Intro to agents and agent-tools

1. Volg de tutorial en omschrijf daarna in één paragraaf wat deze tool anders maakt dan andere programmeertalen, wat zijn de voor- en nadelen? Leg ook uit waarom je programma wel of niet agent-based is.

   Netlogo is een programmeertaal die ontwikkeld is voor agent based modeling. Er worden agents in de vorm van turtles, patches, links en observer gemaakt. De implementatie talen die gebruikt werden om Netlogo te ontwerpen zijn Scala en Java. Netlogo wordt vaak door studenten, docenten en mensen die weinig/geen programmeer ervaring hebben. Dat komt omdat deze tool voor meerdere doelgroepen ontworpen is. Het is een prima tool die wereldwijd gebruikt wordt bij de modeleren van agents in verschillende situaties. Voor het gebruik van Netlogo heb je niet zo veel kennis nodig in vergelijking met de andere gegeven tools in de opdracht. In Netlogo kan je zowel 2D als 3D model maken.

   Voordelen:
   - Makkelijke programmeertaal om te leren.
   - Resultaten zijn eenvoudig te exporteren.
   - Simpele en makkelijke visualisatie.
   - Veel voorbeelden zijn op internet beschikbaar.

   Nadelen:

   - Het is een simpele programmeertaal, waardoor je moeilijke complexe/uitgebreide code kan schrijven.
   - Het is geen bekende programmeertaal waar je niet heel ingewikkelde visualisatie kan maken.
   - De gui ziet niet heel up to date uit.

   Voor deze opdracht heb ik een ziekteverspreiding simulatie gemaakt. Het doel van de simulatie is om te laten zien hoe snel een ziekte verspreid kan worden op basis van verschillende elementen zoals: het houden van 1,5 m afstand, de kans dat mensen elkaar kunnen besmetten, de incubatietijd van 5 dagen, hoe lang iemand ziek kan blijven en de sterfcijfers van de zieke patiënten.
   Naast de linkje die op canvas staan heb ik drie andere bronnen gebruikt, zoals je hieronder kunt zien:

   - *http://ccl.northwestern.edu/netlogo/models/SpreadofDisease*
   - *https://www.youtube.com/watch?v=9MdaR0Y-YEk&ab_channel=sagism*
   - *http://www.netlogoweb.org/launch#http://ccl.northwestern.edu/netlogo/models/models/Sample%20Models/Biology/Virus.nlogo*

2. We definiëren een staat en drie functies waarmee we een stateful agent abstract omschrijven
   - Initiële staat: In mijn tutorial heeft elke agent een kleur, een positie die random aan de agent wordt gegeven, en een beweging snelheid.

- See: In de functie ‘to go’ kan een agent de omgeving van 1 radius zien(een cirkel om zich heen). Wat deze omgeving ontoegankelijk maakt. In de see maakt een agent een perception van of er andere agents om zich heen zijn.

- Act: In de functie ‘to go’ kan op basis van het kleurtje van de agent zelf en de agent die ernaast staat, een andere kans die in de simulatie gebruikt worden, kan de agent kiezen om een andere kleurtje te krijgen. Zo kan er bijvoorbeeld een agent die gezond is “wit” en naast een besmette agent “oranje” staat ook zelf besmet met de ziekte raken. Als er echter geen besmette agents naast die agent staan dan blijft de agent gezond “wit”.

- Update: De update functie zit ook in de ‘to go’ functie. De agent kan zich verplaatsen op basis van een berekening die in de code staat. Op basis van de nieuwe positie van de agent kan er gekozen worden of het kleurtje van de agent veranderd kan worden.

Hieronder zijn de genoemde functies in de formele notatie beschreven:

$$see : S \rightarrow P \qquad action : I \rightarrow A \qquad next : I \times P \rightarrow I$$

Aangezien de ziekteverspreiding model ook gebruik van deze functies maakt, kunnen we deze functies voor de simulatie definiëren.

action(s)= {
Be orange if s = in 1 radius a red or orange
Be red if s = orange and it has been longer than 5 days
Be violet if s = red, it has been longer than 10 days and don’t have a chance to recover
Be gray if s = violet, it has been longer than 10 days and have a chance to die
Be green otherwise
}

Aangenomen dat de set of S alle mogelijke staten bevat. Om het nog duidelijker te maken gaan we statements, de verzameling of s en de functie See ook definiëren.

