Faculteit der Exacte Wetenschappen	Tentamen Ontwerp van Multi-agentsystemen
Vrije Universiteit Amsterdam	20 april 2000

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

Normering:
Het **tentamencijfer** T is gelijk aan (het totaal behaalde punten voor de tentamenopgaven plus 10 punten bonus) gedeeld door 10.

Het **eindcijfer** voor het hoorcollege Ontwerp van Multi-agentsystemen wordt als volgt berekend:

Eindcijfer = (T + H + P) / 3,

waarbij
T = tentamencijfer,
H = cijfer huiswerkopgaven,
P = cijfer voor 1 punts practisch werk.

Verzoek:
U wordt vriendelijk verzocht om voor Uw antwoorden op opgave 4 gebruik te maken van de speciale **antwoordvellen**.

U treft aan:
5 opgaven
appendices
3 antwoordvellen

Opgave 1 (20 punten)

Deze opgave bestaat uit twee onderdelen. Motiveer Uw antwoorden. Deze opgave gaat over het spel Memory (zie Appendix 1).

Opgave 1a (10 punten)

Beschrijf in je eigen woorden twee verschillende strategieën voor het spelen van Memory.

Volgens welke strategie zou jij graag willen dat een software agent het spel Memory speelt?

Motiveer je antwoord.

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 het spel Memory zou kunnen spelen. 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.

Opgave 2 (10 punten)

Bestudeer de grafische specificaties van informatie typen in Appendix 2. Geef de textuele specificaties van de informatie typen maintenance info on world, maintenance on world, world meta-info, en truth indication.

Opgave 3 (20 punten)

