Faculteit der Exacte Wetenschappen

Tentamen Ontwerp van Multi-agentsystemen / Design of Multi-Agent Systems

Vrije Universiteit Amsterdam **29 april 2003**

Opgave/Exercise	**1**	**2**	**3**	**4**	**5**	**bonus**
Punten/points	20	25	20	15	10	10

Normering:	*Norm:*
Het **tentamencijfer T** is gelijk aan het totaal behaalde punten voor de tentamenopgaven plus de bonus punten gedeeld door 10.	The tentamination mark **T** equals sum of the points scored for the exercises plus 10 bonus points divided by 10.
Het **Eindcijfer** voor het hoorcollege Ontwerp van Multi-agentsystemen wordt als volgt berekend.	The endmark **Eindcijfer** for the course Design of Multi-Agent Systems is calculated as follows:
Eindcijfer = (T + H + P) / 3	
Waarbij	Where
T = tentamencijfer H = cijfer huiswerkopgave P = cijfer voor het klein practicum	T = tentamination mark H = mark for the home work exercises P = mark for the small practicum

U treft aan:

5 opgaven
5 appendices

You find:

5 exercises
5 appendices

Opgave 1 (20 punten)

Deze opgave bestaat uit twee onderdelen. Motiveer Uw antwoorden.

Opgave 1a (10 punten)

In hoofdstuk 1 van de syllabus zijn een aantal primitive agentconcepten geïntroduceerd (zie Appendix 1B van de antwoordvellen). In Appendix 1A kun je wat informatie vinden over verzekeringsagent Okke. Analyseer deze informatie aan de hand van de primitieve agentconcepten en vul Appendix 1B (3 antwoordvellen) in. Als de informatie in Appendix 1A naar je zin niet precies genoeg is om de tabel goed te kunnen invullen, dan mag je er zelf informatie bij verzinnen. Denk er aan dat je je antwoorden goed motiveert.

Opgave 1b (10 punten)

In het generieke agentmodel (hoofdstuk 6 van de syllabus) komen een aantal componenten voor. Stel dat je het generieke agentmodel zou gebruiken om een artificiële agent te ontwerpen die Okke's taak zou kunnen overnemen. Gebruik je antwoord op vraag 1a om te beslissen welke van deze componenten je nodig hebt in je ontwerp. Motiveer waarvoor een component nodig is, motiveer ook waarom een component eventueel niet nodig is.

English:
Question 1 (20 points)

This question consists of two parts. Motivate your answers.

Question 1a (10 points)

In chapter 1 of the syllabus a number of primitive agent concepts have been introduced (see Appendix 1B of the answer sheets). In Appendix 1A, you can find some information on insurance agent Okke.
Analyse this information according to the primitive agent concepts and fill out Appendix 1B (3 answer sheets). In case you feel the information in Appendix 1A is not detailed enough to fill out the table properly, you are allowed to make up additional information. Remember to motivate your answers clearly.

Question 1b (10 points)

In the generic agent model (chapter 6 of the syllabus), there are a number of components. Suppose you would use the generic agent model to design an artificial agent that can take over Okkes task. Use your answer to question 1a to decide which of these components you need in your design. Motivate why a component is needed; in case you left out a component, motivate why this component is not necessary.

Question 2 (25 points). Een Nederlandse vertaling van Question 2 is niet beschikbaar.

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 problems of growing sunflowers. Read Appendix 2 "Sunflower 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.

Opgave 3 (20 punten).

Het doel van deze opgave is het bestuderen van informatie toestanden, informatie links, en het revisie proces. Beschouw weer de partiele specificatie van Appendix 3. Neem aan dat de informatie types causes and symptoms hebben de volgende specificatie:

information type **causes**
relations a;
end information type

information type **symptoms**
relations b;
end information type

a) (10 punten) Beschouw informatie link predictions van de specificatie. Beschouw de volgende informatie toestanden:

output informatie toestand van component observation_result_prediction is

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

input informatie toestand van component assumption_evaluation is

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

Geef de input informatie toestand van component assumption_evaluation na uitvoering van de link predictions op basis van deze informatie toestanden.

b) (10 punten) Geef een trace van het gedrag van component assumption_evaluation gegeven dat de achtereenvolgende **input informatie toestanden** van die component als in Appendix 4 gepresenteerd zijn. Gebruik de speciale antwoordformulieren uit Appendix 4.

English:
Question 3 (20 points).

The purpose of this question is to study information states, information links and the revision process. 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

c) (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

[observation_result(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.

d) (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 Appendix 4. Use the answer sheet and fill in your answer in Appendix 4.

