Radboud Universiteit Nijmegen

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Doolhof doorzoeken met een eindige automaat

Bachelorscriptie Wiskunde

Auteur:
Inge Mos
s4797396

Begeleider:
dr. Wieb Bosma

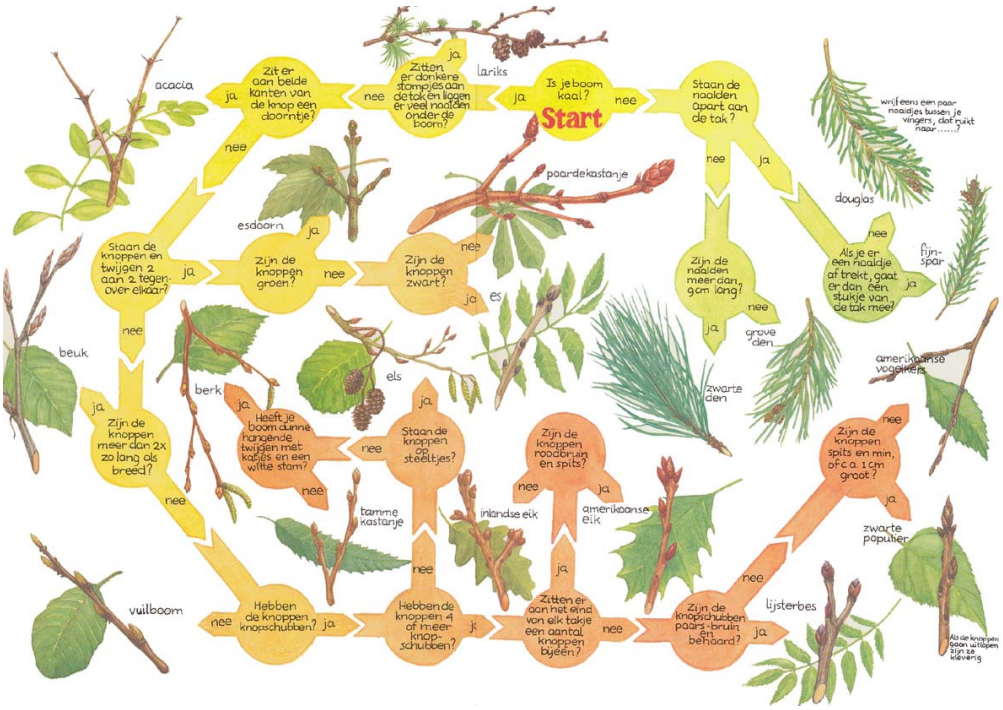
Tweede lezer:
dr. Henk Don

24 augustus 2023

Inhoudsopgave

1 Inleiding

Stel je wordt wakker en je bevindt je midden in een doolhof. Je lijdt aan extreem geheugenverlies. Je weet niet hoe je hier terechtgekomen bent en zelfs je kortetermijngeheugen lijkt volledig gewist. Een gevoel van desoriëntatie en onzekerheid overvalt je, want je beseft dat je alleen kunt waarnemen wat zich op je huidige locatie in het doolhof bevindt. Zodra je een stap hebt gezet, verdwijnt alle herinnering aan je vorige locatie, en je bent niet in staat om de omgeving te herkennen en te bepalen of je al eerder op dezelfde plek in het doolhof bent geweest. Gelukkig ben je niet volledig hulpeloos, want je hebt een rugzakje bij je, en in je rugzakje vind je twee genummerde kiezelsteentjes: kiezelsteentje 1 en kiezelsteentje 2. Ook vind je in je rugzakje een boekje getiteld "Doolhof ontsnapplan voor dummies". In een poging om een uitweg te vinden uit dit labyrint, open je het boekje en begin je te lezen. In het boekje staat niet veel tekst, het enige wat je ziet staan is: "houd je vinger op dit schema om de uitgang te vinden". Daarnaast zie je een groot schema staan. Je langetermijngeheugen doet het blijkbaar nog prima, want het schema dat je ziet, doet je veel denken aan een plantenspotschema van de basisschool (zie Figuur 1). Linksboven in het schema lees je "start". Je besluit je vinger op de start te leggen en verder de instructies te volgen.



Figuur 1: plantenspotschema van de basisschool (van [1])

Bovenstaande situatie vormt de kern van mijn scriptie. Het schema waar we net over praatten, bestaat namelijk echt, en is inderdaad vrij groot, waardoor we het in deze scriptie niet volledig kunnen gaan geven. Het schema stelt een eindige automaat voor, waarbij twee kiezelstenen als extra hulpmiddelen zijn toegevoegd. Een onderwerp waarbij we vooral het artikel van Manuel Blum en Dexter Kozen uit 1978, getiteld "On the power of the compass (or, why mazes are easier to search than graphs)"[2] zullen doorgronden.

In deze scriptie gaan we dieper in op de ideeën die in het genoemde artikel worden gepresenteerd. Blum en Kozen bewijzen in hun artikel dat een doolhof kan worden doorzocht door een eindige automaat met twee kiezelstenen. Voortbouwend op hun werk, geven we in deze scriptie een uitgebreider uitgewerkt algoritme, en delen van de eindige automaat waarvan zij vaststellen dat deze moet bestaan. Om dit te kunnen bereiken, hebben we JavaScript-code geschreven om een robot aan te sturen door een doolhof. Hierbij is er gebruik gemaakt van de website `robot.maarse.xyz` [4], gemaakt door Timo Maarse. Deze code wordt gegeven in Appendix A.

Het doel van deze scriptie is niet alleen om dieper inzicht te krijgen in hoe we een doolhof kunnen doorzoeken met een eindige automaat, daarnaast hopen we aan te tonen dat eindige automaten en beperkte geheugenmogelijkheden krachtige hulpmiddelen kunnen zijn bij het oplossen van een complex probleem.

2 Definitie van een doolhof

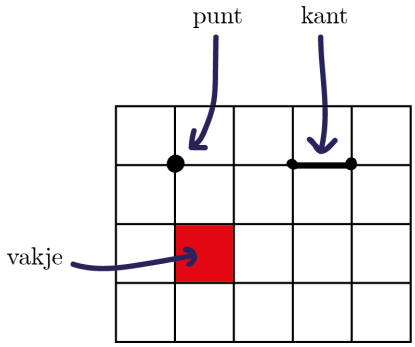
Er zijn veel verschillende manieren waarop je een doolhof kan construeren. Als je een doolhof andere eigenschappen geeft, zijn er weer andere dingen mogelijk, of juist onmogelijk. In dit hoofdstuk leggen we uit hoe het doolhof waar we verder mee gaan werken eruit ziet, wat de eigenschappen zijn, en wat we in zo'n doolhof precies zien.

2.1 Hoe ziet ons doolhof eruit

Om te beginnen, moeten we definiëren hoe het doolhof waarin je staat eruit ziet. Hiervoor introduceren we in deze paragraaf de benodigde definities.

Definitie 2.1 (punt, kant en vakje). We introduceren de volgende definities:

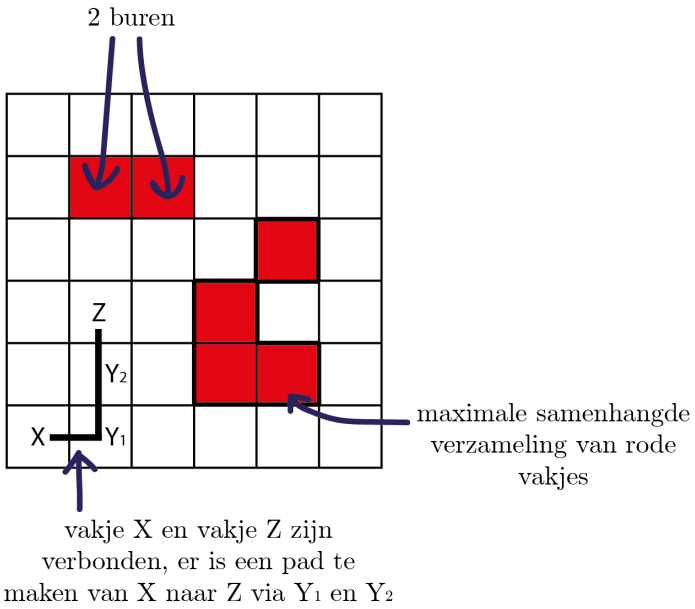
- Een *punt* (x, y) is een punt in het Euclidische vlak met coördinaten (x, y) en $x, y \in \mathbb{Z}$.
- Een *kant* is een lijnstuk van lengte 1 dat twee verschillende punten verbindt.
- Een *vakje* is een gebied ingesloten door 4 punten en 4 kanten.



Figuur 2: een punt, een kant en een vakje

Definitie 2.2 (buren, samenhangend, pad, verbonden en lus). We introduceren de volgende definities:

- Twee vakjes X, Y zijn *buren* als X en Y een kant delen.
- Twee vakjes X, Y zijn *samenhangend* als X en Y een punt delen.
- Er bestaat een *pad* van vakje X naar vakje Z als er een eindige rij van vakjes $Y_1, ..., Y_n$ (met $n \geq 1, X \neq Y_1$ en $Y_n = Z$) bestaat zodanig dat X en Y_1 zijn buren, Y_1 en Y_2 zijn buren, ..., Y_{n-1} en $Y_n = Z$ zijn buren.
- Twee vakjes X en Z zijn *verbonden* als er een pad van X naar Z bestaat.
- We noemen een pad een *lus* als $X{=}Z$.



Figuur 3: 2 buren, samenhangendheid, een pad en verbondenheid

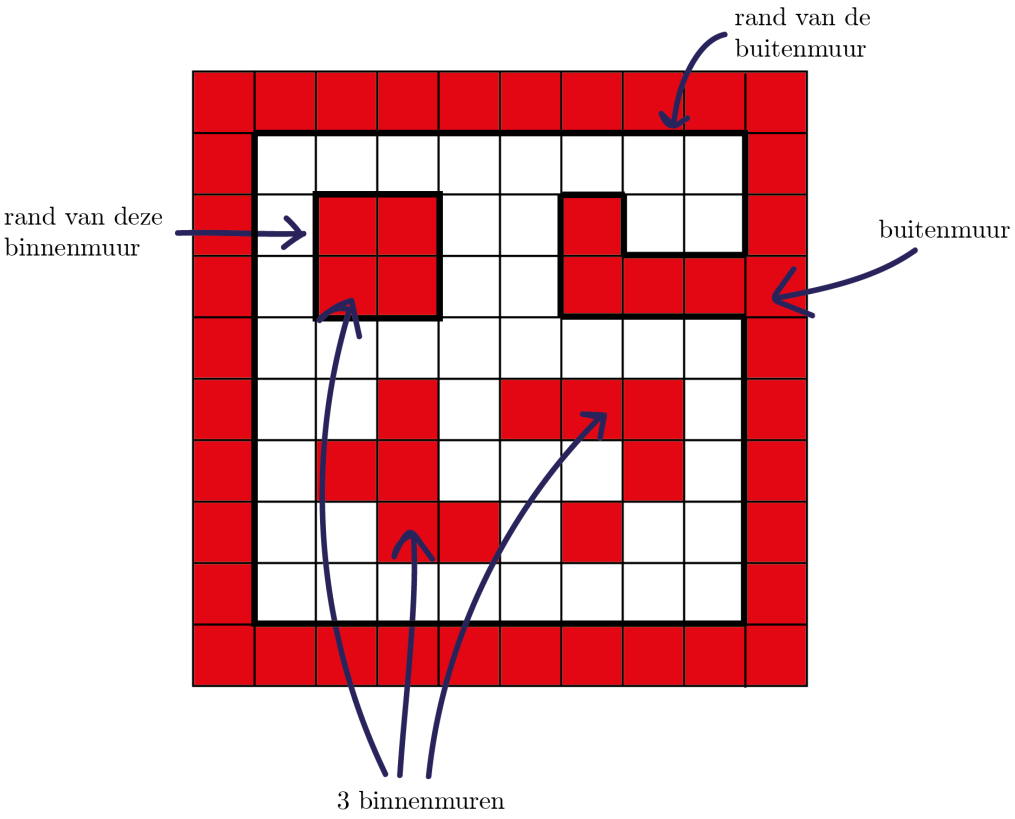
Definitie 2.3 (patroon, hoogte van het patroon, breedte van het patroon en eindig patroon). We introduceren de volgende definities:

- Een *patroon* P is samenhangende verzameling vakjes in het Euclidische vlak.
- Een vakje in een patroon kan wit of rood zijn.
- De *hoogte* h van een patroon P is $h = y_{max} - y_{min}$, met $(x, y_{min}), (x', y_{max}) \in P$ punten (zoals gedefinieerd in Definitie 2.1), waarbij geldt $y_{min} \leq y$ en $y_{max} \geq y'$ voor alle $(z, y), (z', y') \in P$. Indien zo'n punt (x, y_{min}) of (x', y_{max}) niet bestaat dan definiëren we $y_{min} = -\infty$ of $y_{max} = \infty$, en dan geldt: $h = \infty$.
- De *breedte* b van een patroon P is $b = x_{max} - x_{min}$, met $(x_{min}, y), (x_{max}, y') \in P$ punten (zoals gedefinieerd in Definitie 2.1), waarbij geldt $x_{min} \leq x$ en $x_{max} \geq x'$ voor alle $(x, z), (x', z') \in P$. Indien zo'n punt (x_{min}, y) of (x_{max}, y') niet bestaat dan definiëren we $x_{min} = -\infty$ of $x_{max} = \infty$, en dan geldt: $b = \infty$.
- Een *eindig patroon* P is een patroon waarvoor geldt $h < \infty$ en $b < \infty$.

Definitie 2.4 (binnenmuur, buitenmuur en rand). Laat P een eindig patroon zijn. We introduceren de volgende definities:

- een *muur* $M \subset P$ is een maximale samenhangende verzameling van rode vakjes.
- de *rand* van een muur is de verzameling van kanten uit P die de muur (de rode vakjes van P) scheiden van de aangrenzende witte vakjes van P.
- een *binnenmuur* is een muur die overal aan de rand van de muur omringd is door witte vakjes.
- een *buitenmuur* $B \subset P$ is een unieke muur die om alle vakjes $V = P \setminus B$ heenloopt: de vakjes van deze buitenmuur zijn de enige vakjes die samenhangend kunnen zijn

met vakjes uit $\mathbb{Z} \times \mathbb{Z}$ die niet in P zitten. Oftewel, als er een muur is met de eigenschap dat de vakjes van deze muur de enige vakjes zijn die samenhangend zijn met vakjes uit $\mathbb{Z} \times \mathbb{Z} \setminus P$, dan heet dit de unieke buitenmuur B van P.

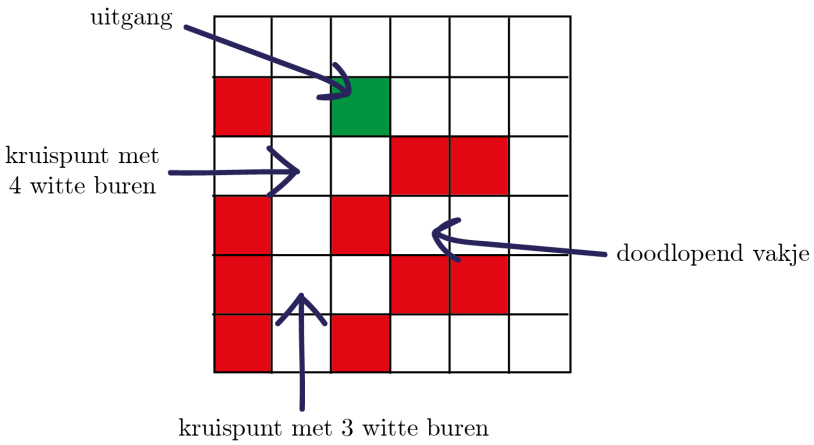


Figuur 4: een eindig patroon P van hoogte 10 en breedte 10 met hierin drie binnenmuren, een buitenmuur en twee randen van deze muren afgebeeld

Definitie 2.5 (doodlopend vakje en kruispunt). Laat P een patroon zijn. We introduceren de volgende definities:

- Een *doodlopend vakje* in P is een wit vakje met precies één witte buur.
- Een *kruispunt* in P is een wit vakje in een doolhof met minimaal drie witte buren.

Definitie 2.6 (uitgang). Een *uitgang* van een doolhof is een wit vakje met als extra eigenschap dat we vanuit dit witte vakje uit het doolhof kunnen ontsnappen (in de plaatjes maken we, om een uitgang duidelijk te kunnen aangeven, een vakje dat een uitgang is groen).



Figuur 5: een eindig patroon P met hierin een doodlopend vakje, twee kruispunten en een uitgang afgebeeld

Definitie 2.7 (doolhof). Een *doolhof* D is een eindig patroon P dat voldoet aan de volgende eisen:

- Een doolhof bevat altijd een eindig aantal vakjes.
- Een doolhof bevat een minimum van 12 vakjes.
- Elk vakje in het doolhof is ofwel *wit* of *rood* gekleurd.
- Een doolhof bevat altijd precies één uitgang (deze uitgang is groen gekleurd in de afbeeldingen, maar is een wit vakje).
- Alle buitenste vakjes van D zijn roodgekleurd en vormen de buitenmuur. Dus geen enkel wit vakje heeft een punt gemeen met een vakje dat niet tot het doolhof behoort.
- Een doolhof bevat een minimum van 2 witte vakjes.
- Alle vakjes in het doolhof zijn verbonden.
- Alle witte vakjes in het doolhof zijn verbonden, oftewel tussen elke twee witte vakjes $X, Y \in D$ bestaat een pad van witte vakjes.

