1. Inleiding

De programmeertaal is gebaseerd voornamelijk op Python, maar is zo simple mogelijk gehouden. Met veel symbolen kun je min of meer gewoon doen wat je verwacht dat ze zouden doen. + telt op, & is een logische AND, enzovoor.

1. Syntax van de taal.

Nmr ::= Int | Name | Num+Num | Num\*Num | Num/Num

Con ::= Bool | not Con | Num<=Num | Con&Con | Con|Con

Cmd ::= wait; | Call Name Int; | if(Con){Pgm} | while(Con){Pgm} | roboCmd;

roboCmd ::= Turn\_left | Turn\_right | Forward | Stop

| Lamp Nmr Nmr Nmr Nmr | Light Name | Dist Name

Pgm ::= [Cmd]  
Bool ::= True | False  
Int ::= Integers  
Name ::= String, naam voor variabelen in de environment.

1. Semantiek van de taal

De taal op zich bestaat uit een series commando’s, Cmd, samengevoegd in blokken, Pgms, omgeven door { } haakjes. Voor de parser is het de bedoeling dat deze ingegeven worden zonder newlines, maar voor de overzichtelijkheid kan de taal met newlines geschreven worden die dan net voor het parsen verwijderd worden.

Nmr zijn numerieke warden – dit is ofwel een literal int, een Name (String) die verwijst naar een getalvariabele in de environment, of een uitdrukking. Uitdrukkingen worden omgeven door ronde haakjes ( ), zonder spaties. Nmr voorbeeld: “1+(2+3)”

Con zijn logische condities. Deze kunnen een booleaanse waarde, True of False aannemen, ofwel de vorm van een logische uitdrukking. Net zoals hierboven moeten deze in ronde haakjes gezet worden. De beschikbare logische uitdrukkingen zijn een logische AND met het teken ‘&’, een logische OR met het teken ‘|’, een logische not doormiddel van het word “not “, of een vergelijking van twee Nmr getallen met ‘<=’. D.m.v. ‘&’ en ‘not’ kunnen hiermee alle andere vergelijkingen gemaakt worden. Con voorbeeld: “True&(2<=3)”

Cmd zijn de commandos waaruit de taal opgebouwd zijn. Een cmd kan wait zijn, hetwelke niks doe ten gewoon wacht. “Call Name Int” voegt een getal met de waarde van Int aan de omgeving toe onder de naam Name. Een roboCmd stelt een commando voor de robot voor, verder uitgebreider uitgelegd. Deze drie varianten moeten afgesloten worden met een puntkomma. Het commando if(con){Pgm} evalueert de conditie die in de ‘if’ staat. Als deze conditie waar is, wordt het blok van commandos “Pgm” uitgevoerd. Hetzelfde geldt voor het while(con){Pgm} commando, maar hier wordt het blok herhaald zolang de conditie True is.

Voor roboCmd zijn er 7 mogelijke opties. Turn\_Left, Turn\_Right, Forward en Stop laten de robot respectievelijk naar links draaien, naar rechts draaien, vooruit rijden of stoppen. Lamp neemt vier Nmr argumenten – het eerste is het nummer van de lamp, en de volgende drie stellen de RGB waarden voor waarop deze moeten ingesteld worden. Light en Dist nemen elk een Name als argument mee, waarin het resultaat opgeslagen zal worden. Hierbij wordt voor Light een “0” gebruikt als beide sensoren zwart zijn, “1” voor de linkersensor wit, “2” voor de rechtersensor wit, en “3” voor beide sensoren wit.

Merk ook op dat het geheel van het inputprogramma op zich ook als programma behandeld wordt, en er dus ook { } haakjes rond moeten staan.

1. Voorbeelden

TBA

1. Implementatie:

De code voor de programmeertaal bestaat uit twee grote delen – de parse-module en de uitvoeringsmodule.

