Talen en Automaten
case study, voor programmeertaal:

Jug

Door:
Rick Erkens s4100573
Sanne Boumans s3031926

1.Informele beschrijving

Jug is een programmeer taal gebaseerd op een bekend raadsel, namelijk het raadsel waarin mensen verteld wordt dat ze een aantal kannen met een bepaalde hoeveelheid water hebben en dan volgens een paar handelingen een andere kan moeten vullen.
Een direct voorbeeld van het raadsel is:

> *Je hebt toegang tot een onbeperkte hoeveelheid water. Je hebt twee lege containers, een van 7 liter, en een van 11 liter. Bij elke bewerking moet een van de twee containers helemaal vol of helemaal leeg zijn. Hoeveel bewerkingen zijn er nodig om een container te vullen met precies 6 liter water?*

Het idee achter de taal Jug is dat er 'water' over bepaalde 'jugs' verdeeld wordt om zo een algoritme te maken. Hier zijn de jugs plaatsen die aangemaakt worden om een integer (het water) vast te houden. Om dit idee te verduidelijken is het belangrijk om eerst de mogelijke functies van Jug uit te leggen.

De functies in Jug worden aangeroepen als:
functienaam (a,b);
In totaal heeft Jug 8 van deze functies:

```
jug(a,b);          -maakt een jug 'a' aan, met een capaciteit van 'b'. returns a (zowel a en b zijn int's)
fill(a);           -vult jug 'a' tot aan zijn capaciteit. returnt a
empty(a);          -maakt jug 'a' leeg. returnt a
pour(a,b);         -verplaatst de inhoud/ het water'water' van jug 'a' naar jug 'b'. Hij stopt voordat jug 'b'
                    overloopt en
echo(a);           -prints 'a'. returnt a
volume(a);         -returnt de inhoud/hoeveelheid 'water' (int) in jug 'a'
if_empty(a){};     -als jug 'a' leeg is doe {...}.
else{};            -komt na de if_empty functie. Dus als jug 'a' niet leeg is doe {...}
drain(a){};        -maakt jug 'a' leeg dmv. een loop {...}
```

Hier zie je dus dat elke jug (in dit geval jug 'a') een bepaalde capaciteit (in dit geval 'b') kan hebben en dat de jug gevuld kan worden tot aan deze capaciteit. Vervolgens kunnen de jugs leeg gemaakt worden en kan de inhoud van een jug naar een andere jug worden verplaatst.
Doordat Jug alleen met integers om kan gaan kan de taal alleen worden gebruikt voor algoritmes die met getallen werken.

Een aantal concrete voorbeelden hiervan zijn:

```
TEL 5 EN 4 BIJ ELKAAR OP:
jug(0,5);          -maakt jug '0' met een capaciteit van 5
jug(1,4);          -maakt jug '1' met een capaciteit van 4
jug(2,100);        -maakt jug '2' met een capaciteit van 100
fill(0);           -vult jug 0 tot aan zijn capaciteit (er zit nu 5 in jug 0)
fill(1);           -vult jug 1 tot aan zijn capaciteit (er zit nu 4 in jug 1)
pour(0,2); -verplaatst de inhoud van jug 0 naar jug 2 (jug 0 is nu leeg, er zit 5 in jug 2)
pour(1,2); -verplaatst de inhoud van jug 1 naar jug 2 (jug 1 is nu leeg, er zit nu in totaal 5+4=9
             in jug 2)
echo(volume(2));   -print de inhoud van jug 2

VERMENIGVULDIG 9 EN 4 MET ELKAAR:
jug(0,9);          -maakt jug '0' met een capaciteit van 9
jug(1,4);          -maakt jug '1' met een capaciteit van 4
jug(2,1);          -maakt jug '2' met een capaciteit van 1
jug(3,100)-maakt jug '3' met een capaciteit van 100
fill(0);           -vult jug 0 tot aan zijn capaciteit (er zit nu 9 in jug 0)
drain(0){  -doe het volgende tot jug 0 leeg is:
```

```
    fill(1);          -vult jug 1 tot aan zijn capaciteit (er zit nu 4 in jug 1)
    pour(1,3);        -verplaatst de inhoud van jug 1 naar jug 3 (er zit nu 0 in jug 1 en 4 in jug 3)
    pour(0,2);        -verplaatst de inhoud van jug 0 naar jug 2 (er zit nu 1 in jug 2 en 8 in jug 0, omdat jug 0
                       niet meer aan kan dan zijn capaciteit.
    empty(2);         -maak jug 2 leeg.
}           (de inhoud van jug 0 wordt dus telkens -1 en de inhoud van jug 3 telkens +4)
echo(volume(3));-print de inhoud van jug 3
```

