION), pos)
  and   observation_result(at_position(food, P:POSITION), pos)
then    to_be_performed(eat)
```

Answer sheet Student: Year:

Input (1)	[assumed(a, pos), predicted_for(b, pos, a, pos)]
Output after revision but before reasoning	
Output after reasoning	
Input (2)	[assumed(a, pos), predicted_for(b, pos, a, pos), observation_result(b, neg)]
Output after revision but before reasoning	
Output after reasoning	
Input (3)	[assumed(a, neg), predicted_for(b, neg, a, neg), observation_result(b, neg)]
Output after revision but before reasoning	
Output after reasoning	

Table 1