2.2 Wat kunnen we zien, wat kunnen we doen, en wat is ons doel als we in het doolhof staan

We weten nu hoe een doolhof in deze scriptie eruit ziet, maar nog niet wat het precies betekent voor een gebruiker om in een doolhof te staan. Deze paragraaf is bedoeld om een idee te geven van wat we kunnen zien en welke acties we kunnen uitvoeren als we als gebruiker van een doolhof in het doolhof staan. Daarnaast willen we ook het doel aangeven dat een gebruiker wil bereiken in een doolhof.

2.2.1 Eisen voorafgaand

Voordat we kunnen beginnen met ons door het doolhof te gaan verplaatsen, moeten we wel zorgen dat aan enkele eisen wordt voldaan:

- het doolhof voldoet aan alle eisen genoemd in de definitie van een doolhof (2.7).
- wij (de gebruiker van het doolhof) staan op een wit vakje in het doolhof. De witte vakjes zijn namelijk de enige bewandelbare paden in een doolhof, en dus beginnen we ook op een wit vakje.

2.2.2 Hulpmiddelen

We zijn als gebruiker van het doolhof gelukkig niet helemaal aan ons lot overgelaten en hebben de volgende hulpmiddelen bij ons:

- een rugzakje met hierin 2 kiezelsteentjes met allebei een unieke naam:
  - kiezelsteentje 1
  - kiezelsteentje 2
- een kompas waarmee we altijd kunnen zien aan welke kant het noorden is.
- een schema (of voor een computer een eindige automaat) die ons de stappen die we zouden moeten zetten geeft waardoor we uit het doolhof zouden kunnen ontsnappen. Welke stappen mogelijk zijn om te zetten, oftewel welke acties we zouden kunnen uitvoeren, lees je in Subparagraaf 2.2.4. Welke acties we kunnen uitvoeren is gebaseerd op wat we om ons heen zien. Dit lees je in Subparagraaf 2.2.3.

2.2.3 Wat zien we om ons heen

De volgende dingen kunnen we zien vanuit het witte vakje in het doolhof waar we staan:

- of we wel of niet op een uitgang staan,
- de kleur van de vier buren van dit witte vakje: of het vakje ten noorden (en oosten, en zuiden, en westen) rood of wit is, waardoor we weten of het vakje ten noorden (en respectievelijk oosten, en zuiden, en westen) van waar we staan een muurvakje is of niet,
- of kiezelsteentje 1 (en kiezelsteentje 2) wel of niet ligt op dit witte vakje,
- of kiezelsteentje 1 (en kiezelsteentje 2) wel of niet in ons rugzakje zit, waardoor we weten of kiezelsteentje 1 (en respectievelijk kiezelsteentje 2) wel of niet al ergens in het veld geplaatst is.

2.2.4 Welke acties kunnen we uitvoeren

Als we in een vakje staan kunnen we ook meerdere acties uitvoeren. De volgende dingen **kunnen we doen** vanuit het witte vakje in het doolhof waar we staan:

- we kunnen uit het doolhof ontsnappen, mits we op een uitgang staan.
- we kunnen onszelf verplaatsen naar een buurvakje, mits dit wit is. We kunnen dus een stap zetten naar het noorden, oosten, zuiden of westen, mits het vakje waar we heen willen wit is.
- we kunnen een kiezelsteentje neerleggen op het witte vakje waar we staan, mits het kiezelsteentje dat we willen neerleggen nog in ons rugzakje zit. Dit kiezelsteentje (kiezelsteentje 1 of kiezelsteentje 2) verdwijnt dan uit ons rugzakje en blijft liggen in het vakje waar we staan.
- we kunnen een kiezelsteentje oppakken, mits dit kiezelsteentje ligt op het witte vakje waar we momenteel staan. Het kiezelsteentje wordt dan terug in het rugzakje gestopt.

2.2.5 Wat is het doel

Dan als laatste nog: wat is ons doel als we in een doolhof staan, en hoe willen we dat gaan bereiken:

- het doel is eigenlijk simpel: we willen de uitgang van het doolhof vinden.
- hoe willen we dit gaan bereiken: in de volgende hoofdstukken van deze scriptie gaan we een algoritme geven waarmee we het hele doolhof kunnen doorzoeken. Hiermee bedoelen we dat we ervoor zorgen dat we een route door het doolhof lopen waarbij we zeker weten dat we op alle witte vakjes in het doolhof geweest zijn, en zo dus het hele doolhof hebben doorzocht. Aangezien we weten dat de uitgang één van de witte vakjes in het doolhof is, zorgt het feit dat we het hele doolhof kunnen doorzoeken er dus ook voor dat we de uitgang sowieso kunnen vinden.

3 Standaard oplossingsalgoritmes van doolhoven

Er zijn veel algoritmes bekend om doolhoven op te lossen. De algoritmes die bekend zijn, kunnen we onderscheiden in twee soorten algoritmes:

1. Algoritmes die ontworpen zijn om gebruikt te worden door een persoon (of computer) waarbij de gebruiker een overzicht heeft van het gehele doolhof, en dus alle informatie van het hele doolhof in een keer kan overzien. Dit is bijvoorbeeld een doolhof-puzzel in een puzzelboekje.

2. Algoritmes die ontworpen zijn om gebruikt te worden door een persoon (of computer) waarbij de gebruiker zich in het doolhof bevindt. In dit geval heeft de persoon (of robot) geen voorkennis van het doolhof, en weet enkel de dingen die hij om zich heen kan zien.

Dit tweede is het soort algoritme waar we ons in deze scriptie op focussen, en dus ook het soort algoritme waar we in dit hoofdstuk enkele bekende voorbeelden van geven: het algoritme om de muur te volgen en Trémaux's algoritme. In de volgende paragrafen lichten we kort toe wat deze algoritmes inhouden en bespreken ook de voordelen en nadelen van het gebruik van deze algoritmes.

3.1 Muur volgen

Eén van de meest bekende algoritmes om een doolhof op te lossen, is het algoritme waarbij we de rand van een muur van een doolhof volgen [6]. Het idee van dit algoritme is dat de gebruiker de rechterhand (of linkerhand) op de rechtermuur (of respectievelijk linkermuur) legt, en met de hand op de muur deze muur blijft volgen totdat we bij de uitgang komen.

Een voordeel van dit algoritme is dat het bijna geen geheugen gebruikt, en aan een persoon (of computer) enkel een paar simpele regels moeten worden meegegeven om uit het doolhof te kunnen ontsnappen. Een (toch wel cruciaal) nadeel is echter, dat dit algoritme niet kan garanderen dat we de uitgang vinden voor elk doolhof D (gedefinieerd zoals in Hoofdstuk 2), startende vanuit X, een wit vakje uit doolhof D. Er is dus ook een kans is dat we bij gebruik van het algoritme om de muur te volgen voor eeuwig rondjes blijven lopen in het doolhof.

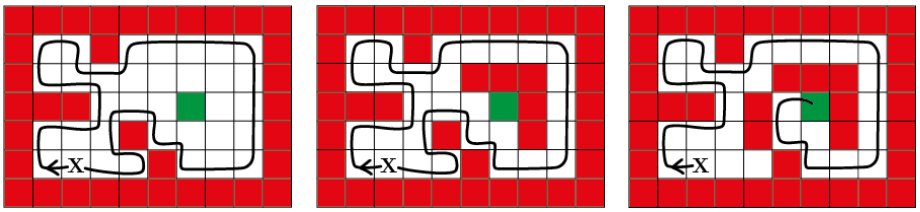
Wanneer werkt het algoritme wel en niet? Er zijn enkele situaties waarin het muur volgen algoritme niet werkt, in Figuur 6 zien we hier wat voorbeelden van. Als we willen dat het volgen van de muur voor elk doolhof D werkt, zouden we wat extra eisen aan een doolhof of waar we in het doolhof beginnen moeten stellen. Dit zijn de onderstaande eisen:

1. de uitgang Z van doolhof D moet aan een muur liggen, oftewel minimaal één van de buren van Z moet een muurvakje zijn.

2. de muur die de gebruiker gaat volgen, moet samenhangend zijn met de muur waar de uitgang Z aan ligt. Als het niet vast ligt op welk wit vakje van doolhof D de gebruiker begint, dan moeten alle muurvakjes samenhangend zijn, om te garanderen dat de gebruiker altijd de uitgang kan vinden.

3. het moet voor de gebruiker duidelijk zijn welke muur te volgen. Dit kan worden opgelost door de gebruiker altijd te laten beginnen op een wit vakje dat naast een muur ligt, dus op een wit vakje waarbij minimaal één van de buren een muurvakje moet zijn, zodat de gebruiker deze muur kan volgen. Een andere oplossing is

dat als de gebruiker op een wit vakje begint met geen muurvakjes als buren, de gebruiker dan eerst zo ver mogelijk naar (bijvoorbeeld) het noorden loopt tot hij bij een muur aankomt, om dan vervolgens deze muur rechtsom (of linksom) te gaan volgen.



(a) Z niet aan muur (b) X en Z aan andere muur (c) Z wordt gevonden

Figuur 6: drie doolhoven waarin we het algoritme toepassen waarin we de rechtermuur volgen, in (a) en (b) wordt de uitgang Z niet gevonden vanuit X, in (c) wel

3.2 Trémaux's algoritme

In [3] en [11] wordt het algoritme van Trémaux beschreven, een vrij eenvoudig algoritme waarmee we, in tegenstelling tot het algoritme om de muur te volgen, in elk doolhof D (gedefinieerd zoals in Hoofdstuk 2) de uitgang kunnen vinden, startende vanuit S, een wit vakje uit doolhof D. We vinden een route naar de uitgang door middel van het aanbrengen van markeringen op de grond. We kunnen de volgende markeringen aanbrengen in een vakje tegen een rand van dat vakje: ✕ of ●. Om die markeringen te kunnen aanbrengen, zijn net zoals bij het algoritme van het volgen van de muur, ook bij Tremaux's algoritme extra eisen nodig om uit het doolhof te kunnen ontsnappen. Ditmaal zijn dit geen eisen aan het doolhof of waar we beginnen in het doolhof, maar zullen we aan de hulpmiddelen in Subparagraaf 2.2.2 het volgende moeten toevoegen (of eigenlijk kunnen we de twee kiezelsteentjes door dit hulpmiddel vervangen): een goedgevulde markeerstift waarmee we zoveel ✕'s of ●'s kunnen markeren als we nodig hebben. Dit kan uiteindelijk best veel geheugen gaan kosten (alle aangebrachte markeringen moeten immers onthouden worden).

We geven zo het algoritme, maar om dat goed te kunnen begrijpen, hebben we eerst nog twee definities nodig:

Definitie 3.1 (Nieuw kruispunt). Een *nieuw kruispunt* N is een kruispunt waarbij tegen geen enkele van de randen van N markeringen (een ✕ of ●) zijn aangebracht.

Definitie 3.2 (Oud kruispunt). Een *oud kruispunt* O is een kruispunt waarbij tegen minimaal één van de randen van O een markering (een ✕ of ●) is aangebracht.

Nu kunnen we het algoritme geven. In Trémaux's algoritme beginnen we vanuit S en kiezen een willekeurige witte buur van S. We zetten een stap in de richting van deze witte buur. Vervolgens gaan we naar Stap 1. van de onderstaande stappen:

1. er zijn vier mogelijkheden (zie Figuur 7):
   (a) het vakje waar we op staan is de uitgang. In dit geval hebben we de uitgang gevonden en stoppen we.
   (b) het vakje waar we op staan is niet de uitgang, en heeft exact één witte buur. In dit geval zetten we een stap naar deze enige witte buur, en gaan naar Stap 2.

(c) het vakje waar we op staan is niet de uitgang, en heeft exact twee witte buren. In dit geval zetten we een stap naar de andere witte buur dan de witte buur waar we zojuist vandaan komen, en gaan terug naar Stap 1.

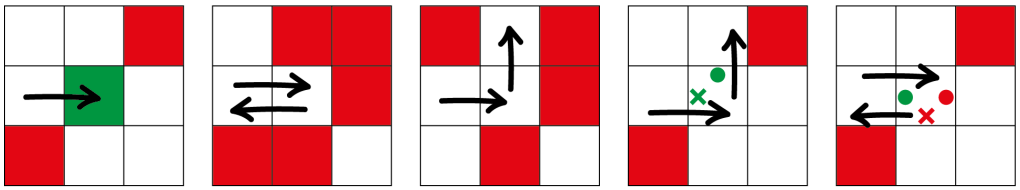
(d) het vakje waar we op staan heeft drie of vier witte buren. In dit geval zijn er twee verschillende mogelijkheden:

i. het vakje waar we op staan is een kruispunt N. In dit geval voeren we de volgende stappen uit:

- we markeren een ✕ in het vakje N tegen de rand tussen N en de buur uit de richting waar we zojuist vandaan komen.
- we kiezen willekeurig één van de ongemarkeerde witte buren van N, en markeren een ● in het vakje N tegen de rand tussen N en de zojuist gekozen buur.
- we zetten een stap in de richting van de zojuist gekozen buur, en gaan terug naar Stap 1.

ii. het vakje waar we op staan is een oud kruispunt O. In dit geval voeren we de volgende stappen uit:

- we markeren een ● in het vakje O tegen de rand tussen O en de buur uit de richting waar we zojuist vandaan komen.
- we draaien om, zetten een stap terug in de richting waar we zojuist vandaan kwamen, en gaan naar Stap 2.



(a) (b) (c) (d)i. (d)ii.

Figuur 7: de verschillende mogelijkheden in Stap 1 van Trémaux's algoritme. Om de situaties duidelijk af te beelden zijn de nieuw aangebrachte markeringen in het plaatje groen afgebeeld, en de markeringen die er al stonden zijn in dit figuur rood. Dit is alleen voor verduidelijking van het figuur: in het algoritme worden alle markeringen gewoon aangebracht in het zwart.
In de situatie (a) hebben we de uitgang gevonden en zijn we dus klaar, in (b) en (d)ii. gaan we (na het uitvoeren van de afgebeelde stappen) naar Stap 2 van het algoritme, en in (c) en (d)i. blijven we (na het uitvoeren van de afgebeelde stappen) in Stap 1.

2. er zijn twee mogelijkheden (zie Figuur 8):

(a) het vakje waar we op staan heeft exact twee witte buren. In dit geval zetten we een stap naar de andere witte buur dan de witte buur waar we zojuist vandaan komen, en gaan terug naar Stap 2.

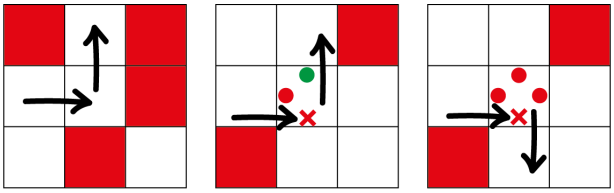
(b) het vakje waar we op staan heeft drie of vier witte buren. In dit geval zijn er twee verschillende mogelijkheden:

i. het vakje waar we op staan is een oud kruispunt O, en er staan minder markeringen in O dan dat O witte buren heeft. In dit geval voeren we de volgende stappen uit:

- we kiezen willekeurig één van de witte buren van O waar nog geen markering tegen de rand is gezet, en markeren een ● in het vakje O tegen de rand tussen O en de zojuist gekozen buur.

- we zetten een stap in de richting van de zojuist gekozen buur, en gaan terug naar Stap 1.

ii. het vakje waar we op staan is een oud kruispunt O, en er staan net zoveel markeringen in O als dat O witte buren heeft. In dit geval kiezen we de rand met de $\times$ (deze is er altijd), zetten een stap in de richting van deze buur en gaan terug naar Stap 2.



(a) (b)i. (b)ii.

Figuur 8: de verschillende mogelijkheden in Stap 2 van Trémaux's algoritme. Om de situaties duidelijk af te beelden zijn de nieuw aangebrachte markeringen weer in het plaatje groen afgebeeld, en de markeringen die er al stonden zijn in dit figuur rood. Dit is alleen voor verduidelijking van het figuur: in het algoritme worden alle markeringen gewoon aangebracht in het zwart.
In de situatie (a) en (b)ii. blijven we (na het uitvoeren van de afgebeelde stappen) in Stap 2 van het algoritme, en in (b)i. gaan we (na het uitvoeren van de afgebeelde stappen) terug naar Stap 1.

Een groot voordeel van Trémaux's algoritme, in tegenstelling tot het algoritme om de muur te volgen, is dat Trémaux's algoritme altijd leidt tot de uitgang van het doolhof. Voor het bewijs hiervan, verwijzen we naar [3]. Een nadeel is dat er meer geheugen nodig is naarmate het doolhof groter of complexer wordt: alle markeringen die we aanbrengen, moeten immers onthouden worden en het aantal markeringen dat we maximaal nodig hebben is evenveel als het totaal aantal witte buren van alle kruispunten bij elkaar opgeteld. Dit kan dus erg veel worden.

Later in deze scriptie gaan we aantonen dat er ook een algoritme bestaat dat:

- ervoor zorgt dat we uit elk doolhof (gedefinieerd zoals in Hoofdstuk 2 kunnen ontsnappen
- een vaste hoeveelheid geheugen gebruikt hoe groot het doolhof ook wordt: altijd één automaat waaraan 2 kiezelsteentjes zijn toegevoegd.

In het volgende hoofdstuk geven we een uitleg van wat een automaat precies is.

