Hertoets Intelligente Systemen (B3IS)

- Het gebruik van boeken, aantekeningen, rekenmachines of andere bronnen is **niet toegestaan**.
- Elke vraag is 9 punten waard, dus je kunt 45 punten verdienen.
- **Antwoord bondig!** Ik trek punten af voor overbodige uitweidingen.

1. **AI, Agenten**
   (a) Russell en Norvig verdelen definities van 'kunstmatige intelligentie' in vier categorieën: Kunstmatig intelligent zijn systemen die *menselijk* of *rationeel*, kunnen *denken* of *handelen*.
      i. (2 punten) Geef een voorbeeld van een systeem dat *rationeel denkt* en een voorbeeld van een systeem dat *rationeel handelt*. Leg uit hoe ze rationeel denken danwel handelen.
      ii. (2 punten) Beschrijf een verschil en een overeenkomst tussen deze twee typen systemen. Verklaar beide.
      iii. (1 punt) Om welke van deze vier soorten AI draait het vak *Intelligente Systemen*? Verklaar je antwoord.
   (b) (4 punten) Geef een PEAS-beschrijving van een robotstofzuiger.

2. **Logica**
   (a) Sommige inferentie-algoritmen zijn gebaseerd op de resolutieregel.
      i. (1 punt) Geef de resolutieregel. Je mag je beperken tot de propositielogische variant.
      ii. (2 punten) Beargumenteer waarom de resolutieregel *correct* is ('*sound*' in het Engels), dus waarom een conclusie die je ermee kunt trekken uit bepaalde premissen, ook inderdaad het *logisch gevolg* is van die premissen.
   (b) (6 punten) Gebruik de resolutiemethode om vast te stellen of de volgende uitspraak waar is.

$$(P(a) \wedge Q(c)), \forall x(P(x) \rightarrow R(x,a)) \models (\exists y R(y,y) \vee \neg \exists z Q(z)).$$

   Geef duidelijk antwoord ('waar' of 'niet waar'), en leg ook uit welke stappen je onderneemt om tot je antwoord te komen, en hoe je antwoord volgt uit je methode.

3. **Planning**
   (a) (1 punt) Als een state-space search algoritme een plan vindt, hoe ziet zo'n plan er dan uit?
   (b) (1 punt) Noem een voordeel van een partial-order plan boven zo'n plan dat door een state-space search algoritme gevonden wordt.
   (c) (3 punten) Een partial-order plan (POP) is het resultaat van een zoekproces in een *andere* gerichte graaf dan die van state-space search, namelijk een plan-space graaf. Wat zijn de knopen ('nodes') in deze graaf? Maak een onderscheid tussen (i) de eerste knoop (waar geen kanten ('edges') naartoe lopen), (ii) knopen waar kanten naartoe én vandaan lopen, en (iii) knopen waar alleen kanten naartoe lopen.
   (d) We willen een schedule maken voor het uitvoeren van de activiteiten in een partial-order plan, waarbij elke activiteit een zekere hoeveelheid tijd kost.
      i. (2 punten) (Kijk vast naar vraag ii.) Geef een concrete instantie van zo'n—uitsluitend temporeel—roosteringsprobleem (een POP waarin de activiteiten een zekere duur hebben dus), bepaal hierin het kritieke pad, en leid de 'earliest start' en 'latest start' schedules af.
      ii. (2 punten) Breid je probleeminstantie van vraag i uit met één of meerdere *resources* die door activiteiten gebruikt worden. Zorg ervoor dat deze instantie illustreert dat, wanneer er resources worden toegevoegd, het earliest start schedule van de kritieke pad methode ongeldig kan worden.

4. **Logisch leren**

   Stel dat we met het Version Space leeralgoritme een predicaat willen leren dat beschrijft of een object een **rode bal** is. We gebruiken het predicaat *Object*(x, y, z), waar x het formaat is (groot of klein), y de kleur (rood, wit of blauw) en z de vorm (bal, blok of kubus), en we willen dat dit predicaat waar is als een voorbeeld positief is, en onwaar als een voorbeeld negatief is. De voorbeelden zijn natuurlijk gelabeld als positief of negatief. Ons doelpredicaat is een atoom (dus geen complexe formule met connectieven), waarin eventuele variabelen universeel gekwantificeerd zijn. We kunnen dan *generaliseren* door in een predicaat een constante te vervangen door een variabele, en *specialiseren* door het omgekeerde te doen: *Priem*(x) generaliseert *Priem*(17), en *Student*(*ewoud*) specialiseert *Student*(x).

   (a) (2 punten) Noem twee *andere* manieren om een (complexe) predicaatlogische formule te generaliseren. (Nota bene: we gaan deze manieren hieronder *niet* gebruiken.)

   (b) We initialiseren de set G in het Version Space leeralgoritme als het meest algemene predicaat, dus $G_{\text{init}} = \{\textit{Object}(x, y, z)\}$, en we initialiseren de set S als $S_{\text{init}} = \{\}$. We geven het algoritme nu achtereenvolgens vier voorbeelden, waarna het algoritme zal zijn geconvergeerd, want dan geldt $G = S = \{\ldots\}$ (niet leeg). Het is handig om alvast te bedenken op welk predicaat het algoritme zal uitkomen. In dit geval bevatten zowel G als S op geen moment meer dan één atoom.

      i. (1 punt) Het eerste voorbeeld is *Object*(*klein, rood, bal*). Moeten we dit labelen als positief of als negatief? Leg uit.

      ii. (2 punten) Welke verzameling (G of S) wordt aangepast, en op welke manier, als gevolg van het verwerken van dit eerste voorbeeld? Verklaar je antwoord. Geef ook de resulterende verzamelingen G en S na deze eerste iteratie.

      iii. (1 punt) Het tweede voorbeeld is *Object*(*klein, blauw, bal*). Leg uit of dit voorbeeld een false positive of false negative is, en voor *welk element* van G of S het dat is.

      iv. (1 punt) Welke aanpassing(en) worden er gedaan in de verzamelingen G en/of S, als gevolg van het verwerken van dit tweede voorbeeld?

      v. (2 punten) Na deze twee voorbeelden volgen nog twee voorbeelden, *Object*(*groot, rood, bal*) en *Object*(*groot, rood, kubus*). Beschrijf de aanpassingen die er in de verzamelingen G en/of S worden gedaan bij het verwerken van elk van deze voorbeelden. Geef de verzamelingen G en S na het verwerken van elk van de voorbeelden.

5. **Prolog**

   (a) Geef voor elk van de volgende queries het antwoord dat Prolog geeft. Geef ook eventuele substituties als Prolog die doet. Verklaar elk antwoord.

      i. (1 punt) `?- tomas = 'Tomas'.`

      ii. (1 punt) `?- Tomas = 'Tomas'.`

      iii. (1 punt) `?- 2 = '2'.`

      iv. (1 punt) `?- 2 is 1+1.`

      v. (1 punt) `?- Twee = 1+1.`

      vi. (1 punt) `?- Twee is 1+1.`

   (b) (3 punten) Schrijf Prolog code die het predicaat `sorted/1` definieert, zodat `sorted(L)` waar is als `L` een lijst is met gehele getallen die in oplopende volgorde staan.

Einde tentamenopgaven