Het onderwerp van deze opgave betreft informatietoestanden en redeneren. Bestudeer de partiële specificatie van Appendix 3. Gegeven is de object level public information state S van 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)]
```

Opgave 3a (8 punten)

Geef een informatietoestand S' die S verfijnt en bovendien gesloten en consistent is ten opzichte van de kennisbank van component mouse_a.

Opgave 3b (12 punten)

- Motiveer dat S' een verfijning (refinement) is van S (4 punten).
- Motiveer dat S' gesloten (closed) is ten opzichte van de kennisbank van component mouse_a (4 punten).
- Motiveer dat S' consistent is ten opzichte van de kennisbank van component mouse_a (4 punten).

Opgave 4 (25 punten)

Het onderwerp van deze opgave betreft informatietoestanden, revisie en links. Bestudeer de partiële specificatie van Appendix 4.

Opgave 4a (10 punten)

Neem aan dat de object level **input** informatietoestand van component maintain current information de toestand S is.

S = [bird(tweety)]

Bestudeer de volgende drie informatietoestanden:

M1 = [true(bird(tweety)), not known(penguin(tweety)), not known (can_fly(tweety))]

M2 = [true(bird(tweety)), not false(bird(tweety)), known(bird(tweety)),
not true(penguin(tweety)), false(penguin(tweety)), known(penguin(tweety)),
not true(can_fly(tweety)), false(can_fly(tweety)), known(can_fly(tweety))]

M2 = [true(bird(tweety)), not false(bird(tweety)), known(bird(tweety)),
not true(penguin(tweety)), not false(penguin(tweety)), not known(penguin(tweety)),
not true(can_fly(tweety)), not false(can_fly(tweety)), not known(can_fly(tweety))]

- (6 punten) Geef voor de volgende paren van informatietoestanden aan of ze level coherent zijn: paar (S, M1), paar (S, M2), paar (S, M3).
- (4 punten) Leg voor elk paar dat niet coherent is uit waarom dit paar niet coherent is.

object level output van component maintain current information
bird(tweety): unknown
penguin(tweety): true
can_fly(tweety): false

Tabel 1

Opgave 4b (8 punten)

In dit onderdeel wordt gekeken naar het effect van link-executie op informatie-toestanden. Stel dat voor executie van de link epistemic information (zie Appendix 4) de object level output informatie toestand van component maintain current information is als in Tabel **1**.

- Geef de **output meta**-level informatietoestand van component maintain current information na uitvoering van de upward reflection en vlak voor executie van de link epistemic information (4 punten).
- Geef de **input object** level informatietoestand van component assumption determination na executie van de link epistemic information (4 punten).

object level input van component maintain current information	meta level input van component maintain current information	object level output van component assumption determination
bird(tweety): true	assumption(bird(tweety), pos): true	possible_assumption(bird(tweety), pos): unknown
penguin(tweety): unknown	assumption(bird(tweety), neg): false	possible_assumption(bird(tweety), neg): unknown
can_fly(tweety): unknown	assumption(penguin(tweety), pos): false	possible_assumption(penguin(tweety), pos): unknown
	assumption(penguin(tweety), neg): false	possible_assumption(penguin(tweety), neg): unknown