Opgave 4 (15 punten)

Beschouw het domein van tafeldekken voor vier personen, zie Figuur 1. Het dekken van een tafel kan worden beschouwd als een proces dat moet worden bestuurd. In het ontwerp van een procesbesturingssysteem voor dit domein wordt gebruik gemaakt van het model voor procesbesturing zoals beschreven is in Chapter 3 van de syllabus. De domeinspecifieke informatietypes moeten worden aangepast.

a) Ontwerp het informatietype domain info voor dit domein. Je kunt dit in één informatietype doen, maar je mag ook meer abstractieniveaus aanbrengen.

b) Ontwerp het informatietype domain actions voor dit domein. Je kunt dit in één informatietype doen, maar je mag ook meer abstractieniveaus aanbrengen.

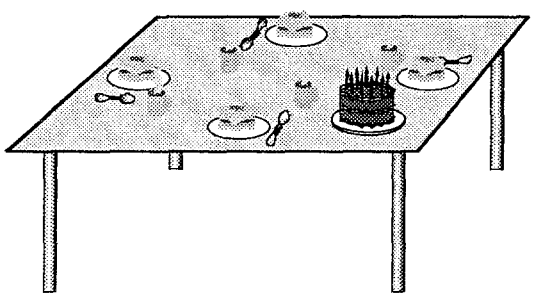


Figure 1

English:

Question 4 (15 points)

Consider the domain of laying a table for four persons for dinner, see Figure 1. The laying of the table can be seen as a process that is to be controlled. In the design of a process controller for this domain, the design of the process control model described in Chapter 3 is to be reused. The domain specific information types have to be adapted.

a) Design the information type domain info for this domain. You can do this in one information type, but you are also allowed to make more levels of abstraction.

b) Design the information type domain actions for this domain. You can do this in one information type, but you are also allowed to make more levels of abstraction.

Opgave 5(10 punten)

Beschouw het volgende informatietype:

```
information type insect_stuf
    sorts          BEE, INSECT;
    subsorts       BEE :   INSECT;
    objects        a, b: BEE;
                   c, d: INSECT;
    relations      can_fly, is_a_bee, is_a_spider: INSECT;
end information type
```

En de volgende kennisbank:

```
knowledge base insect_kb
    information types insect_stuf
contents
    is_a_bee(X: BEE);
    if not can_fly(X: INSECT) then not is_a_bee(X: INSECT);
    if is_a_spider(X: INSECT) then not can_fly(X: INSECT);
    is_a_spider(c);
end knowledge base
```

Geef een minimale verfijning (refinement) van de informatie toestand [ ] die zowel gesloten (closed) en consistent is ten opzichte van de kennisbank insect_kb.

English:
Question 5 (10 points)

Consider the following information type:

```
information type insect_stuf
    sorts          BEE, INSECT;
    subsorts       BEE :   INSECT;
    objects        a, b: BEE;
                   c, d: INSECT;
    relations      can_fly, is_a_bee, is_a_spider: INSECT;
end information type
```

And the following knowledge base:

```
knowledge base insect_kb
    information types insect_stuf
contents
    is_a_bee(X: BEE);
    if not can_fly(X: INSECT) then not is_a_bee(X: INSECT);
    if is_a_spider(X: INSECT) then not can_fly(X: INSECT);
    is_a_spider(c);
end knowledge base
```

Give a minimal refinement of information state [ ] that is both closed and consistent with respect to the knowledge base insect_kb.

Appendix 1A: Verzekeringsagent

Verzekeringsagent Okke is iemand die verzekeringen verkoopt aan mensen en bedrijven. Deze persoon gaat bij potentiële klanten op bezoek en bespreekt met de klant waartegen hij/zij zich wil verzekeren. Er zijn verschillende verzekeringsprodukten en verschillende polissen voor die produkten. Okke werkt voor een verzekeringsmaatschappij met een hoofdkantoor waar bepaald wordt welke verzekeringsprodukten deze maatschappij heeft en welke polissen daarvoor gelden. Als hij daar om vraagt, krijgt Okke van het hoofdkantoor alle informatie over de produkten, polissen en klanten die hij nodig heeft om zijn werk te kunnen doen. De verzekeringsagent wordt zelf geacht in de gaten te houden wat de concurrentie aan produkten heeft.

English:

Insurance Agent

Insurance agent Okke is selling insurances to people and companies. He visits potential customers and discusses with them which risks he/she wants covered. There are different insurance products and different policies for those products. Okke works for an insurance company with head-quarters where decisions are made regarding which insurance products are available and which policies there are. If Okke asks for it, he is provided by head-quarters with all information on products, policies and customers he needs to do his job. The insurance agent is supposed to keep track of the products of the competitors by himself.