4 Introductie automaten

We willen een doolhof, zoals we dat zojuist gedefinieerd hebben, kunnen doorzoeken met een eindige automaat. In dit hoofdstuk geven we een korte uitleg over wat een eindige automaat eigenlijk is, en hoe de automaat eruit ziet die we gaan gebruiken om het doolhof te doorzoeken. Om dit uit te leggen, geven we eerst een definitie van een gerichte graaf, vervolgens beginnen we met het definiëren van een eindige automaat op een wat intuïtievere, informelere manier, en tot slot geven we de wiskundige definitie van een Mealy machine, de automaat die er het meest uit ziet als de automaat die we uiteindelijk gaan gebruiken om uit een doolhof te ontsnappen, zoals gedefinieerd in Hoofdstuk 2.

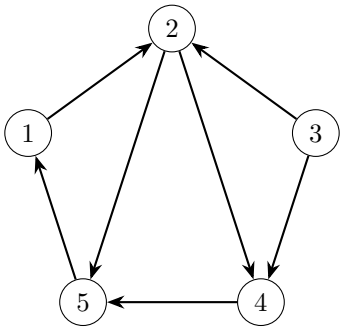
4.1 Definitie gerichte graaf

Een eindige automaat wordt vaak weergegeven als een gerichte graaf. Om straks een eindige automaat te kunnen weergeven, geven we eerst de definitie van een gerichte graaf (zoals deze gegeven is in [8]):

Definitie 4.1 (Gerichte graaf). Een *gerichte graaf* G is een paar $G = (V, A)$, waarbij V een eindige verzameling is, en A een verzameling van geordende paren elementen van V. Een element van V heet een *knoop*, en een element van A noemen we een *pijl*.

We geven een voorbeeld van een gerichte graaf:

Voorbeeld 4.2 (Gerichte graaf). Hieronder zie je de gerichte graaf $G = (\{1, 2, 3, 4, 5\}, \{(1, 2), (2, 4), (2, 5), (3, 2), (3, 4), (4, 5), (5, 1)\})$:



G is dus een gerichte graaf met 5 knopen: $V = \{1, 2, 3, 4, 5\}$, en 7 pijlen: $A = \{(1, 2), (2, 4), (2, 5), (3, 2), (3, 4), (4, 5), (5, 1)\}$.

4.2 Uitleg eindige automaat

Nu we hebben gedefinieerd wat een gerichte graaf is, geven we een korte, intuïtieve, informele uitleg van een eindige automaat. Daarna geven we een eindige automaat weergegeven als een gerichte graaf.

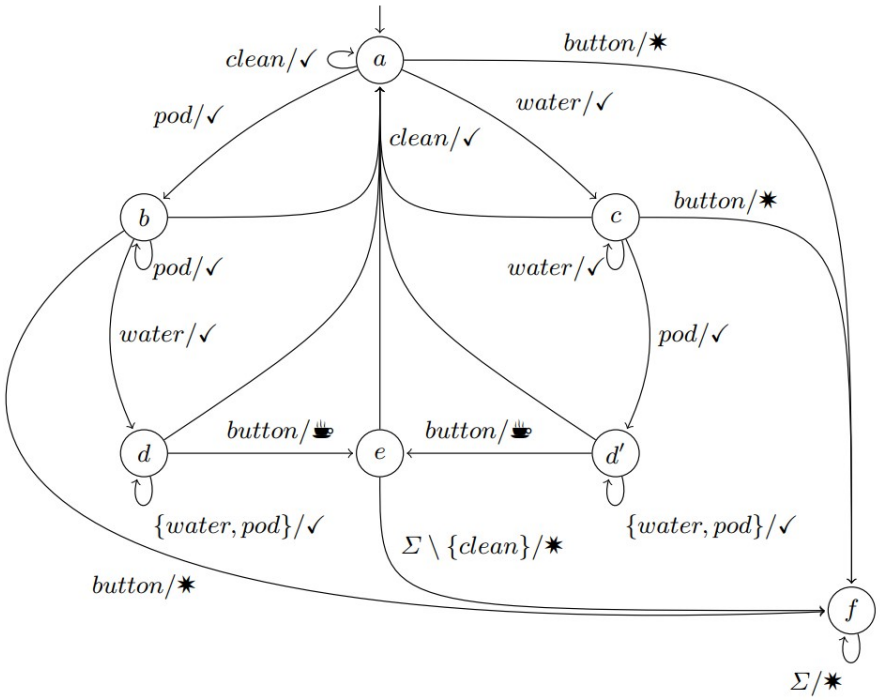
Zoals omschreven in [10] en [7] is een *eindige automaat* een abstract model dat het gedrag van een bepaald systeem kan analyseren of simuleren. Een automaat bestaat uit een aantal interne besturingstoestanden (in een eindige automaat is het aantal interne toestanden eindig) en transities tussen deze toestanden. Tijdens het modelleren van een bepaald systeem begint het proces in de *begintoestand* van een automaat, waarna het zich kan verplaatsen door de verschillende toestanden van de automaat, om uiteindelijk

te stoppen in een *eindtoestand* (mits deze aanwezig is). Als er geen eindtoestand aanwezig is, zal de automaat nooit eindigen en dus oneindig lang doorgaan.

Een automaat bevindt zich altijd in precies één van zijn toestanden tegelijk. Er zijn overgangen tussen de toestanden in een automaat. Dit noemen we ook wel de *transities* tussen de toestanden van een automaat. De transities in een automaat zijn gebaseerd op bepaalde inputwaardes uit de omgeving. Alle mogelijke inputwaardes vormen samen een eindige verzameling: het *input-alfabet*. In een doolhof zoals we dat gedefinieerd hebben in Hoofdstuk 2 bestaat het input-alfabet uit de verschillende waardes onder *'wat zie je'* in Paragraaf 2.2. Bij een bepaalde transitie tussen twee toestanden, hoort (in het geval van onze automaat) ook een outputwaarde. De mogelijke outputwaardes zitten allemaal in een eindige verzameling: het *output-alfabet*. Het output-alfabet bestaat in ons geval uit alle mogelijke acties die uitgevoerd kunnen worden, deze acties zijn omschreven in *'wat kun je'* in Paragraaf 2.2.

Een eindige automaat wordt vaak afgebeeld als een gerichte graaf, waarbij de verschillende toestanden uit de automaat corresponderen met de knopen uit de gerichte graaf, en de verschillende transities uit de automaat corresponderen met de verschillende pijlen uit de gerichte graaf. Er is alleen een transitie mogelijk van toestand x naar toestand y als deze overgang in de transitiefunctie zit (als de pijl $(x, y) \in A$ zit (met A de verzameling pijlen van de gerichte graaf)).

Een automaat kan dus de werking van een bepaald systeem afbeelden. Hieronder staat een voorbeeld van een automaat die de werking van een koffiezetautomaat weergeeft. Dit voorbeeld komt uit [12]:



Figuur 9: Voorbeeld van automaat die de werking van een koffiezetapparaat omschrijft. Deze afbeelding komt uit [12].

In Figuur 9 zien we een automaat die de werking van een koffiezetautomaat afbeeldt. We zien dat de automaat eruitziet als een gerichte graaf. De automaat heeft 7 verschillende toestanden (de knopen van de gerichte graaf): a, b, c, d, d', e en f. De begintoestand (de toestand waarin de automaat begint) is in dit geval toestand a, te herkennen aan het pijltje dat naar a wijst. De automaat bevat geen eindtoestand, maar zodra je in toestand f bent gekomen, zit je hier vast in een oneindige lus.

De pijlen tussen de verschillende toestanden zijn de transities van de automaat. We zien dat bij elke pijl een notitie in de vorm van *woord*/*symbool* staat, waarbij $woord \in \Sigma = \{clean, pod, button, water\}$ en $symbool \in \Omega = \{$ ✓, ☕, ✷$\}$. Hier is Σ het input-alfabet en Ω het output-alfabet dat hoort bij de automaat. Het input-alfabet bestaat uit de mogelijke handelingen die iemand zou kunnen uitvoeren op de machine:

- *clean*: verwijder (de oude) koffiepad en het resterende water
- *pod*: voeg een verse koffiepad toe
- *button*: druk op de knop om de productie van koffie te starten
- *water*: vul de watertank bij.

Het output-alfabet bestaat uit de reactie die de machine zou moeten geven op een mogelijke handeling van de gebruiker:

- ✓: de koffiemachine keurt de handeling van de gebruiker goed
- ☕: de koffiemachine maakt een kop koffie
- ✷: de koffiemachine geeft een error terug op de handeling van de gebruiker.

Het in de automaat beschreven koffiezetapparaat kan alleen koffie zetten nadat het water in de watertank is bijgevuld, en een verse koffiepad aan het apparaat is toegevoegd (beide handelingen kunnen in willekeurige volgorde worden gedaan, het wordt op beide manieren goedgekeurd door het koffiezetapparaat (✓)). Als er op de knop wordt gedrukt om koffie te zetten, nadat de handelingen *water* en *pod* uitgevoerd zijn, wordt er koffie gezet (☕) (de automaat gaat naar toestand e). Vervolgens zou men de oude koffiepad en het resterende water moeten verwijderen (*clean*), voordat er weer opnieuw koffie kan worden gezet. Als de handeling *button* wordt aangeroepen terwijl daarvoor niet zowel *water* als *pod* uitgevoerd zijn, of als na het zetten van een kop koffie niet eerst *clean* wordt gedaan, geeft het apparaat een error (✷) terug, en gaat naar toestand f. Als we eenmaal in toestand f terecht zijn gekomen, blijven we hier vervolgens in vast zitten. Een probleem aan de werking van dit koffiezetapparaat is dus dat zodra het apparaat een error heeft gegeven, het alleen maar errors blijft geven. In de automaat in Figuur 9 zien we een schematische weergave (een automaat dus) van de beschrijving van dit proces.

4.3 Definitie Mealy machine

Nu we een beetje een idee hebben van wat een eindige automaat is, kunnen we de formele definitie geven van een Mealy machine. De automaat van het koffiezetapparaat van [12] is ook een Mealy machine.

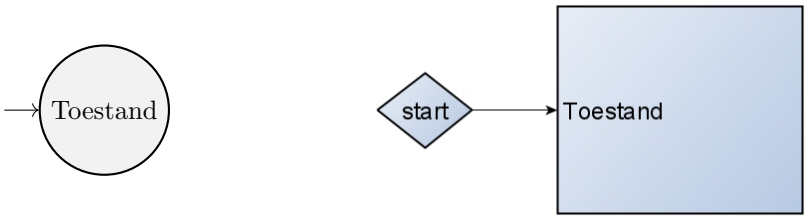
Definitie 4.3 (Mealy machine). Een Mealy machine M is een 5-tupel $M = (Q, q_0, E, I, O, T)$ bestaande uit het volgende:

- Q is een eindige, niet-lege, verzameling toestanden
- $q_0 \in Q$ is de begintoestand
- $E \subset Q$ is de verzameling van eindtoestanden
- I is een eindige verzameling met het input-alfabet
- O is een eindige verzameling met het output-alfabet
- T is een transitiefunctie $T : Q \times I \to Q \times O$.

Hieronder staan de notaties van een begintoestand, gewone toestand en een eindtoestand.

Notatie:
De verschillende toestanden worden op de volgende manier weergegeven:

1. **Begintoestand:**

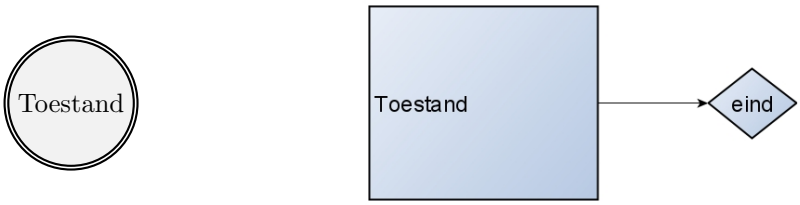


Figuur 10: twee verschillende weergaven van de begintoestand van een automaat: links is de meest gebruikelijke manier om de begintoestand van een automaat aan te duiden (met een pijltje dat vanuit het niks komt en naar deze toestand wijst), rechts is hoe de begintoestand in de schema's in deze scriptie wordt weergegeven.

2. **Gewone toestand:**

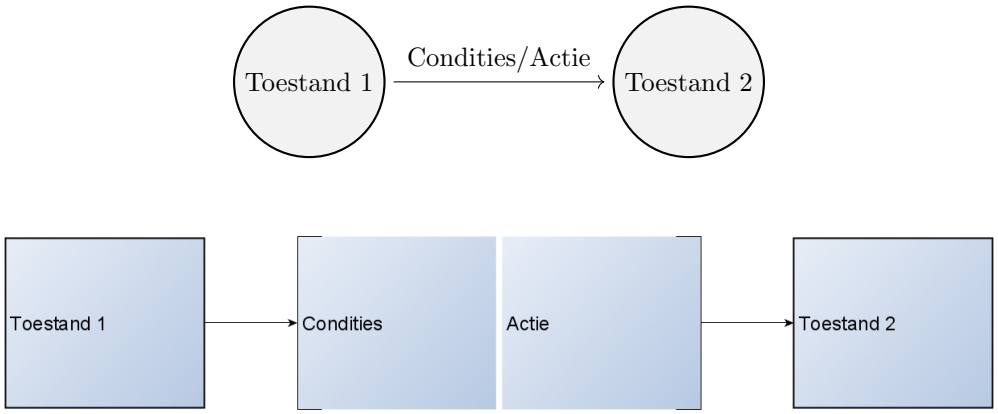
Figuur 11: twee verschillende weergaven van de gewone toestand van een automaat: links is de meest gebruikelijke manier om de toestand van een automaat aan te duiden, rechts is hoe de toestanden in de schema's in deze scriptie worden weergegeven.

3. **Eindtoestand:**



Figuur 12: twee verschillende weergaven van de eindtoestand van een automaat: links is de meest gebruikelijke manier om een eindtoestand van een automaat aan te duiden (met een dubbele rand), rechts is hoe een eindtoestand in de schema's in deze scriptie wordt weergegeven.

4. **Transitie:**
   De afbeeldingen hieronder geven een transitie van q_1 naar q_2 weer, waarbij Condities $\in I$ de input (kenmerken uit de omgeving), en Actie $\in O$ de output is (een actie die uitgevoerd moet worden):



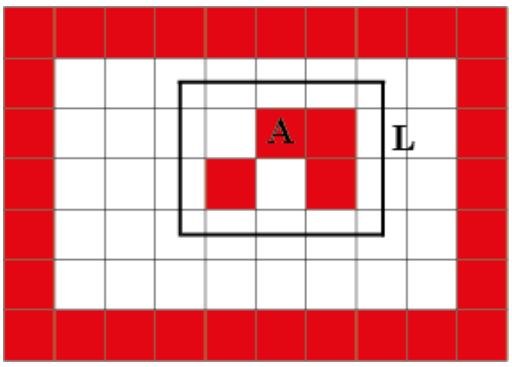
Figuur 13: twee verschillende weergaven van een transitie in een automaat: boven is de meest gebruikelijke manier om een transitie van een automaat aan te duiden (met de input- en outputwaardes boven de pijl met een / ertussen), onder is hoe een transitie in de schema's in deze scriptie wordt weergegeven.

5 Eenvoudig doolhof doorzoeken

Om uiteindelijk elk doolhof volledig te kunnen doorzoeken, gaan we eerst een algoritme geven om een versimpelde versie van een doolhof, een eenvoudig doolhof, te kunnen doorzoeken. In dit hoofdstuk leggen we uit wat een eenvoudig doolhof is en geven we een algoritme om een eenvoudig doolhof te doorzoeken.

Definitie 5.1 (Eenvoudig doolhof). Laat D een doolhof zijn. De verzameling van alle witte vakjes in D is enkelvoudig samenhangend als er voor alle lussen L in D geldt dat L geen muur uit D omsluit. Als dit voor D geldt dan noemen we D een *eenvoudig doolhof*.

Opmerking: Bovenstaande definitie geeft aan dat een doolhof D geen eenvoudig doolhof is als er een lus te vinden is in D die een muur omsluit. Laat $B \subset D$ de buitenmuur zijn van D. Let op dat als D een binnenmuur bevat, zeg $A \subset D$ is een binnenmuur van D, dan bestaat er geen enkel rood vakje $R \in A$ waarvoor geldt dat R samenhangend is met een vakje $X \in B$ van de buitenmuur. Hierdoor zijn er aan alle kanten van A witte vakjes te vinden tussen A en de buitenmuur B, waaruit volgt dat er een lus te maken is in de witte vakjes van D om A heen, zie Figuur 14. Het doolhof in Figuur 14 is dus geen eenvoudig doolhof is. En in het algemeen geldt: een doolhof D is een eenvoudig doolhof als D geen binnenmuren bevat.



Figuur 14: Lus L om binnenmuur A

Stelling 5.2. *Laat D een eenvoudig doolhof zijn, dan is er een eindige automaat die D volledig kan doorzoeken, oftewel alle witte vakjes in D bezoeken door zichzelf door D te verplaatsen.*

Opmerking: Met het bewijs van deze stelling tonen we aan dat we een eenvoudig doolhof volledig kunnen doorzoeken. Het uiteindelijke doel 'vind de uitgang' kan bereikt worden door te zorgen dat we alle witte vakjes bezoeken. De uitgang is een wit vakje en wordt dan sowieso ook bezocht.