	assumption(can_fly(tweety), pos): false	possible_assumption(can_fly (tweety), pos): true
	assumption(can_fly(tweety), neg): false	possible_assumption(can_fly (tweety), neg): unknown

Tabel 2

Opgave 4c (7 punten)

In dit onderdeel wordt gekeken naar het effect van link-executie op informatietoestanden. Stel dat voor executie van de link assumptions (zie Appendix 4) de object en meta-level input informatietoestanden van component maintain current information en de object level output informatietoestand van component assumption determination zijn als in Tabel **2**.

- Geef de **input meta**-level informatietoestand van component maintain current information na executie van de link assumptions (4 punten).
- Geef de **input object** level informatietoestand van component maintain current information na executie van de link assumptions (en na uitvoering van de downward reflection) (3 punten).

Opgave 5 (15 punten)

Beschouw het domein van tafeldekken voor vier personen, zie Figuur 2. 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.

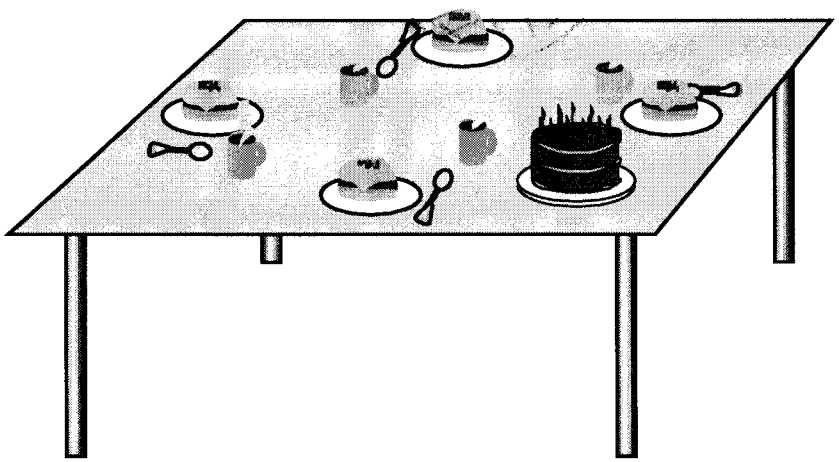


Figure 2

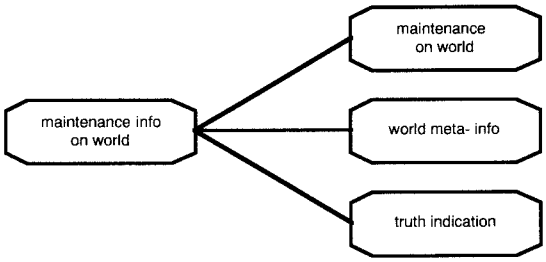
Appendix 1 Memory

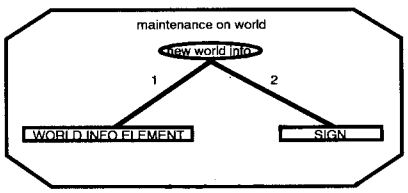
Memory is a card game with an even number of cards. For every card in the deck there is exactly one other card in the deck that has the same picture on it. The cards are spread out face down on the table. At least two players participate in the game. During his turn, the player is allowed to to subsequently look at two cards. If the second card has the same picture as the first card, both cards are removed from the gave, the player gets a point, and the player can again look at two cards, etcetera. If the second card differs from the first, the player's turn ends and the next player's turn begins.

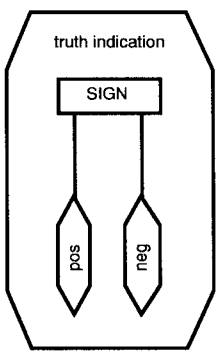
Although these rules are simple, the game requires a good memory, and a good strategy in order to score well.

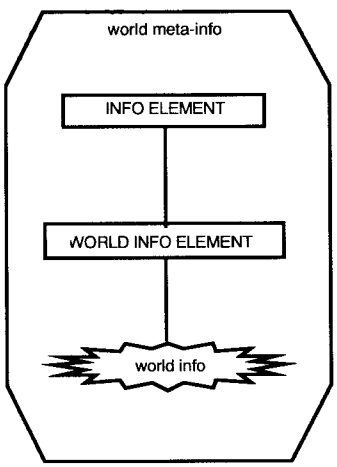
Appendix 2

Grafische specificaties van de informatie typen maintenance info on world, maintenance on world, world meta-info, en truth indication.





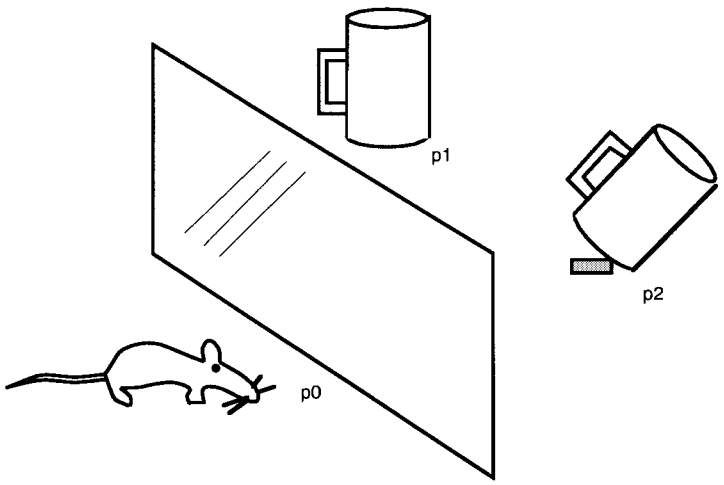




Appendix 3 Mouse A

3.1 Problem Description

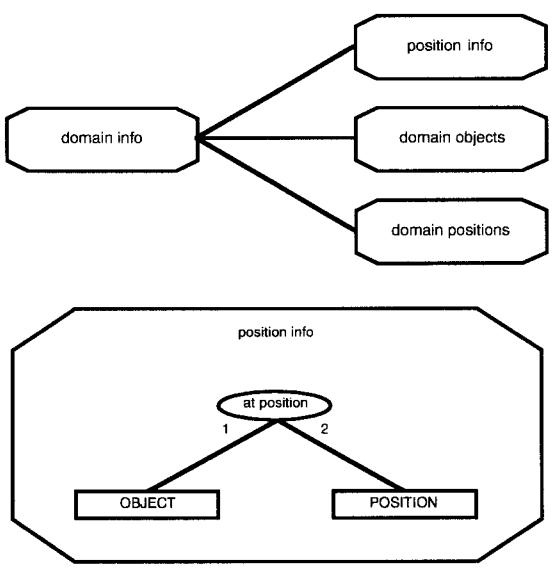
Separated by a transparent screen (a window, at position p0), at each of two positions p1 and p2 a cup (upside down) and/or a piece of food can be placed. At some moment (with variable delay) the screen is raised, and the mouse is free to go to any position. A genuine mouse is known to go to food and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

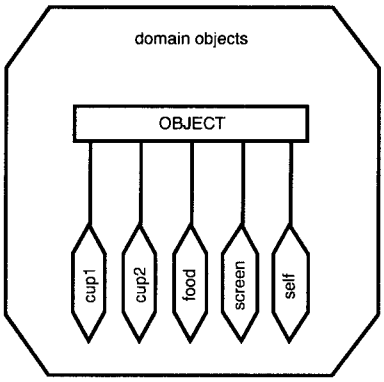


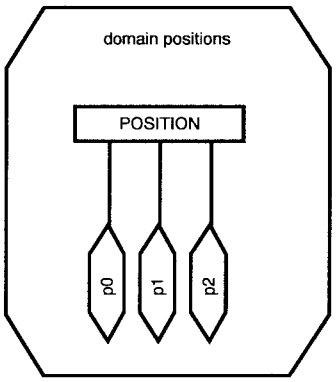
In sections 3.2 and 3.3 of this Appendix a partial specification can be found of the artificial mouse.

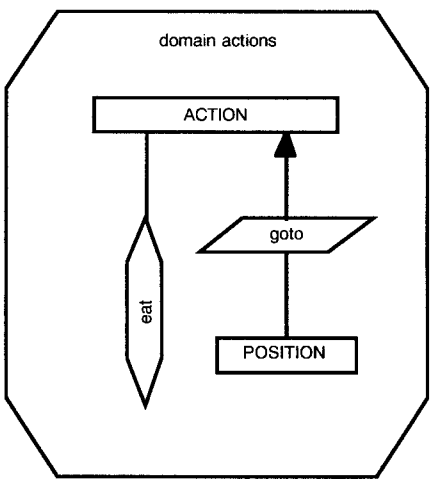
3.2 Informatie typen

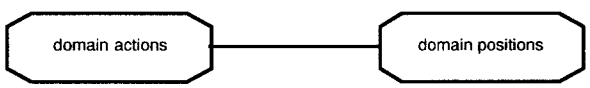
De informatie typen die in het hele systeem gebruikt worden zijn:











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

information type action_info
        information types       actions_to_be_performed,
                                domain_actions;
end information type
```