De parse-module is opgedeeld in parsers voor de verschillende datatypes. De parser-file voor elk datatype bevat de definitie van het datatype, een eventuele evaluator voor dit datatype alsook de parser voor dit datatype. De parsers bestaan op zich uit enkele kleine parser-modules, en worden samengehecht d.m.v. `mplus` uit monadplus om uiteindelijk een grote parser voor programmas te bekomen. De finale pgmParser parst geen commandos, maar programmas – het resultaat van de parse is een lijst van commandos die dan sequentieel door de uitvoeringsmodule uitgevoerd kunnen worden.  
Bij NmrParser is ook het type Env gedefinieerd – dit stelt een lijst voor van paren van names – strings – en de bijhorende getallen.

In de mainmethode worden eerst de eventuele newlines weggetrimd, en wordt de Mbot geopend. Daarna wordt het inputdocument gelezen en in zijn geheel aan de parser doorgegeven. Die maakt hiervan een lijst van commandos. Met behulp van de methode runPgm wordt deze lijst omgezet in een StateT Env IO (). Dit is een monad transformer die de State – onze omgeving – combineert met IO voor de robot. Een object hiervan stelt een state-transformatie voor die een state zal omzetten in de volgende. Hierbij is onze state telkens de lijst met variabelen die we bijhouden. Met behulp van de runStateT methode hierin kunnen we dan de IO operatie bovenhalen die het programma als geheel voorstelt. Elk commando wordt hierbij omgezet naar een StateT Env IO (), die dan met bind aaneengekoppeld kunnen worden. Uiteindelijk wordt de Mbot terug gesloten.

De monad transformer gaf echter problemen, en deze heb ik niet correct kunnen doorgronden. Hierdoor heb ik de uitvoeringsmodule niet af kunnen maken. Deze is echter wel helemaal uitgedacht, dus hieronder volgt de basisopbouw van de verschillende commandos.

Wait is simpel, en returnt gewoon een IO die niets doet en niets teruggeeft – of, eventueel, gewoon eventjes wacht.

“Call Name Int” is een commando dat enkel op de State moet werken. Het neemt de omgeving en voegt hieraan een nieuw element toe met de gegeven naam en waarde, en verwijdert eventueel het oude element met deze waarde. De IO voor dit commando doet niets.

If en While roepen beiden de evaluator voor logische expressies op om te evalueren of de meegegeven propositie True of False is. Als ze False is, gebeurt er niets. Als ze True is, dan worden de commandos in het Pgm blok dat bij de if/while hoort uitgevoerd. If stopt na deze uitvoering, While zal zichzelf terug oproepen, en zo blijven herhalen tot de conditie niet meer waar is. Deze twee commandos beinvloeden de State zelf niet en hebben zelf geen IO output, maar het resultaat dat ze geven zal de veranderingen aan de State en de IO hebben van de commandos die ze binnenin uitvoeren.

De roboCmd’s worden elk individueel geimplementeerd.

Turn\_Left heeft geen invloed op de State. De IO hiervan is sendcommand setMotor -255 255. De robot begint dan naar links te draaien.

Turn\_Right is analoog, maar met setMotor 255 -255 om naar rechts te draaien, Forward met setMotor 255 255 om vooruit te rijden, en Stop met setMotor 0 0 om te stoppen.

Lamp Nmr Nmr Nmr Nmr verandert de state niet en geeft als IO een sendCommand setRGB met de vier ints als argumenten.

Light Name heeft als IO het commando readLineFollower van de robot. Het resultaat hiervan wordt dan in een nummer omgezet en de State wordt aangepast door het toevoegen van de variabele Name met de waarde van de lightsensor, zoals hierboven beschreven.

Dist Name werkt hetzelfde als Light. De IO van dit commando is het commando readUltraSonic voor de robot, en die wordt dan analoog als bij light toegevoegd aan de Sate onder de meegegeven naam.

1. Conclusie

De programmas worden succesvol geparsed, maar de Monad Transformer is een integraal onderdeel van de daadwerkelijke uitvoering van de programmas, en zonder deze goed te verstaan kunnen de programmas niet goed uitgevoerd worden. De parser slaagt echter wel in het mooi ontleden van programmas, en de stappen die moeten genomen worden voor elk commando zijn duidelijk uitgetekend, dus eens het probleem met de monad transformer in orde is lijkt dit een mooie basis te zijn voor een werkende programmeertaal om de robot aan te sturen.