Tentamen Intelligente Systemen (B3IS)

- Het gebruik van boeken, aantekeningen, rekenmachines of andere bronnen is **niet toegestaan**.
- Je kunt 45 punten verdienen. Je cijfer is $1 + \frac{\text{aantal punten}}{5}$.
- **Antwoord bondig!** Ik trek punten af voor overbodige uitweidingen.

1. **Logica**
   (a) i. (3 punten) Maak een waarheidstabel om te onderzoeken of de volgende uitspraak waar is.

$$((p \wedge q) \rightarrow \neg r), (p \vee \neg q), (\neg q \rightarrow p) \models \neg r.$$

   ii. (1 punt) Is de uitspraak waar?
   iii. (1 punt) Leg uit hoe je antwoord bij ii volgt uit je waarheidstabel van vraag i. Nummer, indien gewenst, de kolommen en rijen in de waarheidstabel en verwijs naar deze nummers in je uitleg.

   (b) (4 punten) Gegeven is de volgende verzameling Γ met daarin zes disjuncties, die het resultaat zijn van herschrijven naar CNV, Skolemiseren, en verwijderen van universele kwantoren.

$$\begin{aligned}\Gamma = \{&C(a), \neg C(w) \vee D(w), \neg L(v,y) \vee \neg D(z) \vee \neg K(y,z),\\ &L(g(b),b) \vee \neg D(f(b)), L(g(x),x) \vee D(f(x)), K(b,a) \vee K(c,a)\}\end{aligned}$$

   Voor het gemak zijn de variabelen in verschillende disjuncties al 'standardized apart', dus elke variabele komt maar in één disjunctie voor: $v, \ldots, z$ zijn (universeel gekwantificeerde) variabelen, a, b en c zijn constanten, en f en g zijn functiesymbolen. Geef een resolutie-afleiding die bewijst dat $\Gamma \models K(c,a)$. Tip: Zet alle benodigde formules genummerd *onder elkaar* (dus *niet naast elkaar*), en geef dan steeds een volgende (genummerde) formule die je met de resolutie-regel afleidt uit twee voorgaande formules. Geef aan uit *welke* twee formules je een formule afleidt, en welke substitutie je hebt gebruikt om literalen te laten matchen. Als je factoring doet, geef dat dan ook aan.

2. **Classical Planning**
   (a) Als we PDDL gebruiken om een planningsprobleem te specificeren, resulteert dit in een bepaalde gerichte graaf waarin state-space search planningsalgoritmen (forward of backward) een plan zoeken.
      i. (1 punt) Wat is een knoop in zo'n graaf?
      ii. (1 punt) Wat representeert een kant (s,t) van knoop s naar knoop t in zo'n graaf?
      iii. (2 punten) Als een state-space search algoritme een plan vindt, hoe ziet dat er dan uit?

   (b) Een partial-order plan (POP) is een partiële ordening van activiteiten, die het resultaat is van een zoekproces in een *andere* gerichte graaf dan die van state-space search.
      i. (1 punt) Wat is het voordeel van een POP boven een plan dat state-space search vindt?
      ii. (2 punten) Beschrijf nauwkeurig hoe de eerste knoop in deze graaf, waar het zoekproces mee begint, eruit ziet.
      iii. (2 punten) Als het PO-planningsalgoritme een bepaalde knoop s in deze graaf evalueert, wat kenmerkt dan de knopen waar naartoe vanaf s kanten lopen?

3. **Plannen met resources**
   (a) (1 punt) Beschouw een partial-order plan voor het uitvoeren van activiteiten die een zekere hoeveelheid tijd kosten. Als je zo'n POP als graaf weergeeft, hoe vind je daarin dan een kritiek pad?
   (b) (1 punt) Wat is er 'kritiek' aan een kritiek pad?
   (c) (3 punten) (Kijk vast naar vraag d!) Geef een concrete instantie van zo'n (uitsluitend temporeel) roosteringsprobleem, bepaal het kritieke pad, en leidt de 'earliest start' en 'latest start' schedules af.
   (d) (2 punten) Breid je probleeminstantie van vraag c 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.