3.3 Fragmenten van specificatie van de component

De component is primitief en wordt hier kort beschreven.

De component mouse_a

De interfaces worden gedefiniëerd door:
input interface: de informatietypen observation_result_info;
output interface: het informatietype action_info;

De targets bij task control focus determine_action zijn:
target(determine_action, to_be_performed(X: ACTION), confirm);

De initial extent is: all-p

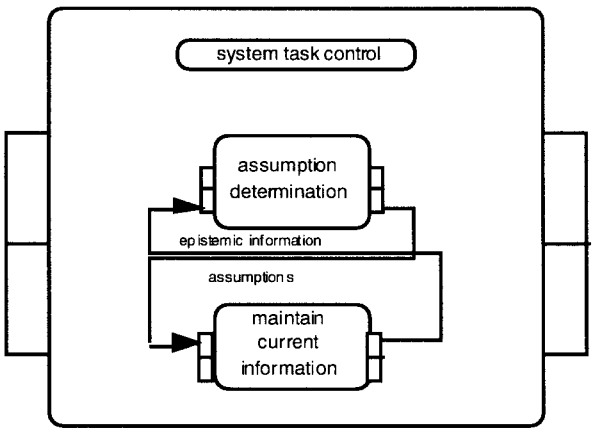
De kennisbank is:

```
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)
```

Appendix 4
Informatietoestanden, links en revisie

In default reasoning, (default) assumptions are determined based on specific type of knowledge, called *default rules*. Default reasoning can be modelled as described in this part of the appendix. An overview of the system is given in Figure A.1.



