Student name: ______________________________

Student number: ______________________________

Faculteit der Exacte Wetenschappen

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

Vrije Universiteit Amsterdam **15 december 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 Alice. Alice is de assistent van een film director. 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 Alice'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 Alice. Alice is the assistant to a movie to director.
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.

Student name: ______________________________
Student number: ______________________________

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 Alice's 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 turbulence. Read Appendix 2 "Turbulence 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 partiële specificatie van Appendix 3. Neem aan dat de informatie types causes and symptoms de volgende specificatie hebben:

```
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

[assumed(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. Motiveer Uw antwoord.

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

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

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. Motivate your answer.

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 Appendix 4. Use the answer sheet and fill in your answer in Appendix 4. Motivate your answer.

Opgave / Question 4 (15 points)

Een Nederlandse vertaling van Question 4 is niet beschikbaar.

Consider a partial specification of a small reasoning component (see Appendix 5).
For the extents **any, every, all-p, any_new** specify the output produced by the component and motivate your answer.

Opgave 5 (10 points)

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. Motiveer Uw antwoord.

Student name: ____________________
Student number: ____________________

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. Motivate your answer.

Student name: ____________________
Student number: ____________________

Appendix 1A: Assistant of a Movie Director

Alice is an assistant of a movie director. Part of her duties is to recruit actors for new roles. To this end, she investigates the suitability of an actor to different types of roles, e.g. whether he/she is capable to play the role of a villain, a hero, etc. In this connection, Alice regularly looks for new actors on the Internet and also attends new plays in theatres watching actors perform on stage. She discusses the conclusions of her investigations with the director, and later on she communicates the director's decisions to the actors. The actors in their turn ask Alice's professional advice on their performance and make various requests to Alice such as copying scripts, etc. It is her duty to fulfill such requests.

Another part of her obligations is monitoring of the filming conditions such as lighting of the stage, sound effects, etc. She discusses her observations with the director, and after receiving his instructions Alice informs stage workers what must be improved.

Together with the director and the crew she does her best to produce a picture good enough to be nominated for a national film festival. Together with the director she is responsible for making a schedule, so that the motion picture is finished on time.
We asked the director about Alice's personal characteristics. Below is the interview with him.

I know Alice for five years. She is an energetic person committed to producing the best picture possible. So I need not check her work constantly and allow her freedom in her decisions.
She started with this job five years ago and was inexperienced. She has learned a lot during the time on this job and became an excellent assistant. She is indispensable to our group. Creative people like ourselves often have difficult features in their characters.
I have bad temperament and never want to go into details of any routine matter and my actors are capricious and demanding.
Alice often takes initiative to mediate our conflicts and always finds unexpected and effective solutions.
Alice is ambitious. In time she wants to be a movie director herself, so she studies my work carefully and this year she will negotiate with investors to get money for a small budget movie she is planning to shoot herself.

Student name: ______________________________
Student number: ______________________________

Appendix 1B: Answer sheet (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	

Student name: ____________________
Student number: ____________________

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	

Student name: ______________________________
Student number: ______________________________

Appendix 1B Answersheet (3 out of 3)

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

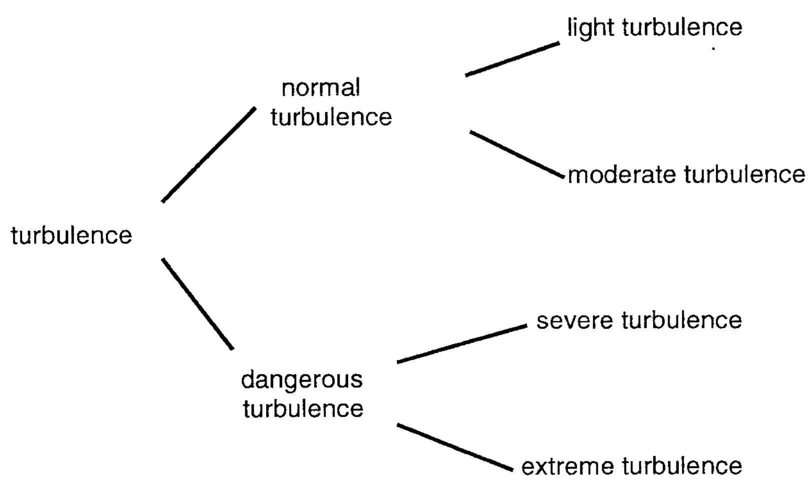
Student name: ____________________
Student number: ____________________

Appendix 2: Turbulence Problem

Many air travellers have experienced turbulence, an irregular air motion that often occurs unexpectedly. This sudden and vicious phenomenon is invisible to radars, and when it occurs the pilots must report the occurrence back to flight control for it might have most dramatic consequences for the crew and passengers. To diagnose the type of turbulence they use their knowledge of symptoms of different types of turbulence.

Suppose we have modelled an agent who would take over this duty.
To determine the type of the turbulence, our agent uses a line of reasoning modelled by the generic model for reasoning with and about assumptions (see Appendix 3). This 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.

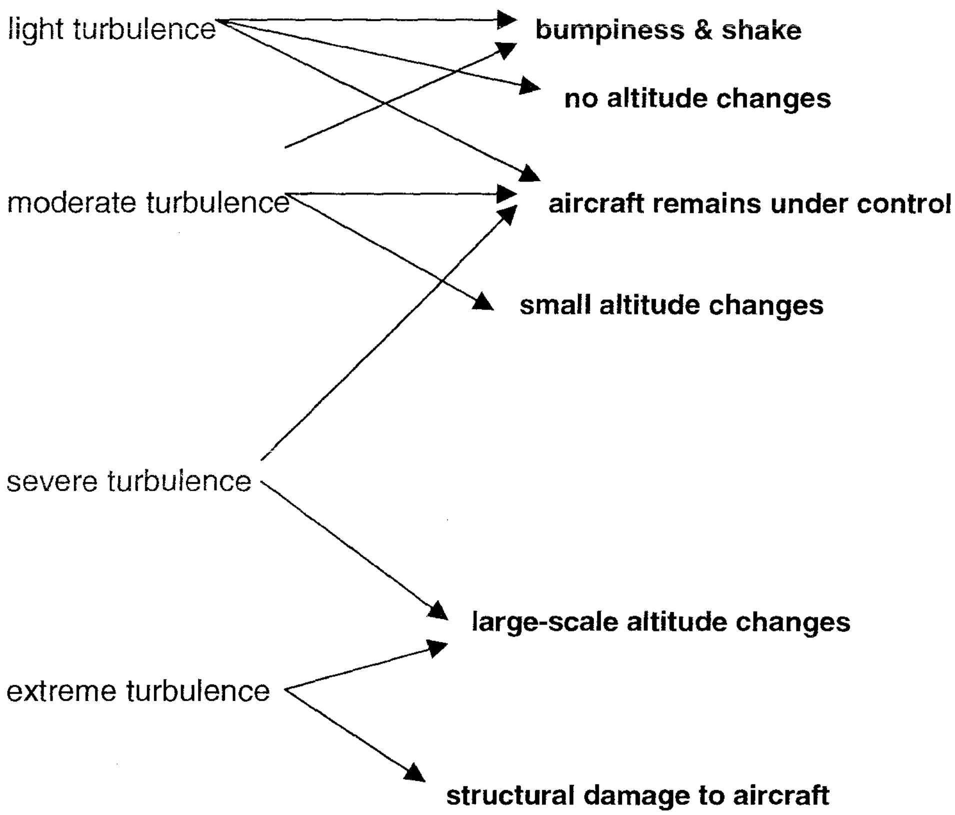
The agent uses the following hierarchy (taxonomy) of types of turbulence, that he uses to efficiently order the possible assumptions:



The relations between causes and symptoms are depicted below:

Student name: ____

Student number: ____



If light turbulence occurs, then the passengers experience light bumpiness and shake without noticeable changes in altitude, and the aircraft remains fully under the pilot's control. During moderate turbulence passengers and crew notice bumps and jolts and small changes in altitude are possible, and the aircraft is under control at all times. Severe turbulence causes large, abrupt changes in altitude, although the crew keeps the aircraft under control most of the time. This may cause injury to passengers but the aircraft itself experiences no damage.

In case of extreme turbulence, a very dangerous event, the aircraft is impossible to control. It is violently tossed about and this mostly results in structural damage to the aircraft.

Student name: ______________________________
Student number: ______________________________

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

Student name: ____________________
Student number: ____________________

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 type information

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

Student name: ____
Student number: ____

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

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
```