Statements:

x= “the color is red”

y= “the color is orange”

z= “the color is violet”

a= “it has been 5 days”

b= “it has been 10 days”

c= “chance to recover”

d= “chance to die”

s = {s1, s2, …s6} =

{{x∨y, ¬ (y∧a), ¬ (x∧b∧(¬c)), ¬ (x∧b∧c), ¬(z∧b∧d), ¬ (z∧b∧(¬d))} ,

{ ¬ (x∨y) , y∧a, ¬ (x∧b∧(¬c)), ¬ (x∧b∧c), ¬(z∧b∧d), ¬ (z∧b∧(¬d))} ,

{¬ (x∨y) , ¬ (y∧a) , x∧b∧(¬c), ¬ (x∧b∧c), ¬(z∧b∧d), ¬ (z∧b∧(¬d))} ,

{¬ (x∨y) , ¬ (y∧a) , ¬(x∧b∧(¬c)), x∧b∧c, ¬(z∧b∧d), ¬ (z∧b∧(¬d))} ,

{¬ (x∨y) , ¬ (y∧a) , ¬(x∧b∧(¬c)), ¬ (x∧b∧c), ¬(z∧b∧d), ¬ (z∧b∧(¬d))} ,

{¬ (x∨y) , ¬ (y∧a) , ¬(x∧b∧(¬c)), ¬ (x∧b∧c), ¬(z∧b∧d), z∧b∧(¬d)}}

Percepts:

P1 = “Be orange”

P2 = “Be red”

P3 = “Be violet”

P4 = “Be green”

P5 = “Be gray”

Dus de *see* functie ziet als volgt uit:

see(s) = { P1 if s = s1; P2 if s = s2; P3 if s = s3; P4 if s = s4 or s = s6; P5 if s = s5 }

3. Beschrijf je omgeving op basis van de dichotomies die hier op pagina 6 beschreven staan, en licht toe (dus niet alleen termen opsommen):

- Accessible vs inaccessible
  De omgeving in de ziekteverspreiding model is **inaccessibel** omdat de agent alleen een radius om zich heen kan zien. Dus hij heeft niet alle informatie van de omgeving tot beschikking .

- Deterministic vs non-deterministic (stochastisch)
  We kunnen de resultaten van dit model moeilijk voorspellen aangezien de agents bewegen volgens een bepaalde regel waar random in staat. Daardoor kunnen we dit model als **stochastisch** model classificeren.

- Episodic vs non-episodic
  In deze simulatie kan een agent beslissen welke actie uitgevoerd moet worden op basis van de huidige situatie. Omdat ze niet naar de vorige of de toekomstige "episode", maakt dat deze simulatie **episodic**.

- Static vs dynamic
  Ondanks dat de agents allemaal blijven bewegen, wordt niets aan de "wereld" veranderd. Dus er kunnen verschillende resultaten uit de simulatie leiden. Daardoor kunnen we zeggen dat deze simulatie **static** is.

- Discrete vs continuous
  Deze simulatie vindt plaats op een open veld. Dus er zijn geen vaste stappen die uitgevoerd kunnen worden. In dit geval kunnen we deze simulatie als een **continuous** simulatie beschouwen.

4. Bedenk een voorbeeld waarbij minimaal 3 dichotomies precies tegenovergesteld zijn en beargumenteer waarom dit wel of niet van belang is om je simulatie nuttig te maken.

   - Accessible ipv inaccessible

     Als we de simulatie accessible maken dat als de agent op afstand kan zien dat hij dichterbij een oranje/rode agent komt en vervolgens de afstand van deze agent neemt, zou dat niet veel toegevoegde waarde hebben op de simulatie omdat je in het echt moeilijk van afstaand kan weten of iemand met bepaalde ziekte besmet is of niet.

   - Non-episodic ipv episodic

     Als we de simulatie non-episodic zouden maken, dan kunnen agents naar het verleden van andere agents kunnen kijken. Dus als ze bijvoorbeeld al weten dat er een agent eerder met de ziekte besmet geweest is, dan zou hij dichtbij die immuun agent kunnen komen. Alleen deze aanpassing heeft niet zo veel zin in de simulatie. Aangezien de immuun agents een andere kleur krijgen.

   - Non-deterministic ipv deterministic

     Als we de simulatie deterministic zouden maken, dan zouden de agents volgens een bepaalde 'route' moeten lopen. Dat voegt weinig aan de simulatie toe. Omdat in de echte wereld het gedrag van mensen moeilijk te voorspellen is.