Figuur A.1

4.1 Informatietypen

De informatietypen die in het hele systeem gebruikt worden zijn:

```
information type          domain_info
        sort                                    ANIMAL;
        objects           tweety        :       ANIMAL;
        relations
                          bird,
                          penguin,
                          can_fly       :       ANIMAL;
 end information type

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

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

```
information type          known_info
        information types
                          domain_meta_info,
                          truth_indication;
        relations         known_to_hold   :     INFO_ELEMENT * SIGN;
end information type

information type          possible_assumptions
        information types
                          domain_meta_info,
                          truth_indication;
        relations         possible_assumption:  INFO_ELEMENT * SIGN;
end information type
```

4.2 Fragmenten van specificatie van de componenten

De componenten zijn primitief en worden hier kort beschreven.

De component assumption_determination

De interfaces worden gedefiniëerd door:

input interface:	het informatietype	known_info;
output interface:	het informatietype	possible_assumptions;

De targets bij task control focus bepaal zijn:

```
target(bepaal, X: OA, determine);
```

De initial extent is: all-p

De kennisbank is:

```
if known_to_hold(bird(tweety), pos)
   and not known_to_hold(penguin(tweety), pos)
then possible_assumption(can_fly(tweety), pos)
```

De component maintain_current_information

De interfaces worden gedefiniëerd door:

input interface:	het informatietype	domain_info;
output interface:	het informatietype	domain_info;

De targets bij task control focus maintain zijn:

```
target(maintain, X: OA, determine);
```

De initial extent is: all-p

4.3 Links

De links die in het systeem gebruikt worden staan hier onder.

```
private link epistemic_information: epistemic-object
        domain maintain_current_information
                level mci_meta_level
                information types epistemic_domain_info;
        co-domain assumption_determination
                level ad_object_level
                information types known_info;
        sort links      (IOA, INFO_ELEMENT),
                        (ANIMAL, ANIMAL);
        object links identity
atom links
        (true(X:IOA), known_to_hold(X:INFO_ELEMENT, pos))       :
                < <true, true>, <false, false> >
        (false(X:IOA), known_to_hold(X:INFO_ELEMENT, neg))      :
                < <true, true>, <false, false> >
 end link

private link assumptions: epistemic-object
        domain assumption_determination
                level ad_object_level
                information types possible_assumptions;
        co-domain maintain_current_information
                level mci_meta_level
                information types assumption_domain_info;
        sort links      (INFO_ELEMENT, IOA),
                        (ANIMAL, ANIMAL);
        object links identity
atom links
        (possible_assumption(X:INFO_ELEMENT, S:SIGN), assumption(X:IOA, S:SIGN)) :
                < <true, true> >
 end link
```

Antwoordvellen

Faculteit der Exacte Wetenschappen	Tentamen Ontwerp van Multi-agentsystemen
Vrije Universiteit Amsterdam	21 april 2000

Studentnaam en nummer: _____

Antwoord 4a

Paar van informatie toestanden	Level coherent ja of nee
(S, M1)	
(S, M2)	
(S, M3)	

Tabel A1

Motivatie antwoord 4a:

Studentnaam en nummer: _____

Antwoord 4b

meta level output of component maintain current information after upward reflection and before execution		object level input of component assumption determination after execution	
atom	truth-value	atom	truth-value

Tabel A2

Studentnaam en nummer: _____

Antwoord 4c

meta level input after execution

atom	truth-value
assumption(bird(tweety), pos)	
assumption(bird(tweety), neg)	
assumption(penguin(tweety), pos)	
assumption(penguin(tweety), neg)	
assumption(can_fly(tweety), pos)	
assumption(can_fly(tweety), neg)	

Tabel A3

object level input

after execution and downward reflection

atom	truth-value
bird(tweety)	
penguin(tweety)	
can_fly(tweety)	

Tabel A4