Bewijs. We weten dat D een eenvoudig doolhof is, en dat D dus geen binnenmuren bevat. Laat X een willekeurig wit vakje van D zijn. We willen een constructie geven om D volledig te doorzoeken. Het idee van deze constructie is dat we de buitenmuur van het doolhof volgen met de muur steeds aan onze linkerkant. Hierbij maken we een tussenstap bij elke stap waarbij we langs een zuidmuur lopen: we lopen helemaal naar het noorden, en weer terug naar de zuidmuur, om vervolgens de muur weer verder te volgen. Hierdoor verbeteren we dus eigenlijk het algoritme "Muur volgen" uit Paragraaf 3.1, waardoor de extra eisen we daar genoemd hebben niet meer nodig zijn. We kunnen dit algoritme uitwerken, startende vanuit X, volgens de volgende stappen:

1. We willen zo ver mogelijk naar het noorden lopen om daarna vervolgens zo ver mogelijk naar het zuiden te kunnen lopen. Er vindt een van de volgende situaties plaats:

   (a) Er is geen muur in het noorden. Verplaats één vakje naar het noorden. Ga vervolgens terug naar Stap 1.

   (b) Er is wel een muur in het noorden, maar er is geen muur in het zuiden. We zijn dus zover mogelijk naar het noorden gelopen (we staan daar bij een muur). We willen nu dus zover mogelijk naar het zuiden lopen. Verplaats één vakje naar het zuiden, en ga naar Stap 2.

   (c) Er is een muur in het noorden en een muur in het zuiden, maar geen muur in het westen. We zijn nu al zo ver mogelijk naar het noorden en zover mogelijk terug naar het zuiden gelopen, je kan namelijk beide kanten niet op. Nu willen we de zuidmuur gaan volgen met de muur aan onze linkerkant. Verplaats één vakje naar het westen, en ga naar Stap 3.

   (d) Er is een muur in het noorden, een muur in het zuiden en een muur in het westen, maar geen muur in het oosten. We zijn nu al zo ver mogelijk naar het noorden en zover mogelijk terug naar het zuiden gelopen, je kan namelijk beide kanten niet op. Nu willen we de muur waardoor we omringd zijn, gaan volgen met de muur aan onze linkerkant. Verplaats één vakje naar het oosten, en ga naar Stap 4.

2. We hebben zojuist een stap naar het zuiden gezet en willen nu zo ver mogelijk naar het zuiden lopen tot dat niet meer kan. Zodra dat niet meer kan, willen we de muur volgen met de muur aan onze linkerkant. Er vindt een van de volgende situaties plaats:

   (a) Er is geen muur in het zuiden. Verplaats één vakje naar het zuiden. Ga vervolgens terug naar Stap 2.

   (b) Er is wel een muur in het zuiden, maar geen muur in het westen. We zijn nu al zo ver mogelijk naar het zuiden gelopen (we staan bij een zuidmuur). We willen nu dus de zuidmuur volgen, met de muur aan onze linkerkant. Verplaats één vakje naar het westen, en ga naar Stap 3.

   (c) Er is een muur in het zuiden, en een muur in het westen, maar geen muur in het noorden. Dit is altijd zo, omdat we zojuist een stap naar het zuiden hebben gezet, en dus uit het noorden kwamen, waardoor in het noorden geen muur kan zijn. We zijn nu zo ver mogelijk naar het zuiden gelopen (we staan bij een zuidmuur). We willen nu deze zuidmuur volgen met de muur aan onze linkerkant. Verplaats één vakje naar het noorden, en ga naar Stap 5.

3. We hebben zojuist een stap naar het westen gezet en hebben daarmee de muur gevolgd. Er vindt een van de volgende vier situaties plaats:

   (a) Er is geen muur in het zuiden. In dit geval willen we de muur nog gewoon blijven volgen. We lopen namelijk alleen helemaal naar het noorden en vervolgens weer terug naar het zuiden als we langs een zuidmuur lopen, en dat is nu niet het geval. We blijven de muur dus volgen met de muur aan de linkerkant. Verplaats één vakje naar het zuiden, en ga naar Stap 6.

   (b) Er is een muur in het zuiden, maar geen muur in het noorden. We zijn nu dus zojuist een stap naar het westen gegaan door de muur te volgen, en hebben een muur langs de zuidkant. We moeten nu dus zo ver mogelijk naar het noorden lopen, om daarna weer terug naar het zuiden te lopen, en de muur

weer verder te volgen. Omdat er geen muur in het noorden zit, kunnen we naar het noorden. Verplaats één vakje naar het noorden, en ga terug naar Stap 1.

(c) Er is een muur in het zuiden en een muur in het noorden, maar geen muur in het westen. We zouden nu dus weer zo ver mogelijk naar het noorden moeten lopen, om vervolgens zo ver mogelijk terug naar het zuiden te lopen, maar we staan tussen een noord- en zuidmuur in, waardoor dit niet kan (of we het eigenlijk al gedaan hebben). Hierdoor willen we de muur gewoon verder volgen. Verplaats één vakje naar het westen, en ga terug naar Stap 3.

(d) Er is een muur in het zuiden, een muur in het noorden, een muur in het westen, maar geen muur in het oosten. We zouden nu dus weer zo ver mogelijk naar het noorden moeten lopen, om vervolgens zo ver mogelijk terug naar het zuiden te lopen, maar we staan tussen een noord- en zuidmuur in, waardoor dit niet kan (of we het eigenlijk al gedaan hebben). Hierdoor willen we de muur gewoon verder volgen. We kunnen niet verder naar het westen, want daar zit een muur. Verplaats één vakje naar het oosten, en ga naar Stap 4.

4. We hebben zojuist een stap naar het oosten gezet en hebben daarmee de muur gevolgd. Er vindt een van de volgende vier situaties plaats:

   (a) Er is geen muur in het noorden. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het oosten bent verplaatst). Verplaats één vakje naar het noorden, en ga naar Stap 5.

   (b) Er is een muur in het noorden, maar geen muur in het oosten. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het oosten bent verplaatst). Verplaats één vakje naar het oosten, en ga terug naar Stap 4.

   (c) Er is een muur in het noorden en een muur in het oosten, maar er is geen muur in het zuiden. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het oosten bent verplaatst). Verplaats één vakje naar het zuiden, en ga terug naar Stap 6.

   (d) Er is een muur in het noorden, een muur in het oosten en een muur in het zuiden, maar geen muur in het westen. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het oosten bent verplaatst). Verplaats één vakje naar het westen, en ga terug naar Stap 3.

5. We hebben zojuist een stap naar het noorden gezet en hebben daarmee de muur gevolgd. Er vindt een van de volgende vier situaties plaats:

   (a) Er is geen muur in het westen. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het noorden bent verplaatst). Verplaats één vakje naar het westen, en ga terug naar Stap 3.

   (b) Er is een muur in het westen, maar geen muur in het noorden. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het noorden bent verplaatst). Verplaats één vakje naar het noorden, en ga terug naar Stap 5.

(c) Er is een muur in het westen en een muur in het noorden, maar er is geen muur in het oosten. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het noorden bent verplaatst). Verplaats één vakje naar het oosten, en ga terug naar Stap 4.

(d) Er is een muur in het westen, een muur in het noorden en een muur in het oosten, maar geen muur in het zuiden. We willen de muur gewoon blijven volgen (we volgen namelijk op het moment niet een zuidmuur, die zit namelijk niet aan de linkerkant als je zojuist een vakje naar het noorden bent verplaatst). Verplaats één vakje naar het zuiden, en ga naar Stap 6.

6. We hebben zojuist een stap naar het zuiden gezet en hebben daarmee de muur gevolgd. Er vindt een van de volgende vier situaties plaats:

   (a) Er is geen muur in het oosten. We willen de muur gewoon blijven volgen (we willen niet naar de situatie om helemaal naar het noorden te lopen en vervolgens weer terug naar het zuiden te lopen, omdat we op het moment niet de muur aan het volgen zijn langs een zuidmuur. Dat is namelijk niet de muur aan de linkerkant als je zojuist een vakje naar het zuiden bent verplaatst). Verplaats één vakje naar het oosten, en ga terug naar Stap 4.

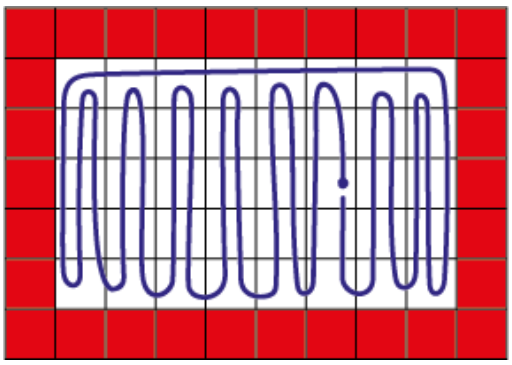
   (b) Er is een muur in het oosten, maar er is geen muur in het zuiden. We willen de muur gewoon blijven volgen (we willen niet naar de situatie om helemaal naar het noorden te lopen en vervolgens weer terug naar het zuiden te lopen, omdat we op het moment niet de muur aan het volgen zijn langs een zuidmuur. Dat is namelijk niet de muur aan de linkerkant als je zojuist een vakje naar het zuiden bent verplaatst). Verplaats één vakje naar het zuiden, en ga terug naar Stap 6.

   (c) Er is een muur in het oosten en een muur in het zuiden, maar er is geen muur in het noorden. Dit is altijd zo, omdat we zojuist een stap naar het zuiden hebben gezet, en dus uit het noorden kwamen, waardoor in het noorden geen muur kan zijn. We zijn nu bij een zuidmuur aangekomen, die ook aan onze linkerkant zit, en we dus aan het volgen zijn. Dit betekent dat we eerst weer zo ver mogelijk naar het noorden moeten lopen, en daarna terug naar het zuiden, voordat we de muur verder volgen. Verplaats één vakje naar het noorden, en ga terug naar Stap 1.

□

Opmerking: In bovenstaand algoritme blijven we oneindig lang rondlopen door doolhof D. We kunnen het algoritme wel stoppen door in het doolhof een uitgang toe te voegen en in de automaat in elke toestand te controleren of we op een uitgang staan. Als dit zo is, dan kunnen we het algoritme stoppen en zijn we uit het doolhof ontsnapt.

In Figuur 15 zie je een weergave van hoe het algoritme werkt om het eenvoudige doolhof te doorzoeken.

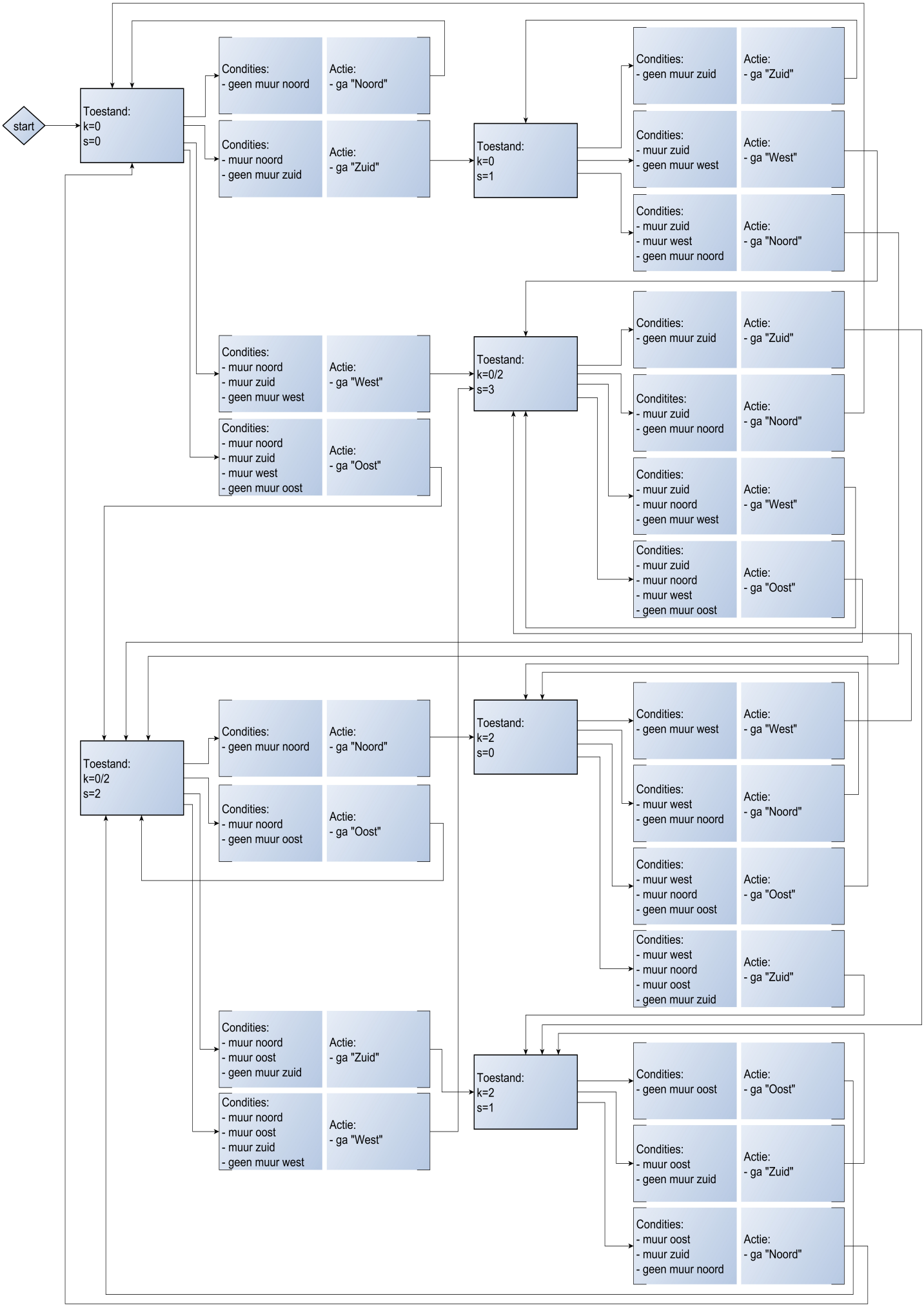


Figuur 15: Eenvoudig doolhof doorzoeken

We kunnen dit algoritme verder uitwerken in de precieze stappen die de automaat zou moeten maken om op deze manier een doolhof te doorzoeken. Hiervoor is ook een programma geschreven om een robot te laten rondlopen door een eenvoudig doolhof op `https://robot.maarse.xyz/`. De JavaScript-code staat in Appendix A.1. Om de verschillende toestanden van de automaat te bepalen, zijn een aantal eigenschappen van belang, daarom introduceren we de volgende variabelen:

- zeg s geeft aan uit welke richting we de laatste stap vandaan kwamen.
  - als $s = 0$ dan was de laatst gezette stap naar het noorden
  - als $s = 1$ dan was de laatst gezette stap naar het zuiden
  - als $s = 2$ dan was de laatst gezette stap naar het oosten
  - als $s = 3$ dan was de laatst gezette stap naar het westen.
- zeg k geeft aan met welk deel van het algoritme we momenteel bezig zijn.
  - als $k = 0$ dan zijn we bezig met het deel van het algoritme waarin we eerst zover mogelijk naar het noorden lopen totdat we bij de muur aankomen, en vervolgens weer terugkeren naar de zuidmuur
  - als $k = 2$ dan zijn we bezig met het deel van het algoritme waarin we de muur volgen. Dit doen we met de muur aan de linkerkant. Dat is in dit geval met de klok mee.

We kunnen nu de verschillende toestanden van de automaat definiëren, namelijk $Q = \{q_{s,k} \mid s = \{0,1,2,3\} \land k = \{0,2\}\}$. Hierdoor zouden we kunnen zeggen dat we 8 toestanden nodig hebben, maar toestanden $q_{2,0}$ en $q_{2,2}$ en toestanden $q_{3,0}$ en $q_{3,2}$ hebben hetzelfde gedrag, dus we kunnen ze samenvoegen: $q_{2,0} = q_{2,2}$ en $q_{3,0} = q_{3,2}$. Hierdoor zijn er dus 6 toestanden in deze automaat. Dit is logisch, omdat je altijd als je een stap naar het westen hebt gezet, je de muur aan het volgen bent, en nooit in de situatie bent waarbij je zo ver mogelijk naar het noorden en vervolgens weer terug naar het zuiden aan het lopen bent (want dan is je laatste stap altijd naar het noorden of zuiden geweest). Op de volgende pagina zie je een schematische weergave van hoe de automaat zou werken. Vanaf start, kun je aan de hand van de condities die je steeds ziet, je door de verschillende toestanden verplaatsen, waardoor je zoals weergegeven in Figuur 15 door het doolhof loopt.

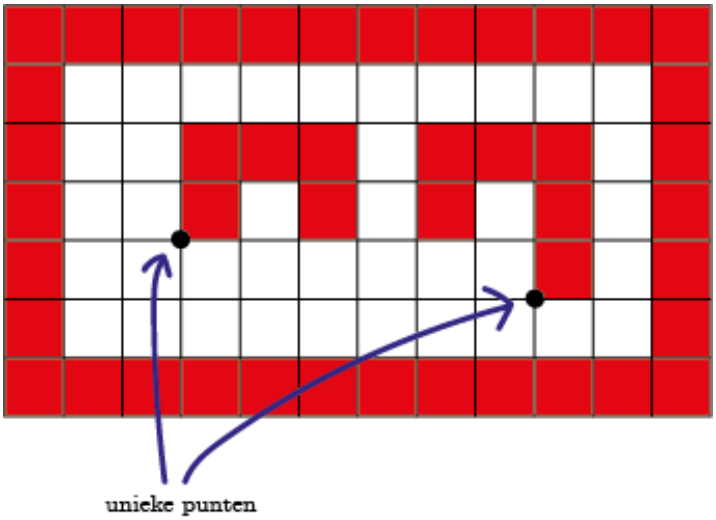

start
Toestand:
k=0
s=0
Condities:
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
- geen muur zuid
Actie:
- ga "Zuid"
Toestand:
k=0
s=1
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur west
Actie:
- ga "West"
Condities:
- muur zuid
- muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
- muur zuid
- geen muur west
Actie:
- ga "West"
Condities:
- muur noord
- muur zuid
- muur west
- geen muur oost
Actie:
- ga "Oost"
Toestand:
k=0/2
s=3
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Conditites:
- muur zuid
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur zuid
- muur noord
- geen muur west
Actie:
- ga "West"
Condities:
- muur zuid
- muur noord
- muur west
- geen muur oost
Actie:
- ga "Oost"
Toestand:
k=0/2
s=2
Condities:
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Toestand:
k=2
s=0
Condities:
- geen muur west
Actie:
- ga "West"
Condities:
- muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur west
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur west
- muur noord
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur noord
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur noord
- muur oost
- muur zuid
- geen muur west
Actie:
- ga "West"
Toestand:
k=2
s=1
Condities:
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur oost
- muur zuid
- geen muur noord
Actie:
- ga "Noord"

6 Doolhof doorzoeken als unieke punten gemarkeerd zijn

We kunnen nu een eenvoudig doolhof doorzoeken, maar natuurlijk is niet elk doolhof eenvoudig. Om een doolhof met binnenmuren te doorzoeken, moeten we eerst definiëren wat het unieke punt van een binnenmuur is.