Student name: ______________________________
Student number: ______________________________

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

Student name: ______________________________
Student number: ______________________________

Appendix 4 Answersheet (1 out of 1)

Input (1)	[assumed(a, pos), predicted_for(b, pos, a, pos)]
Output after revision but before reasoning	[ ] /*given*/
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	

Student name: ____________________
Student number: ____________________

Appendix 5: A small reasoning component

1 Information types

The information types used throughout the system are:

```
information type input_C
        relations
                a,
                b,
                c;
end information type

information type output_C
        relations
                c,
                d,
                e;

end information type
```

2 Fragments of the specification of the component

The component is primitive. A short description is as follows.

The component comp_C

The interfaces are defined by:

input interface: the information type input_C;

output interface: the information type output_C;

The targets associated to task control focus determine_facts are:

target(determine_facts, X: OA, determine);

The initial extent is: all-p

The knowledge base contains the following knowledge elements:

```
if    a    then    c ;

if    b    then    d ;

if    c    then    e ;
```

all-p

input	output
[ ]	[ ]
[a]	
[b]	
[a, b, c]	

any

input	output
[ ]	[ ]
[a]	
[b]	
[a, b, c]	

every

input	output
[ ]	[ ]
[a]	
[b]	
[a, b, c]	

any_new

input	output
[ ]	[ ]
[a, b]	
[a, b, c]	