Appendix 1B: Answersheet (1 out of 3)

I. External primitive concepts	
A. Interaction with the world	
passive observations	
active observations	
performing actions	
B. Communication with other agents	
incoming	
outgoing	

Appendix 1B Answersheet (2 out of 3)

II. Internal primitive concepts	
A. World Model	
B. Agent Models	
C. Self Model	
D. History	
E. Goals	
F. Plans	
G. Group Concepts	
Joint goals	
Joint plans	
Commitments	
Negotiation protocols	
Negotiation strategies	

Appendix 1B Answersheet (3 out of 3)

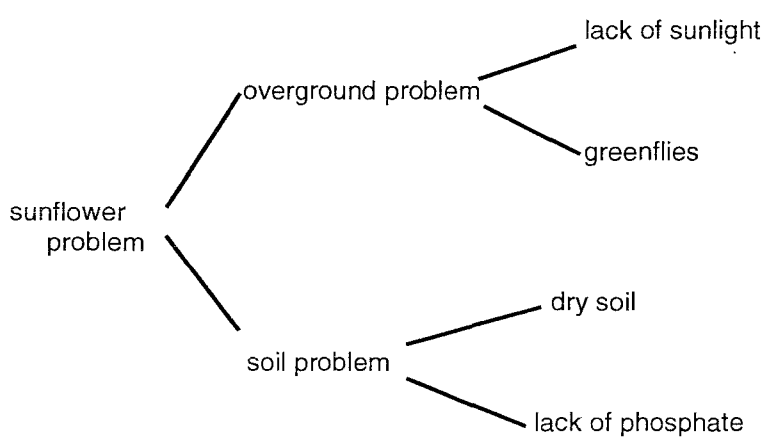
III. Types of behaviour	
Autonomy	
Responsiveness	
Pro-activeness	
Social behaviour	
Own adaptation and learning	

Appendix 2: Sunflower Problem

(Een Nederlandse vertaling van Sunflower Problem is niet beschikbaar.)

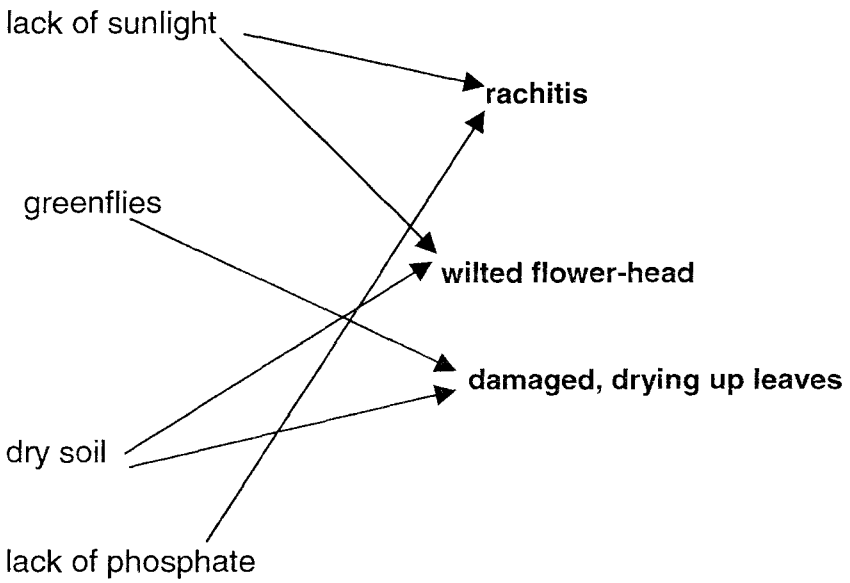
Consider the following situation, which involves two agents, an owner of a sunflower and a specialist in growing sunflowers. The owner observes that his sunflower is in a poor shape. As he is not able to find out himself why it is so, he decides to call the specialist and ask him to find out the problem causing the current state of the sunflower.

Since the specialist has no possibility to observe the plant, he asks the owner to make certain observations and communicate them back to the specialist.
To determine the nature of the problem, the specialist 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. The specialist uses the following hierarchy (taxonomy) of the subproblems of sunflower problems, that he uses to efficiently order the assumptions he can make:



The specialist can instruct the owner to make observations on the state of the sunflower leaves (whether they are drying up and damaged), on the height of the plant (whether it is rachitic) and on the state of its flower-head (whether it is wilted).

The relations between causes and symptoms are depicted below:



If there is a lack of sunlight then the plant becomes small and stunted (rachitic) and its flower-head becomes wilted. If there are greenflies (bladluizen) they can cause brown, damaged leaves which eventually will dry up. Damaged, drying up leaves can be also a result of dry soil, and the wilted flower-head is often caused by dry soil too. Phosphate is an important chemical ingredient of soil: without it the sunflower becomes rachitic.

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

information type **world_meta_info**
 sorts WORLD_INFO_ELEMENT,
  INFO_ELEMENT
meta-descriptions
  world_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;

knowledge base assumption_evaluation_local_kbs

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 Owner

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
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>, <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

information type observation_result_info; /* object level */

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 4 Answersheet (1 out of 1)

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	