Definitie 6.1 (uniek punt van een binnenmuur). Laat D een doolhof zijn, en laat X een binnenmuur in D zijn. Laat R de verzameling van punten op de rand van X zijn. Het *unieke punt* P_X van X is het punt (x_0, y_0) in R, waarvoor geldt $y_0 < y \vee (x_0 < x \wedge y_0 = y)$ voor alle $(x, y) \neq (x_0, y_0)$ in R.

Opmerking: bovenstaande definitie houdt dus in dat het unieke punt het meest westelijke punt onder de meest zuidelijke punten op de rand van een binnenmuur is, zie Figuur 16. De buitenmuur heeft geen uniek punt.

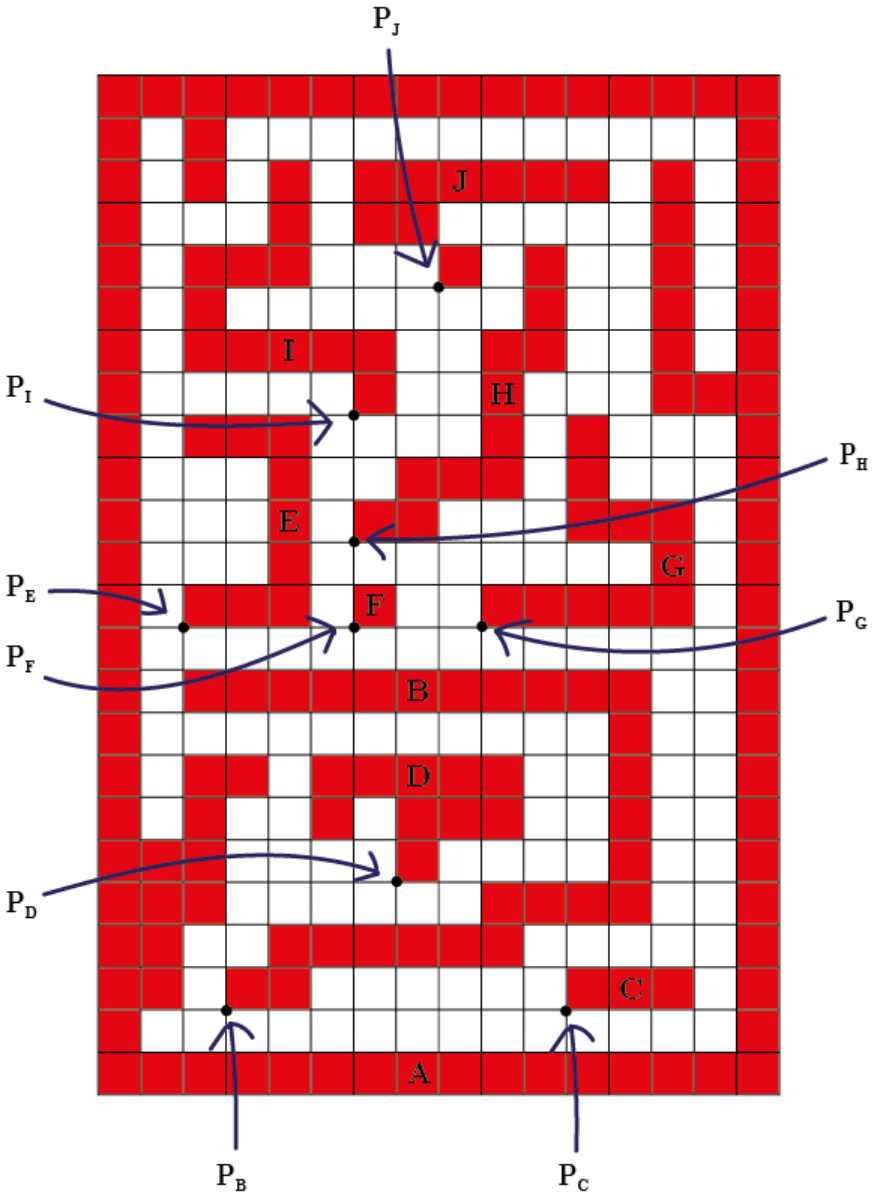


Figuur 16: unieke punten van dit doolhof

Vervolgens willen we een partiële ordening maken op de muren van het doolhof.

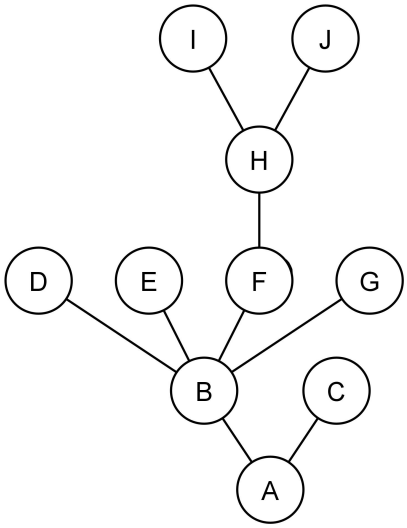
Definitie 6.2 (partiële ordening op de muren van een doolhof). Laat D een doolhof zijn en laat X en Y twee muren zijn van D. Dan definiëren we een partiële ordening op X en Y door: $X < Y \iff X$ is niet de buitenmuur van D en Y is de eerste muur die bereikt wordt als je je in een rechte lijn naar het zuiden verplaatst vanuit het unieke punt P_X (het unieke punt van binnenmuur X).

Op de volgende pagina in Figuur 17 zie je een doolhof met daarin alle muren en zijn unieke punten gemarkeerd.



Figuur 17: een doolhof met de unieke punten gemarkeerd

Van de partiële ordening op de muren van een doolhof kunnen we een schematische boom maken, zie Figuur 18. Dit kan doordat elke muur exact één muur heeft die groter is, behalve de buitenmuur. Daarom is de buitenmuur de wortel van de boom.



Figuur 18: partiële ordening op de muren van het doolhof uit Figuur 17, hierbij staat de grootste muur onder.

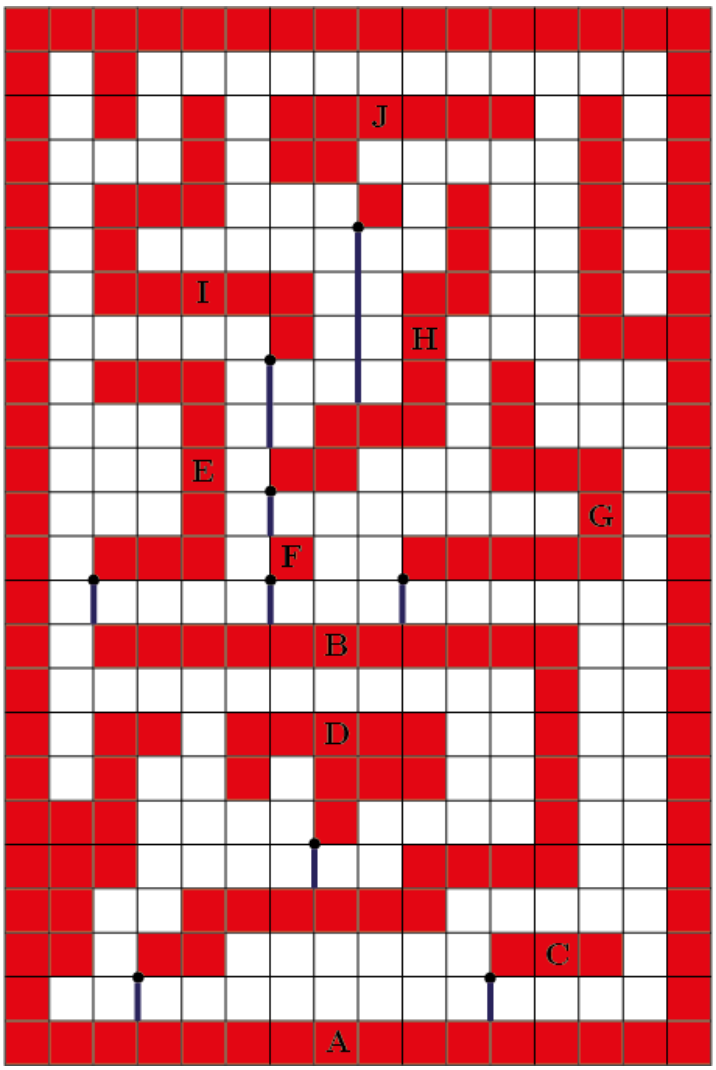
Definitie 6.3 (uniek$_{NO}$, uniek$_{NW}$, uniek$_{ZO}$, uniek$_{ZW}$). Laat D een doolhof zijn. Voor een vakje X in D definiëren we:

- X is *uniek*$_{NO}$ als het noordoostelijke punt van X een uniek punt is
- X is *uniek*$_{NW}$ als het noordwestelijke punt van X een uniek punt is
- X is *uniek*$_{ZO}$ als het zuidoostelijke punt van X een uniek punt is
- X is *uniek*$_{ZW}$ als het zuidwestelijke punt van X een uniek punt is.

Definitie 6.4 (extra muur). Laat D een doolhof zijn met buitenmuur X_0 en n binnenmuren $X_1, ..., X_n$, waarbij $n \in \mathbb{N}$ en $n \geq 1$. Laat $P_{X_0}, ..., P_{X_n}$ de unieke punten van de muren van D zijn, met P_{X_0} het unieke punt van de buitenmuur X_0 en $P_{X_1}, ..., P_{X_n}$ de unieke punten van respectievelijk de binnenmuren $X_1, ..., X_n$. We hebben een partiële ordening op $X_0, ..., X_n$ volgens Definitie 6.2. We kunnen nu *extra muren* toevoegen aan D door gebruik te maken van deze partiële ordening. Een *extra muur* is een denkbeeldige muur van uniek punt $P_{X_i}, 1 \leq i \leq n$ naar het zuiden tot aan de eerste muur X_j, waarbij geldt $1 \leq j \leq n$ en $X_i < X_j$.

Opmerking: een extra muur ziet er anders uit dan een gewone muur in een doolhof, doordat een extra muur niet uit vakjes bestaat in tegenstelling tot een gewone muur. Een extra muur bestaat uit randen van vakjes.

Merk ook op dat aan elk doolhof net zoveel extra muren kunnen worden toegevoegd als dat er binnenmuren in het doolhof zijn. Dit komt doordat elke binnenmuur exact één muur heeft die groter is volgens de partiële ordening uit Definitie 6.2. Alleen de buitenmuur heeft geen grotere muur. De extra muren van het doolhof in Figuur 17 zijn toegevoegd in Figuur 19.



Figuur 19: in donkerblauw zijn extra muren toegevoegd aan het doolhof uit Figuur 17

Stelling 6.5. *Laat D een doolhof zijn, waarin alle extra muren toegevoegd zijn. Dan is D een eenvoudig doolhof.*

Bewijs. Om dit te bewijzen, moeten we eigenlijk drie dingen bewijzen:

1. elke extra muur heeft exact één muur als bron. Dat dit zo is, volgt uit het feit dat de partiële ordening op de muren een boomstructuur vormt, en dus elke binnenmuur exact 1 muur heeft die groter is dan zijn eigen muur. Dit is te zien in Figuur 18.

2. alle witte vakjes in D zijn na het toevoegen van de extra muren nog steeds met elkaar verbonden. Dit gaan we als volgt bewijzen: we zeggen $Y_0, Y_1, ..., Y_n$ muren van D met Y_0 de buitenmuur en $Y_1, ..., Y_n$ de n binnenmuren van D. We bewijzen eerst dat alle witte vakjes nog steeds verbonden zijn als $n = 0$ en $n = 1$:

   - $n = 0$: als $n = 0$ dan heeft D geen enkele binnenmuur en worden er dus geen extra muren aan D toegevoegd. In de definitie van een doolhof staat dat alle witte vakjes verbonden moeten zijn, dus voor D met alle extra muren

toegevoegd (wat dus geen extra muren zijn) zijn alle witte vakjes nog steeds verbonden.

- $n = 1$: als $n = 1$ dan heeft D één binnenmuur Y_1. We weten dat we een lus kunnen maken om Y_1, oftewel er bestaat een pad over witte vakjes, waardoor we vanaf willekeurig wit vakje X een pad om binnenmuur Y_1 heen kunnen lopen. Hieruit volgt dat we dus langs twee kanten van Y_1 kunnen lopen om van X naar een ander willekeurig vakje Z te komen. Als we dan dus de extra muur toevoegen vanuit het unieke punt P_{Y_1} naar de buitenmuur, dan sluiten we een van deze twee mogelijke paden van X naar Z af, maar de andere die langs de andere kant van de muur gaat, is dan nog steeds toegankelijk. Alle witte vakjes blijven daardoor na het toevoegen van de extra muur verbonden.

We weten dus dat het geldt voor $n = 0$ en $n = 1$. Nu willen we ook dat het geldt voor n binnenmuren voor een willekeurige $n \in \mathbb{N}$, $n > 1$. We weten dat we uiteindelijk net zoveel extra muren moeten toevoegen aan D als dat we binnenmuren hebben. Elke keer als we dus een extra muur toevoegen aan D, verkleinen we het totaal aantal binnenmuren in D met 1. Als we dus $n - 1$ extra muren hebben toegevoegd aan D (met n binnenmuren) is er nog precies 1 binnenmuur over. Nu geldt hetzelfde argument weer dat we voor $n = 1$ hebben gebruikt: er is nog een lus om die laatste binnenmuur te maken waarbij er dus nog langs twee kanten van deze laatste binnenmuur mogelijke paden van een wit vakje X naar een willekeurig ander wit vakje Z te maken zijn. Als we dus die laatste extra muur toevoegen, dan sluiten we één van deze twee mogelijke kanten af, en is er nog steeds een pad te vinden van elk willekeurig wit vakje X naar elk ander willekeurig wit vakje Z. Alle witte vakjes zijn dan dus nog steeds verbonden.

3. alle witte vakjes in D zijn enkelvoudig samenhangend, oftewel voor alle lussen L in D geldt dat L geen muur in D omsluit. Dit is met soortgelijke argumenten te bewijzen als het feit dat alle witte vakjes nog steeds verbonden zijn.

□

Door alle extra muren als een muur aan een doolhof toe te voegen, wordt het doolhof dus omgezet naar een eenvoudig doolhof, zie als voorbeeld Figuur 19. Nu kunnen we in principe het algoritme en schema van het eenvoudige doolhof gebruiken om het volledige doolhof te doorzoeken. Echter zit hier nog een probleem: de extra muren zijn denkbeeldig, en niet te zien in het doolhof. We kunnen dus niet zomaar deze extra muren volgen, maar moeten dus tijdens het doorzoeken van het doolhof controleren waar zo'n extra muur is.