(e) (2 punten) Beschrijf een methode die je kunt gebruiken om (oplosbare) instanties van dit Resource-Constrained Project Scheduling Problem toch op te lossen—dat hoeft niet met de kortste makespan, en ook niet in een polynomiale hoeveelheid tijd. Beargumenteer wel hoe je methode een oplosbare instantie inderdaad oplost.

4. **Kennis leren**

Bij inductief leren van kennis uitgedrukt in logica, zoeken we in een ruimte $\mathcal{H} = \{h_1, \ldots, h_n\}$ van hypothesen, naar (bijvoorbeeld) een definitie van een predicaat Goal(x). (In het voorbeeld over wel of niet op een tafeltje wachten was dit het predicaat WillWait(x): we willen uit de attributen van situatie x leren of iemand wel of niet wacht in situatie x.) In het bijzonder zoeken we een definitie die alle reeds bekende examples correct classificeert, en bovendien nieuwe examples goed voorspelt. Elke kandidaat-hypothese h_j heeft de vorm van een logische formule die $C_j(x)$ heet—dus de rij van 5 tekens $C_j(x)$ is niet zélf een formule—en we zoeken een formule zodat

$$\forall x \in \mathcal{X} \quad (\text{Goal}(x) \leftrightarrow C_j(x))$$

waar is. Hierin is $\mathcal{X}$ de verzameling examples, Goal(x) de waarde van het te leren predicaat in example x, en $C_j(x)$ een formule over de attributen van example x, bijvoorbeeld een disjunctie van conjuncties over de attributen van situatie x.

(a) (2 punten) Beschrijf in termen van de uiteenzetting hierboven nauwkeurig wat een false positive example x_p en een false negative example x_n voor een zekere hypothese genaamd h_3 zijn. Betrek in je antwoorden ook de bi-implicatie hierboven.

(b) (2 punten) Het version-space leeralgoritme houdt twee verzamelingen van hypothesen bij: De G-set van meest algemene, en S-set van meest specifieke hypothesen. Hoe initialiseert het algoritme de G-set en de S-set voordat het examples gaat evalueren? Verklaar deze initialisatie.

(c) (3 punten) Wat doet het version-space leeralgoritme als het een example te verwerken krijgt dat een false negative blijkt te zijn voor hypothese S_i in de S-set? Verklaar je antwoord, en gebruik daarin ook de notie van de 'extensie' van een hypothese.

(d) (2 punten) In het algemeen is het doel dus een hypothese te vinden zodat uit de 'Hypothese' en de 'Beschrijvingen' van examples in termen van attributen, de 'Classificaties' van de examples *logisch volgen*. We kunnen hierbij op verschillende manieren van 'Achtergrondkennis' gebruik maken. Schrijf één of meer entailment relaties ($\ldots \models \ldots$), waarin je de vier genoemde concepten aan elkaar relateert, die duidelijk maken hoe van achtergrondkennis gebruik wordt gemaakt in algoritmen voor Knowledge-Based Inductive Learning—die bijvoorbeeld in Inductive Logic Programming worden gebruikt.

5. **Prolog**

(a) (4 punten) Beschouw de volgende Prolog code, die het predicaat `d/2` definieert. Voor welke combinaties van invoerwaarden `X` en `Y` is dit predicaat waar? Met andere woorden, wat moet de relatie tussen `X` en `Y` zijn zodat `d(X,Y)` slaagt?

```
d([],[]).
d([X],[X]).
d([X,X|T],Y) :- d([X|T],Y).
d([X,Y|T],[X|Z]) :- X \= Y, d([Y|T],Z).
```

(b) (5 punten) Schrijf Prolog code die het predicaat `range/3` definieert, zodat `range(X,Y,L)` waar is als `X` en `Y` gehele getallen zijn en `L` een lijst is met alle gehele getallen tussen `X` en `Y` (inclusief `X` en `Y`), dus als `L = [X,...,Y]`. Dit moet werken voor *alle* paren gehele getallen `X` en `Y`.

Einde tentamenopgaven