Het enige andere wat Jug kan zonder gebruik van integers is het printen van een string. Hierbij moet de string tussen dubbele aanhalingstekens geplaatst worden en kan hij vervolgens geprint worden met de echo(); functie. Hier een voorbeeld van een functie die “Hello, world!” print.

```
echo(“Hello, world!”);
```

2.Waarom Jug niet regulier is -

De taal L van Jug is dus: L={jug(x,y);, fill(x);, empty(x);, pour(x,y);, echo(x);, volume(x);, if_empty(x){};, else{};, drain(x){}; | x >= 0 en y >= 0}
Als we aannemen dat L regulier is betekend dit dat L geaccepteerd kan worden door een DFA met k states. Als dit zo is dan kan doormiddel van het pomp-lemma elke string z∈L met lengte >= k geschreven worden als: z = uvw, met lengte (uv) =< k, lengte (v) > 0 en $uv^iw \in L$ voor elke i >= 0. Dit betekent dus dat als L regulier is, alle strings een gedeelte (v) hebben wat gepompt kan worden mits de lengte van de string gelijk of groot dan het aantal states is.
Bekijk nu de volgende string die in L zit:

```
drain(0){
fill(1);
pour(1,3);
pour(0,2);
empty(2);
}
```

Stel nu dat:

u =	v =	w =
drain(0)	{fill(1);	pour(0,2);
	pour(1,3);	empty(2);
		}

Dan produceert het pompen van v het volgende:

```
drain(0){
fill(1);
pour(1,3);
{
fill(1);
pour(1,3);
{
fill(1);
pour(1,3);
pour(0,2);
empty(2);
}
```

Dit zit niet in L (elke { wordt afgesloten met een } in L). v is dus niet pompbaar wat betekend dat L niet regulier is.

3.Waarom Jug context-vrij is

Een taal is context-vrij als er een grammatica of een PDA voor de taal gemaakt kan worden.
Deze grammatica accepteert elk geldig programma geschreven in Jug:

S → jug(x, y) A | jug(x, y) B | λ | echo(“string”)
A → fill(x) A | empty(x) A | R | λ
B → jug(x, y) B | fill(x) B | empty(x) B | pour(x, y) B | if_empty(x){B} C B |
B → drain(x){B}| RB | λ
C → else{B} | λ
R → echo(“string”) | echo(volume(x))

Toelichting:
Elke taal begint met een of meerdere Jugs die aangemaakt moet worden. Programma’s die alleen een string schrijven, of lege programma’s bestaan ook.
Om alles netjes compileerbaar te houden is er een onderscheid tussen een programma met 1 Jug en een programma met meerdere Jugs. Een programma met 1 Jug kan bijvoorbeeld geen pour-functie uitvoeren. Het is ook nutteloos om daarbij een if-else constructie in te maken. Wel kun je nog vullen en strings printen.
Programma’s met 2 of meerdere Jugs hebben meer mogelijkheden. Hierbij kun je ook loops maken met de drain-functie en een if-else statement maken. In tegenstelling tot een in dit geval onmogelijke reguliere expressie, is het alleen mogelijk om functies met haakjes ook daadwerkelijk af te sluiten met een haakje.