Stelling 6.6. *Laat D een doolhof zijn, waarin de unieke punten gemarkeerd zijn. Dan is er een eindige automaat die D volledig kan doorzoeken, oftewel alle witte vakjes in D bezoeken door zichzelf door D te verplaatsen.*

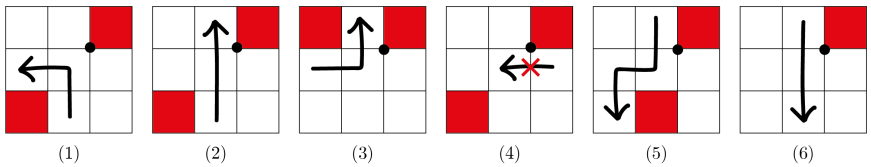
Bewijs. We weten dat we D kunnen omzetten tot een eenvoudig doolhof door vanuit alle unieke punten van de binnenmuren van D extra muren toe te voegen. In principe kunnen we dan het algoritme van het doorzoeken van een eenvoudig doolhof gebruiken om D volledig te doorzoeken. Echter zijn alleen de unieke punten gemarkeerd in D, en zijn de extra muren denkbeeldig, en dus niet te zien. Het is dus belangrijk om in verschillende situaties te controleren waar zich eventueel een extra muur bevindt als je door het doolhof loopt. Dit doen we door in verschillende situaties te kijken of we een uniek punt tegenkomen. Zeg X is een wit vakje uit doolhof D. In de volgende situaties is het belangrijk te controleren of er ergens een extra muur is, om dan langs die extra muur te lopen:

1. X is $uniek_{NO}$. Dit kan in verschillende situaties voorkomen (zie Figuur 20), maar er is eigenlijk maar één situatie waarvoor we een nieuwe stap moeten toevoegen aan het algoritme van het eenvoudige doolhof doen. En dat is in het geval dat de volgende voorwaarden gelden:

   - je laatste stap was naar het zuiden
   - je bent in de situatie waarin je de muur aan het volgen bent (je bent dus niet bezig met zo ver mogelijk naar het noorden om vervolgens weer terug naar het zuiden te lopen)
   - er is geen muur in het zuiden

   Als bovenstaande voorwaarden gelden dan zet je een stap naar het zuiden en ga je naar Stap (a). Als deze voorwaarden niet gelden dan kun je het unieke punt in het noordoosten negeren, en gewoon de stappen van het algoritme van het eenvoudige doolhof blijven volgen. Dit is omdat:

   - als je laatste stap was naar het:
     - noorden, en je bent de muur aan het volgen, dan kun je gewoon doorgaan met het volgen van de muur. Je volgt namelijk een andere muur dan de muur waarvan het unieke punt in het noordoosten is. Er is geen situatie waarbij je daarmee eventueel door de extra muur heen kan lopen, wel kan het nog voorkomen dat je bij het volgen van de muur volgens het algoritme van het eenvoudige doolhof door een andere extra muur heen loopt. Deze situatie wordt omschreven in Situatie 3. Als je in de situatie bent waarbij je zover mogelijk naar het noorden aan het lopen bent, om vervolgens weer naar het zuiden te lopen, dan kun je hier ook gewoon mee doorgaan. Er bevindt zich geen muur in het noorden als je in een $uniek_{NO}$ vakje staat, dus je zet een stap naar het noorden, en gaat door. Er is hierbij geen risico dat je door de extra muur vanuit het unieke punt heen loopt.
     - oosten, dan ben je een muur aan het volgen die niet dezelfde muur is als de muur van het unieke punt in het noordoosten (dan zou het immers geen uniek punt aan die muur zijn, omdat er dan sowieso nog een westelijker punt aan die muur is). Je kunt gewoon doorgaan met het volgen van deze muur, er is geen risico dat je door de extra muur vanuit het unieke punt in het noordwesten heen loopt.
     - westen. Dit komt niet voor, omdat je dan door de extra muur heen zou zijn gelopen die vanuit het unieke punt in het noordoosten loopt, en dit kan niet.
   - als je in de situatie bent waarbij je zover mogelijk naar het noorden aan het lopen bent om vervolgens naar het zuiden te lopen, dan kun je hier gewoon mee doorgaan zonder dat er een risico is dat je door de extra muur vanuit het unieke punt in het noordoosten heen loopt.
   - als er wel een muur zou zijn in het zuiden, dan kan je gewoon overgaan op het volgen van deze zuidmuur. Je hebt de extra muur vanuit het unieke punt in het noordoosten dan eigenlijk al gevolgd. Deze extra muur is dan maar één vakje lang. Merk op dat je nu wel een zuidmuur aan het volgen bent en dus eerst weer zover mogelijk naar het noorden moet lopen om vervolgens weer naar het zuiden terug te keren, voordat je verder gaat met het volgen van de muur.



Figuur 20: (1) laatste stap was noord, je bent de muur aan het volgen, je komt niet langs de extra muur
(2) laatste stap was noord, je bent zo ver mogelijk naar het noorden aan het lopen, om vervolgens weer naar het zuiden te lopen, je komt niet langs de extra muur
(3) laatste stap was oost, je bent de muur aan het volgen, je komt niet langs de extra muur
(4) laatste stap was west, kan niet, dan zou je door de extra muur heen zijn gegaan
(5) laatste stap was zuid, je bent de muur aan het volgen en komt gelijk bij een zuidmuur aan. Je volgt deze verder (en hebt de extra muur van lengte 1 gevolgd)
(6) laatste stap was zuid, je moet nu de extra muur gaan volgen. Hiervoor wordt een stap toegevoegd aan het algoritme van het eenvoudige doolhof.

Dus alleen in de situatie dat de genoemde voorwaarden gelden, ga je naar de volgende Stap:

(a) Je komt in deze stap als je zojuist een stap naar het zuiden hebt gezet, en de muur aan het volgen was. De muur die je aan het volgen was, kun je nu niet verder volgen, omdat we dan door de extra muur heen zouden lopen die vanuit het unieke punt in het noordoosten loopt. We willen dus die extra muur volgen. De volgende situaties kunnen plaatsvinden:

i. er bevindt zich geen muur in het oosten, en geen muur in het zuiden. Dit betekent dat de extra muur nog steeds aan de oostkant van je loopt. We willen deze volgen. We zetten een stap naar het zuiden, en gaan terug naar Stap (a).

ii. er bevindt zich een muur in het oosten, maar geen muur in het zuiden. De extra muur die we aan het volgen waren, is nu geëindigd op de muur die zich in het oosten bevindt. We gaan nu dus verder met het volgen van deze oostmuur. We zetten een stap naar het zuiden, en gaan naar Stap 6 van het algoritme van het eenvoudige doolhof.

iii. er bevindt zich een muur in het zuiden. De extra muur die we aan het volgen waren, is nu geëindigd op de muur die zich in het zuiden bevindt. We hoeven de extra muur dus niet meer te volgen en gaan verder met het volgen van de zuidmuur. Omdat we een zuidmuur volgen, moeten we wel eerst weer helemaal naar het noorden lopen, om vervolgens weer terug naar het zuiden te lopen en de muur verder te volgen. We zetten een stap naar het noorden (dit kan omdat we zeker weten dat er geen muur in het noorden is, we hebben immers de laatste stap naar het zuiden gezet), en gaan naar Stap 1 van het algoritme van het eenvoudige doolhof.

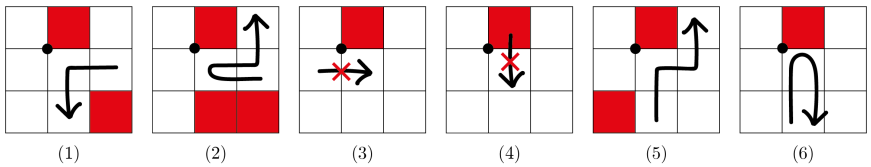
2. X is $uniek_{NW}$. Dit kan in verschillende situaties (zie Figuur 21) voorkomen, maar er is eigenlijk maar één situatie waarvoor we drie nieuwe stappen moeten toevoegen aan wat we in het algoritme van het eenvoudige doolhof doen. En dat is in het geval dat de volgende voorwaarden gelden:

- je laatste stap was naar het noorden

- je bent in de situatie waarin je bezig bent met zo ver mogelijk naar het noorden lopen om vervolgens weer terug naar het zuiden te lopen (dus niet de muur aan het volgen)

Als bovenstaande voorwaarden gelden dan zet je een stap naar het zuiden en ga je naar Stap (a). Als deze voorwaarden niet gelden dan kun je het unieke punt in het noordwesten negeren, en gewoon de stappen van het algoritme van het eenvoudige doolhof blijven volgen. Dit is omdat:

- als je laatste stap was naar het:
  - westen, dan ben je een muur aan het volgen wat niet dezelfde muur is als de muur van het unieke punt in het noordwesten (dan zou het immers geen uniek punt aan die muur zijn, omdat er dan sowieso nog een zuidelijker punt aan die muur is). Nu zijn er twee situaties:
    * Er bevindt zich geen muur in het zuiden. Je kunt gewoon doorgaan met het volgen van de muur die je al aan het volgen was. Er is geen risico dat je door de extra muur vanuit het unieke punt in het noordoosten heen loopt. Eventueel kan je na een tijdje deze muur te hebben gevolgd wel weer in de situatie komen dat je moet uitkijken dat je niet door deze extra muur heen loopt. Echter, heb je dan altijd tussendoor een zuidmuur gevolgd, en ben je dus zoveel mogelijk naar het noorden gaan lopen, om vervolgens weer terug te hebben gelopen. Je komt dan in een situatie waarbij je wél aan alle drie de voorwaarden voldoet en dan dus wel in de nieuwe Stap terecht komt.
    * Er bevindt zich wel een muur in het zuiden. Je weet nu dat er een muur in het zuiden en in het noorden is. In het noorden is namelijk sowieso een muur omdat je anders niet in een vakje kan staan dat $uniek_{NW}$ is. In het westen heb je de extra muur dat één vakje lang is. We zetten een stap naar het oosten, en gaan door met het volgen van de noordmuur: we gaan naar Stap 4 van het algoritme van het eenvoudige doolhof. Hiermee hebben we de extra muur dus gevolgd.
  - oosten. Dit komt niet voor, omdat je dan door de extra muur heen zou zijn gelopen die vanuit het unieke punt in het noordwesten loopt, en dit kan niet.
  - zuiden. Dit komt niet voor, omdat als je in een vakje staat dat $uniek_{NW}$ is dat je dan altijd een muur ten noorden van je hebt. Je laatste stap kan dus nooit naar het zuiden zijn geweest.
- als je in de situatie bent waarbij je de muur aan het volgen bent (en je laatste stap was naar het noorden), dan was je een muur aan de westkant aan het volgen, en zou je hier eigenlijk een bocht naar het westen moeten maken om deze muur te blijven volgen. Let op: er kan geen muurvakje in het westen zitten, want dan zou je niet op een vakje kunnen staan dat $uniek_{NW}$ was. Dan is er namelijk nog een zuidelijker en westelijker punt. Die situatie komt dus niet voor, maar het kan wel zo zijn dat het vakje ten zuidwesten van het vakje waar je staat een muurvakje is die je aan het volgen was. In dit geval ga je verder met het volgen van de noordmuur en zet een stap naar het oosten. Hiermee hebben we dan de extra muur van lengte 1 gevolgd.



Figuur 21: (1) laatste stap was west, je bent de muur aan het volgen en er zit geen muur in het zuiden, je komt niet langs de extra muur
(2) laatste stap was west, je bent de muur aan het volgen en er zit een muur in het zuiden, je volgt de noordmuur verder (en hebt de extra muur van lengte 1 daarmee gevolgd)
(3) laatste stap was oost, kan niet, dan zou je door de extra muur heen zijn gegaan
(4) laatste stap was zuid, kan niet, daar moet een muurvakje zitten als we op een vakje staan dat $uniek_{NW}$ is
(5) laatste stap was noord, je bent de muur aan het volgen en komt bij de noordmuur aan. Je volgt deze verder (en hebt de extra muur van lengte 1 gevolgd)
(6) laatste stap was noord, je bent bezig met zo ver mogelijk naar het noorden lopen om vervolgens weer terug naar het zuiden te lopen, je moet terug naar het zuiden lopen, maar rekening houden met de extra muur. Hiervoor worden nog drie stappen toegevoegd aan het algoritme van het eenvoudige doolhof.

Dus alleen in de situatie dat de genoemde voorwaarden gelden, ga je naar de volgende Stap:

(a) Je komt in deze stap als je zojuist een stap naar het noorden hebt gezet, en bezig was met zover mogelijk naar het zuiden lopen om vervolgens weer zoveel mogelijk naar het noorden te lopen. We staan nu in een vakje dat $uniek_{NW}$ is en weten dus dat er een extra muur aan onze westkant loopt. We weten echter nog niet of deze extra muur helemaal doorloopt tot de zuidmuur waar we vandaan komen, of dat deze muur eerder stopt bij een andere muur aan de westkant. We moeten dus nog controleren of we deze muur nu moeten volgen, of dat we gewoon terug naar het zuiden moeten gaan om vervolgens de zuidmuur verder te volgen. We willen hiervoor eerst zover mogelijk terug naar het zuiden lopen. De volgende situaties kunnen plaatsvinden:
   i. er bevindt zich geen muur in het zuiden. Zet een stap naar het zuiden en ga terug naar Stap (a).
   ii. er bevindt zich wel een muur in het zuiden, maar geen muur in het westen. Zet een stap naar het noorden en ga naar Stap (b).
   iii. er bevindt zich een muur in het zuiden en een muur in het westen. Zet een stap naar het noorden en ga naar Stap (c).

(b) We zijn zojuist helemaal terug naar het zuiden gelopen en hebben daar gezien dat er geen muur aan de westkant zit. We willen nu elke stap die we zetten controleren of we een muur aan de westkant zien. Namelijk, zodra we daar een muur zien, weten we dat de extra muur vanuit het unieke punt in het noordwesten niet helemaal tot het zuiden loopt, maar dat deze extra muur op een andere muur (of deel van de muur) aan de westkant strandt. We willen in de situatie dat dat zo is, terug naar het zuiden om de zuidmuur te volgen. Als we eerder terug aankomen in het vakje dat $uniek_{NW}$ is dan weten we dat de extra muur wel helemaal tot aan het zuiden loopt en dat we deze dus moeten volgen. De volgende situaties kunnen plaatsvinden:
   i. Er bevindt zich geen muur west en geen muur noord. Zet een stap naar het noorden en ga terug naar Stap (b).

ii. Er bevindt zich een muur in het westen en geen muur in het noorden. We weten nu dus dat de extra muur niet helemaal tot aan het zuiden loopt, en we deze nog niet moeten volgen. We willen terugkeren naar het zuiden om daar vervolgens de zuidmuur verder te volgen. We zetten een stap naar het zuiden en gaan naar Stap 2 van het algoritme van het eenvoudige doolhof.

iii. Er bevindt zich een muur in het noorden. We zijn nu teruggekomen in het vakje dat $uniek_{NW}$ is, zonder dat we nog een muur aan de westkant hebben gezien. We weten nu dus dat de extra muur vanuit het unieke punt in het noordwesten helemaal tot aan de zuidmuur loopt, en dat we deze muur dus moeten volgen. We zijn zojuist weer naar het noorden gelopen, en hebben deze muur dus al gevolgd, en willen nu deze noordmuur gaan volgen. We zetten een stap naar het oosten (we weten zeker dat er geen muur is in het oosten, omdat als er wel een muur was, dan was het vakje waar we staan niet $uniek_{NW}$, omdat er dan nog een zuidelijker punt aan de muur was), en gaan naar Stap 4 van het algoritme van het eenvoudige doolhof.

(c) We zijn zojuist helemaal terug naar het zuiden gelopen en hebben daar gezien dat er een muur aan de westkant zit. We willen nu elke stap die we zetten controleren of we een muur aan de westkant zien. Namelijk, zodra we daar geen muur zien, kunnen we naar Stap (b) gaan om te controleren tot waar de extra muur loopt. De volgende situaties kunnen plaatsvinden:

i. Er bevindt zich geen muur west en geen muur noord. We komen nu dus in de situatie van Stap (b): we hebben een keer geen westmuur gezien bij het lopen vanaf de zuidmuur naar het noorden. Zet een stap naar het noorden en ga terug naar Stap (b).

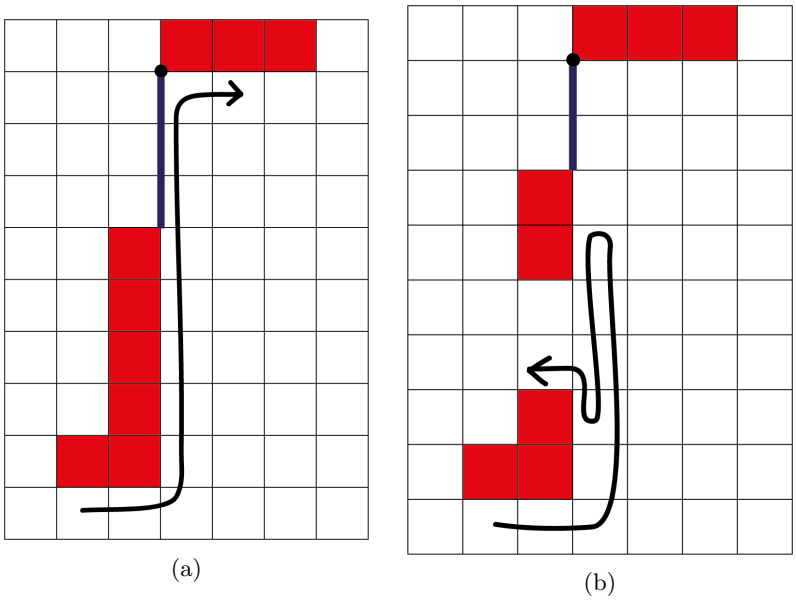
ii. Er bevindt zich wel een muur in het westen en geen in het noorden. We zijn nu nog steeds in dezelfde situatie als we al waren: er bevindt zich een aan de westkant een muur die nog aan de zuidmuur vastzit. We zetten een stap naar het noorden, en gaan terug naar Stap (c).

iii. Er bevindt zich een muur in het noorden. We zijn nu teruggekomen in het vakje dat $uniek_{NW}$ is, terwijl we tot aan de laatste stap steeds een muur aan de westkant hebben gezien. Deze muur zit vast aan de zuidmuur. We weten nu dus dat we niet eerder hadden kunnen afslaan en de extra muur vanuit het unieke punt in het noordwesten moeten volgen. We hebben deze muur dus al gevolgd (want we staan bij de noordmuur), en willen nu deze noordmuur gaan volgen. We zetten een stap naar het oosten (we weten zeker dat er geen muur is in het oosten, omdat als er wel een muur was, dan was het vakje waar we staan niet $uniek_{NW}$, omdat er dan nog een zuidelijker punt aan de muur was), en gaan naar Stap 4 van het algoritme van het eenvoudige doolhof.

3. X is een wit vakje met aan de westkant een muur, en je hebt zojuist een stap naar het noorden gezet terwijl je de muur aan het volgen was. Je bent nu dus langs die westmuur aan het lopen. In het eenvoudige doolhof zouden we deze muur blijven volgen naar het noorden, en zodra er geen muur meer aan de westkant is naar het westen lopen. Echter, kunnen we nu niet zomaar naar het westen lopen, omdat we niet weten of hier een eventuele extra muur ligt (zie Figuur 22). We introduceren drie nieuwe stappen voor om te controleren of er een extra muur is. Je komt in Stap (a) als de volgende voorwaarden gelden:

- je laatste stap was naar het noorden

- je bent in de situatie waarin je de muur aan het volgen bent (je bent dus niet bezig met zo ver mogelijk naar het noorden om vervolgens weer terug naar het zuiden te lopen)
- je hebt een muur aan de westkant
- je hebt geen muur aan de noordkant
- je staat niet in een $uniek_{NW}$ vakje (wat vanzelfsprekend is als er geen muur aan de noordkant zit).



(a) (b)

Figuur 22: twee gevallen waarbij we controleren of er een extra muur vanuit een uniek punt eindigt op de westmuur die we nu aan het volgen zijn. In geval (a) komen we erachter dat er inderdaad een extra muur eindigt op de westmuur die we aan het volgen zijn, en moeten we dus deze extra muur volgen. In geval (b) komen we terwijl we aan het controleren zijn of er een extra muur op de westmuur die we aan het volgen zijn een andere westmuur tegen. Hierdoor weten we dat als er eventueel een $uniek_{NW}$ vakje in het noorden zit, dat deze dan tot een ander muurstuk loopt, en niet tot het deel dat we nu aan het volgen zijn.

Als bovenstaande voorwaarden voldoen, dan kom je in een Stap (a) waarbij je wilt gaan controleren of er ergens ten noorden van je een vakje is die $uniek_{NW}$ is, om zo te ontdekken of er een extra muur is die je moet volgen.

(a) Je bent in een situatie waarbij je de muur aan het volgen was met de muur aan de westkant, en wilt nu erachter komen of we deze muur kunnen blijven volgen of dat er een extra muur is die we moeten volgen. De volgende situaties kunnen plaatsvinden:

i. er is een muur in het westen en geen muur in het noorden. Zet een stap naar het noorden en ga terug naar Stap (a).

ii. er is geen muur in het westen en geen muur in het noorden. Als we in een eenvoudig doolhof de muur zouden volgen dan zouden we nu een stap naar het westen zetten. Maar nu willen we controleren of er ergens een vakje ten noorden van ons is dat $uniek_{NW}$ is. We zetten een stap naar het noorden en gaan naar Stap (b).

iii. er is geen muur in het westen, maar wel een muur in het noorden. We zijn niet in een $uniek_{NW}$ vakje. Nu weet je dat er zich geen extra muur bevindt. Je kan dus gewoon verder gaan met de muur volgen die we al aan het volgen waren. We zetten een stap naar het westen, en gaan naar Stap 3 van het algoritme van het eenvoudige doolhof.

iv. er is geen muur in het oosten en we zijn in een $uniek_{NW}$ vakje (merk op dat er zich dus ook een muur ten noorden van ons bevindt, want dat is altijd het geval als je in een $uniek_{NW}$ vakje staat). Nu weten we dat er inderdaad een extra muur naast ons loopt die eindigt op de muur die we zojuist aan het volgen waren. We hebben deze extra muur al gevolgd door naar het noorden te lopen, en willen nu de muur volgen waar we bij aangekomen zijn, die ten noorden van waar we staan. We zetten een stap naar het oosten, en gaan naar Stap 4 van het algoritme van het eenvoudige doolhof.

v. er is geen muur in het oosten, maar wel een muur in het noorden en in het westen. We weten nu dat de muur ten westen van ons niet eindigt in een $uniek_{NW}$ vakje (dat kan niet als er een muur in het westen is), en er dus geen extra muur loopt. We gaan gewoon verder met het volgen van deze muur. We zetten een stap naar het oosten, en gaan naar Stap 4 van het algoritme van het eenvoudige doolhof.

vi. er is geen muur in het zuiden, maar wel in het noorden, westen en oosten. We weten nu dat de muur ten westen van ons niet eindigt in een $uniek_{NW}$ vakje (dat kan niet als er een muur in het westen is), en er dus geen extra muur loopt. We gaan gewoon verder met het volgen van deze muur. We zetten een stap naar het zuiden, en gaan naar Stap 6 van het algoritme van het eenvoudige doolhof.

(b) Je komt in deze stap als je in Stap (a) geen muur meer aan de westkant had. We willen in deze stap controleren of er ergens ten noorden van ons een vakje is dat $uniek_{NW}$ is, zodat er dus eventueel een extra muur eindigt op de muur die we eerst volgden. Dan moeten we namelijk die extra muur volgen. De volgende situaties kunnen plaatsvinden:

i. Er is geen muur in het westen en geen muur in het noorden. Je wilt nu naar het noorden blijven lopen om te controleren of er ergens een vakje $uniek_{NW}$ is. Zet een stap naar het noorden en ga terug naar Stap (b).

ii. Er is een muur in het westen, maar geen muur in het noorden. We hebben nu een andere muur gevonden (of in ieder geval een ander deel van de muur) dan de muur waar we vandaan komen. Je weet nu dat er geen extra muur is die we zouden moeten volgen. Als er toch ergens een $uniek_{NW}$ vakje ten noorden van ons is, dan eindigt de extra muur die daar vandaan komt op de nieuwe muur (of het nieuwe muurdeel) dat we hebben gevonden, of nog op een muur(deel) dat zich nog noordelijker bevindt. We willen nu dus teruglopen naar de muur die we zojuist aan het volgen waren, om deze weer verder te volgen. Zet een stap naar het zuiden en ga naar Stap (c).

iii. Er is een muur in het noorden, geen muur in het zuiden en we staan niet in een $uniek_{NW}$ vakje. We weten nu dat er geen extra muur is die we zouden moeten volgen. We hebben namelijk een muur ten noorden van ons gevonden terwijl we geen uniek punt in het noordwesten hebben gevonden. Hierdoor weten we dat hier geen extra muur loopt. We willen nu teruglopen naar de muur die we zojuist aan het volgen waren om deze weer verder te volgen. Zet een stap naar het zuiden en ga naar Stap (c).

iv. We staan in een $uniek_{NW}$ vakje. Merk op dat er zich dus een muur ten noorden van ons bevindt, want dat is altijd het geval als je in een $uniek_{NW}$ vakje staat. Merk ook op dat er zich geen muur bevindt in zowel het oosten als het westen, omdat dat ook altijd zo is in een $uniek_{NW}$ vakje (anders was er immers wel een zuidelijker punt aan de muur te vinden, en dit dus geen uniek punt). Nu weten we dat er inderdaad een extra muur naast ons loopt die eindigt op de muur die we zojuist aan het volgen waren. We hebben deze extra muur al gevolgd door naar het noorden te lopen, en willen nu de muur volgen waar we bij aangekomen zijn, die ten noorden van waar we staan. We zetten een stap naar het oosten, en gaan naar Stap 4 van het algoritme van het eenvoudige doolhof.

(c) Je komt in deze Stap als je zojuist naar het noorden bent gelopen om te controleren of er een extra muur was die we moesten volgen, en je hebt ontdekt dat er geen extra muur is die gevolgd moet worden. We willen nu dus teruglopen naar de muur waar we vandaan kwamen, om vanaf daar die muur weer te gaan volgen volgens de stappen in het algoritme van het eenvoudige doolhof. De volgende situaties kunnen plaatsvinden:

i. er bevindt zich geen muur in het zuiden, en geen muur in het westen. In dit geval zijn we nog niet aangekomen bij de muur die we zojuist aan het volgen waren. We zetten een stap naar het zuiden en blijven in Stap (c).

ii. er bevindt zich een muur in het westen, maar geen muur in het noorden (merk op dat er zich nooit een muur in het noorden bevindt, want je bent zojuist naar het zuiden gelopen, dat had niet gekund als daar een muur zou zijn). We hebben de westmuur teruggevonden die we aan het volgen waren voordat we gingen controleren of er een extra muur was. Deze muur willen we dus weer volgen. We zetten een stap naar het noorden, en gaan naar Stap 5 van het algoritme van het eenvoudige doolhof.

□

Het hele algoritme om een doolhof te doorzoeken als de unieke punten al gemarkeerd zijn, is te zien in het schema op Pagina 39. Dit schema is weer een weergave van de eindige automaat die een doolhof kan doorzoeken waarin de unieke punten gemarkeerd zijn. De bijbehorende JavaScript-code staat in Appendix A.2. Het algoritme, en dus ook het bijbehorende schema en de bijbehorende code, bevatten weer geen uitgang. Het idee van het algoritme is weer om het hele doolhof te kunnen doorzoeken, zodat als er een uitgang is, deze sowieso gevonden kan worden. Doordat er geen eindtoestand is in de automaat, blijft de automaat weer oneindig lang doorgaan. Dit kan weer worden opgelost door een uitgang toe te voegen aan het doolhof en in de automaat in elke toestand een controle toe te voegen of we op het huidige vakje op een uitgang staan, zodat de automaat stopt zodra we de uitgang hebben gevonden.

De variabelen s en k in het schema betekenen hierin nog steeds hetzelfde als in het schema van het eenvoudige doolhof. De volgende variabelen zijn hieraan toegevoegd:

- zeg x geeft aan of we een extra muur aan het volgen zijn.
  - als $x = 0$ dan zitten we in de situatie waarin we het algoritme van het eenvoudige doolhof volgen: we volgen geen extra muur.
  - als $x = 1$ dan zitten we in de situatie waarbij we een uniek punt in het noordoosten hebben gezien, en de extra muur hier vanuit moeten volgen.

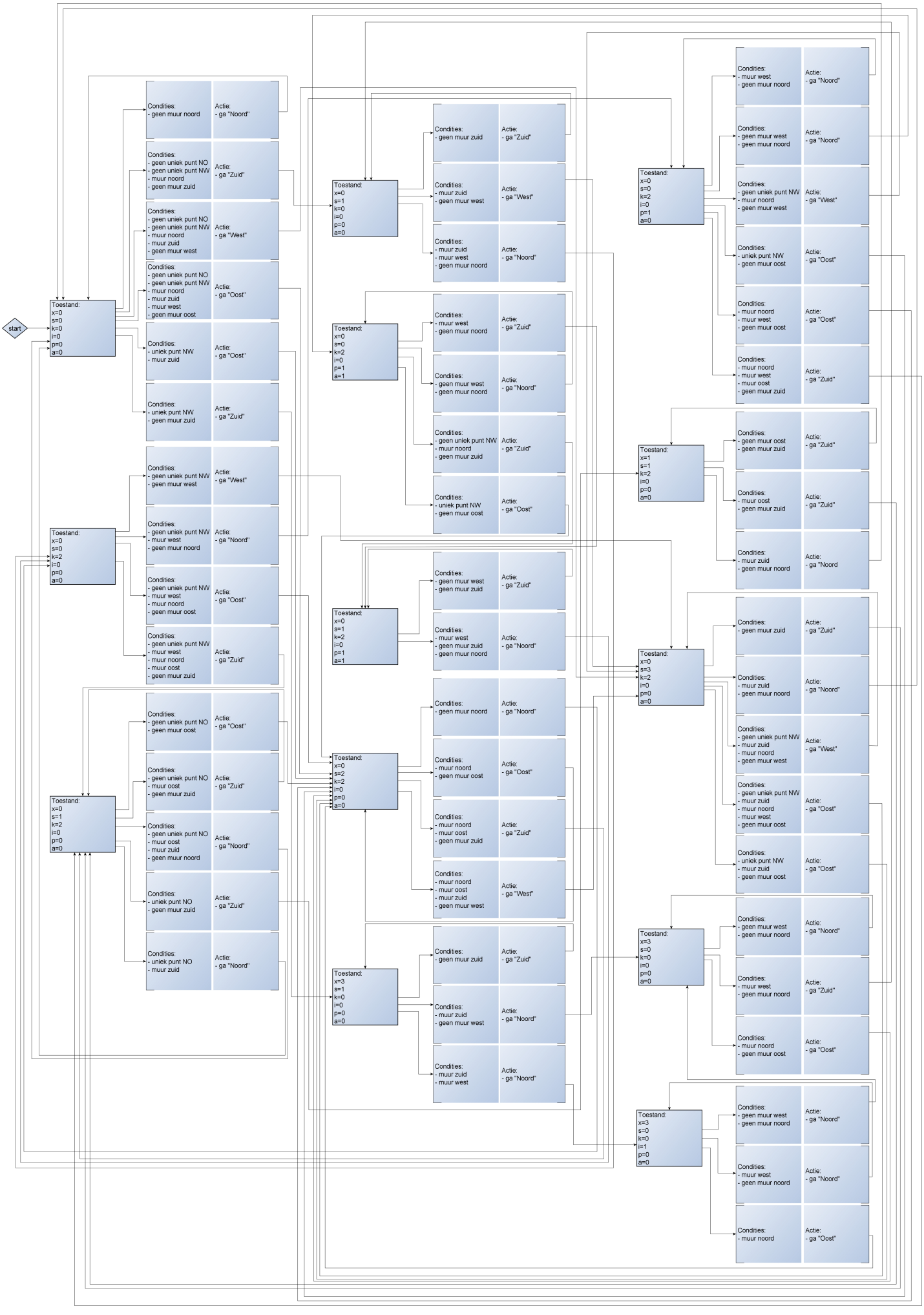
  - als $x = 3$ dan zitten we in de situatie waarbij we een uniek punt in het noordwesten hebben gezien, en moeten controleren of we de extra muur hier vanuit moeten volgen.

- zeg i wordt gebruikt in de situatie waarbij $x = 3$ en we nog moeten controleren of er we de extra muur wel of niet moeten volgen, i heeft te maken met of er wel of niet een muur in het westen is.
  - als $i = 0$ dan hebben we een keer geen muur aan de westkant gezien terwijl we vanaf het zuiden naar het noorden aan het lopen zijn.
  - als $i = 1$ dan is er vanaf het zuiden de hele tijd nog een muur aan de westkant geweest, terwijl we naar het noorden lopen.

- zeg p wordt alleen gebruikt in de situatie dat je de muur aan het volgen bent aan de westkant en zojuist een stap naar het noorden hebt gezet.
  - als $p = 0$ dan zitten we in de situatie waarin we het algoritme van het eenvoudige doolhof volgen: we zijn niet naar het noorden aan het lopen om te controleren of er ergens een extra muur is die we moeten volgen.
  - als $p = 1$ dan zijn we wel aan het controleren of er een extra muur is die we moeten volgen. Zolang we bezig zijn

- zeg a wordt alleen gebruikt in de situatie dat $p = 1$.
  - als $a = 0$ dan is er de hele tijd nog een muur aan de westkant geweest, terwijl we naar het noorden lopen.
  - als $a = 1$ dan is er een keer geen muur aan de westkant geweest, en controleren we wanneer we weer moeten afslaan.

Met bovenstaande variabelen erbij, kunnen we in totaal 7 toestanden toevoegen aan de 6 toestanden uit het schema van het eenvoudige doolhof. Dat het er maar 7 extra zijn en niet 32 ($= 3 * 2 * 2 * 2$, voor de 3 verschillende mogelijke waardes van de variabele x, 2 mogelijkheden van i, 2 mogelijkheden van p en 2 mogelijkheden van a) komt doordat we weer verschillende toestanden die hetzelfde gedrag hebben, kunnen samenvoegen. Dat dit kan komt doordat sommige variabelen eigenlijk alleen in bepaalde situaties nodig zijn, zoals dat $a = 1$ alleen kan voorkomen als $p = 1$. De automaat die het hele doolhof kan doorzoeken terwijl de unieke punten al gemarkeerd zijn heeft dus 13 toestanden. Hiervan is een schema te zien op de volgende pagina.


start
Toestand:
x=0
s=0
k=0
i=0
p=0
a=0
Condities:
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NO
- geen uniek punt NW
- muur noord
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- geen uniek punt NO
- geen uniek punt NW
- muur noord
- muur zuid
- geen muur west
Actie:
- ga "West"
Condities:
- geen uniek punt NO
- geen uniek punt NW
- muur noord
- muur zuid
- muur west
- geen muur oost
Actie:
- ga "Oost"
Condities:
- uniek punt NW
- muur zuid
Actie:
- ga "Oost"
Condities:
- uniek punt NW
- geen muur zuid
Actie:
- ga "Zuid"
Toestand:
x=0
s=1
k=0
i=0
p=0
a=0
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur west
Actie:
- ga "West"
Condities:
- muur zuid
- muur west
- geen muur noord
Actie:
- ga "Noord"
Toestand:
x=0
s=0
k=2
i=0
p=1
a=1
Condities:
- muur west
- geen muur noord
Actie:
- ga "Zuid"
Condities:
- geen muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NW
- muur noord
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- uniek punt NW
- geen muur oost
Actie:
- ga "Oost"
Toestand:
x=0
s=2
k=2
i=0
p=0
a=0
Condities:
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur noord
- muur oost
- muur zuid
- muur west
Actie:
- ga "West"
Toestand:
x=0
s=0
k=2
i=0
p=0
a=0
Condities:
- geen uniek punt NW
- geen muur west
Actie:
- ga "West"
Condities:
- geen uniek punt NW
- muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NW
- muur west
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Condities:
- geen uniek punt NW
- muur west
- muur noord
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Toestand:
x=0
s=1
k=2
i=0
p=0
a=0
Condities:
- geen uniek punt NO
- geen muur oost
Actie:
- ga "Oost"
Condities:
- geen uniek punt NO
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- geen uniek punt NO
- muur oost
- muur zuid
- geen muur noord
Actie:
- ga "Noord"
Condities:
- uniek punt NO
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- uniek punt NO
- muur zuid
Actie:
- ga "Noord"
Toestand:
x=1
s=1
k=0
i=0
p=0
a=0
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur west
Actie:
- ga "West"
Condities:
- muur zuid
- muur west
- geen muur noord
Actie:
- ga "Noord"
Toestand:
x=1
s=0
k=2
i=0
p=1
a=1
Condities:
- muur west
- geen muur noord
Actie:
- ga "Zuid"
Condities:
- geen muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NW
- muur noord
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- uniek punt NW
- geen muur oost
Actie:
- ga "Oost"
Toestand:
x=1
s=0
k=2
i=0
p=1
a=0
Condities:
- muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NW
- muur noord
- geen muur west
Actie:
- ga "West"
Condities:
- uniek punt NW
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur west
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur west
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Toestand:
x=1
s=1
k=2
i=0
p=0
a=0
Condities:
- geen muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur noord
Actie:
- ga "Noord"
Toestand:
x=1
s=1
k=2
i=0
p=1
a=1
Condities:
- geen muur west
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur west
- geen muur zuid
- geen muur noord
Actie:
- ga "Noord"
Toestand:
x=1
s=2
k=2
i=0
p=0
a=0
Condities:
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur noord
- muur oost
- muur zuid
- geen muur west
Actie:
- ga "West"
Toestand:
x=1
s=3
k=0
i=1
p=0
a=0
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur west
Actie:
- ga "Noord"
Condities:
- muur zuid
- muur west
- geen muur noord
Actie:
- ga "Noord"
Toestand:
x=2
s=0
k=2
i=0
p=1
a=0
Condities:
- muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NW
- muur noord
- geen muur west
Actie:
- ga "West"
Condities:
- uniek punt NW
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur west
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur west
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Toestand:
x=2
s=0
k=2
i=0
p=0
a=0
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur noord
Actie:
- ga "Noord"
Condities:
- geen uniek punt NW
- muur zuid
- muur noord
- geen muur west
Actie:
- ga "West"
Condities:
- geen uniek punt NW
- muur zuid
- muur noord
- muur west
- geen muur oost
Actie:
- ga "Oost"
Condities:
- uniek punt NW
- muur zuid
- geen muur oost
Actie:
- ga "Oost"
Toestand:
x=2
s=1
k=2
i=0
p=0
a=1
Condities:
- geen muur west
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur west
- geen muur zuid
- geen muur noord
Actie:
- ga "Noord"
Toestand:
x=2
s=2
k=2
i=0
p=0
a=0
Condities:
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Condities:
- muur noord
- muur oost
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur noord
- muur oost
- muur zuid
- geen muur west
Actie:
- ga "West"
Toestand:
x=3
s=0
k=0
i=0
p=0
a=0
Condities:
- geen muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur west
- geen muur noord
Actie:
- ga "Zuid"
Condities:
- muur noord
- geen muur oost
Actie:
- ga "Oost"
Toestand:
x=3
s=1
k=0
i=0
p=0
a=0
Condities:
- geen muur zuid
Actie:
- ga "Zuid"
Condities:
- muur zuid
- geen muur west
Actie:
- ga "Noord"
Condities:
- muur zuid
- muur west
Actie:
- ga "Noord"
Toestand:
x=3
s=0
k=0
i=1
p=0
a=0
Condities:
- geen muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur west
- geen muur noord
Actie:
- ga "Noord"
Condities:
- muur noord
Actie:
- ga "Oost"

7 Het vinden van een uniek punt

Een automaat kan dus een doolhof volledig doorzoeken, gegeven dat de unieke punten gemarkeerd zijn. Echter willen we ook een doolhof kunnen doorzoeken waarbij deze punten niet gemarkeerd zijn, want deze punten zijn ook niet gemarkeerd in een doolhof zoals wij deze gedefinieerd hebben in Hoofdstuk 2. In dit hoofdstuk willen we deze punten ook niet gaan markeren, omdat we dan (net zoals in Trémaux's algoritme) een goedgevulde markeerstift nodig zullen hebben. Hiermee zouden we in het geval van het markeren van de unieke punten wel iets minder markeringen nodig hebben dan in Trémaux's algoritme, maar we zouden nog steeds net zoveel punten moeten markeren als dat er binnenmuren in het doolhof zijn. En naarmate het doolhof meer binnenmuren bevat, hebben we dan dus een grotere hoeveelheid markeringen nodig.

We willen dus niet elk uniek punt gaan markeren in een doolhof. Dus dan luidt de vraag: *hoe willen we de unieke punten wel gaan vinden?* We willen het algoritme dat we kennen vanuit vorig hoofdstuk gaan gebruiken om het hele doolhof te doorzoeken, maar in plaats van dat we een uniek punt gelijk kunnen herkennen in het doolhof, willen we zodra we aankomen in een vakje dat mogelijk een $uniek_{NW}$ of $uniek_{NO}$ is, gaan controleren of dit vakje ook daadwerkelijk $uniek_{NW}$ of $uniek_{NO}$ is. Dit doen we door vanuit dat vakje wat rond te lopen door het doolhof, om vervolgens naar ditzelfde vakje terug te keren met de kennis of dit vakje wel of niet $uniek_{NW}$ of $uniek_{NO}$ is. In dit hoofdstuk beschrijven we het algoritme voor hoe we kunnen controleren of een vakje $uniek_{NW}$ of $uniek_{NO}$ is. Hiervoor hebben we een automaat nodig waaraan één lege teller (zie Definitie 7.1), of twee kiezelstenen zijn toegevoegd. We definiëren eerst wat een teller is, en vervolgens hoe een automaat met kiezelstenen eruit ziet. De definitie van een automaat met kiezelstenen komt uit [9].

Definitie 7.1 (teller). In een *teller* t kan een bepaalde waarde uit $\mathbb{Z}$ worden opgeslagen. Deze waarde kan worden aangepast, door het volledig te overschrijven door een nieuwe waarde uit $\mathbb{Z}$ of door een waarde uit $\mathbb{Z}$ bij de waarde die op dat moment in t staat op te tellen.

Definitie 7.2 (automaat met kiezelstenen). Een automaat met K kiezelstenen M^K is een eindigetoestandsautomaat met K kiezelstenen m_1, m_2, ..., m_K. De automaat heeft de volgende eigenschappen over een doolhof:

- M^K start altijd op een wit vakje van het doolhof.
- M^K kan verplaatsen naar een buurvakje, mits dat wit is.
- M^K kan controleren of er een kiezelsteentje (en welk kiezelsteentje m_i, $i \in \{1, ..., k\}$) ligt op het witte vakje waar hij staat. Als er een kiezelsteentje ligt, kan de automaat met een actie uit het output-alfabet het kiezelsteentje weghalen. Ook kan de automaat de kiezelsteen laten liggen.
- M^K weet welke kiezelsteentjes m_i, $i \in \{1, ..., k\}$ hij nog bij zich heeft, en dus nog niet op een vakje geplaatst heeft.
- M^K kan een kiezelsteen m_i, $i \in \{1, ..., k\}$ plaatsen op het witte vakje waar hij staat, mits m_i nog niet op een ander wit vakje in het doolhof ligt.

Let op: M^K verplaatst een kiezelsteen m_i van wit vakje A naar wit vakje B, door zichzelf eerst te verplaatsen naar wit vakje A, vervolgens hier m_i op te pakken, vervolgens naar wit vakje B te verplaatsen en als laatste hier m_i weer neer te leggen.

Stelling 7.3 (controleren of een punt in een doolhof een uniek punt is). *Laat D een doolhof zijn. Er bestaat een automaat met een teller (een M^2) dat vanuit elk wit vakje X van D samen met een lege teller (met 2 kiezelstenen), na wat rondlopen door D, kan terugkeren naar X met een lege teller (2 kiezelstenen), en stoppen in toestanden q_{NO}^{JA} of q_{NW}^{JA} als het punt ten noordoosten of respectievelijk ten noordwesten uniek is, of in de toestanden q_{NO}^{NEE} of q_{NW}^{NEE} als het punt ten noordoosten of respectievelijk noordwesten niet uniek is.*

Opmerking: de automaat (met twee kiezelsteentjes of met één teller) wil vanuit een vakje X de volgende twee dingen controleren:

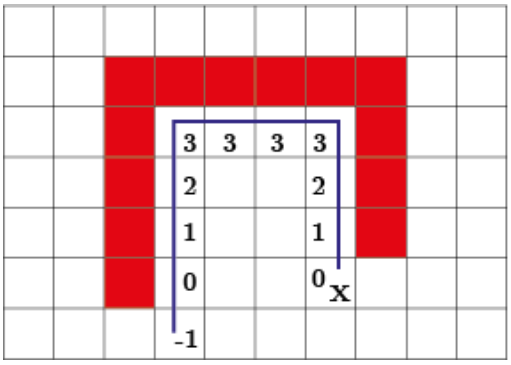
1. of X wel of niet $uniek_{NO}$ is.
2. of X wel of niet $uniek_{NW}$ is.

Om beide deze punten te controleren, moet de automaat twee keer vanuit X door het doolhof lopen. Bij het controleren van punt 1. zal de automaat, na wat rondlopen in doolhof D, weer terugkomen in vakje X en daar eindigen in een van de volgende twee toestanden: q_{NO}^{JA} of q_{NO}^{NEE}. Als de automaat ook punt 2. wil controleren, zal het opnieuw moeten starten vanuit X, en opnieuw wat door het doolhof moeten rondlopen, om vervolgens weer terug te komen in X en te eindigen in de toestand q_{NW}^{JA} of q_{NW}^{NEE}. Hoe dit precies gaat, wordt uitgelegd in de bewijsschets van Stelling 7.3.

Let op: het is niet mogelijk dat de automaat vanuit een vakje X in zowel q_{NO}^{JA} als q_{NO}^{NEE} (of q_{NW}^{JA} en q_{NW}^{NEE}) zou kunnen eindigen. Immers, als het punt ten noordoosten van X een uniek punt is, dan eindigt de automaat in q_{NO}^{JA}, en zou deze dus nooit in q_{NO}^{NEE} (de toestand die aangeeft dat er geen uniek punt ten noordoosten is) kunnen eindigen.

Let ook op: het is ook niet mogelijk dat de automaat vanuit een vakje X in zowel q_{NO}^{JA} als q_{NW}^{JA} zou kunnen eindigen. Namelijk als X $uniek_{NW}$ is, dan is het punt ten noordwesten van X het meest westelijke van het meest zuidelijke punt van de rand van de desbetreffende binnenmuur. En als dit het geval is, dan is het vakje ten noorden van X sowieso een muurvakje, anders is het punt ten noordwesten van X nooit uniek. Dan volgt dat X nooit $uniek_{NO}$ is, want als er ook een muurvakje ten noorden van X zit, dan is er sowieso een westelijker punt aan de rand van de muur te vinden, namelijk het punt ten noordwesten van X.

We geven later een volledige bewijsschets van Stelling 7.3, maar we leggen alvast kort uit wat het idee van de teller is (die dus te vervangen is door twee kiezelstenen, wat we bewijzen in Stelling 7.4). De teller kunnen we gebruiken om bij te houden of we noordelijker, zuidelijker of op dezelfde verticale hoogte zijn als ons beginpunt X uit doolhof D. Dit doen we door elke keer als we een stap naar het noorden zetten $+1$ bij de waarde op de teller op te tellen, en elke keer als we een stap naar het zuiden zetten -1 bij de waarde op de teller op te tellen. Hierdoor weten we dat zodra we vanuit X beginnen te lopen (om bijvoorbeeld een muur te volgen), dat dan geldt dat wanneer de teller weer op 0 staat dat we dan op dezelfde verticale hoogte in het doolhof zijn als dat we begonnen zijn. Dus als we beginnen te lopen vanuit vakje X en ondertussen de teller bijhouden, en we zien dat de teller op 0 staat zodra we een stap hebben gezet naar vakje Y uit D dan weten we dat Y zich op dezelfde verticale hoogte bevindt als X. Op de volgende pagina in Figuur 23 zien we een voorbeeld van hoe een teller zich zou gedragen als we vanuit X naar het noorden lopen en de muur aan de rechterkant volgen. We zien dat we op dezelfde verticale hoogte zijn als beginpunt X zodra de teller weer op 0 staat.



Figuur 23: waarden van de teller als je vanuit X naar het noorden loopt en de muur volgt

We kunnen in plaats van de teller ook twee kiezelstenen gebruiken om bij te houden op welke verticale hoogte we ons bevinden in verhouding tot ons beginpunt. Hoe we dit met twee kiezelstenen doen, komt uit [9] en wordt bewezen in Stelling 7.4.

Stelling 7.4 (automaat met 2 kiezelstenen gebruiken als teller). *Laat D een doolhof zijn en laat x een punt op de rand van een muur W van D zijn. Er is een M^2 (een automaat met twee kiezelstenen) die, beginnend in x, stopt op een punt y op de rand van W zodanig dat x en y zich verticaal op dezelfde hoogte bevinden (en dat x en y zich dus op de rand van dezelfde muur bevinden).*

Bewijs. We geven een algoritme dat een automaat met twee kiezelstenen M^2 omschrijft, dat de rand van de muur $W \subset D$ volgt vanuit een punt x dat op de rand van W ligt. De M^2 verplaatst de kiezelstenen m_1 en m_2 op zo'n manier dat de afstand tussen de twee kiezelstenen langs de rand de afstand tussen de verticale hoogte van punt x en kiezelsteentje m_1 representeert. De M^2 stopt als deze afstand 0 is. Dit is als m_1 en m_2 op hetzelfde punt op de rand liggen, en dit punt is dan het punt y.

We voeren bovenstaand algoritme volgens de volgende stappen uit (zie Figuur 24 voor een afbeelding van het algoritme):

1. start op het punt x. Leg hier m_1 en m_2 neer, en ga vervolgens naar Stap 2.
2. verplaats m_1 een afstand 1 (oftewel precies de lengte van de kant van één vakje) over de rand van de muur, bijvoorbeeld in de richting waarbij we de muur aan onze rechterkant volgen. Ga vervolgens naar Stap 3.
3. nu moeten we m_2 verplaatsen zodanig dat de afstand tussen m_1 en m_2 de verticale afstand tussen x en m_1 is. Dit doen we als volgt: We bewegen langs de rand en vinden m_2. Nu zijn er drie verschillende opties:
   (a) we hebben m_1 zojuist een afstand 1 over de rand naar het *oosten of westen* verplaatst.
   In dit geval verplaatsen we m_2 ook een afstand 1 in voorwaartse richting over de rand (dezelfde voorwaartste richting over de rand als we zojuist m_1 hebben verplaatst, bijvoorbeeld in de richting waarbij we de muur aan onze rechterkant volgen). Hierdoor blijft de totale afstand tussen m_1 en m_2 gelijk. Dit is wat we willen omdat, doordat we m_1 naar het oosten of westen hebben verplaatst, de verticale afstand tussen beginpunt x en m_1 ook gelijk is gebleven.
   Ga naar Stap 4.

(b) we hebben m_1 zojuist een afstand 1 over de rand naar het *noorden* verplaatst. In dit geval verplaatsen we m_2 met een afstand 2 over de rand in dezelfde voorwaartse richting als we zojuist m_1 hebben verplaatst (bijvoorbeeld in de richting waarbij we de muur aan onze rechterkant volgen). Hierdoor verandert de afstand tussen m_1 en m_2 met een afstand 1 over de rand in, zeg, positieve waarde. Dit is wat we willen omdat de verticale afstand tussen x en m_1 ook met een afstand 1 is veranderd.
Ga naar Stap 4.

(c) we hebben m_1 zojuist een afstand 1 over de rand naar het *zuiden* verplaatst. In dit geval verplaatsen we m_2 niet. Hierdoor verandert de afstand tussen m_1 en m_2 met een afstand 1 over de rand in, zeg, negatieve waarde. Dit is wat we willen omdat de verticale afstand tussen x en m_1 ook met een afstand 1 is veranderd in tegenovergestelde richting dan bij (b).
Ga naar Stap 4.

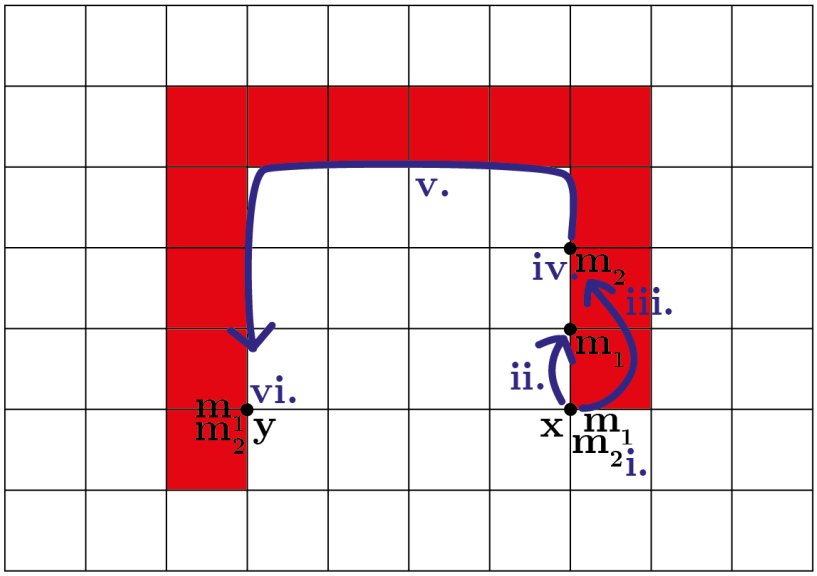
4. er zijn nu twee opties:

(a) m_1 en m_2 liggen op hetzelfde punt op de rand.
In dit geval is het punt waar m_1 en m_2 liggen het punt y dat we zoeken dat op dezelfde verticale hoogte ligt als punt x.
We zijn klaar en de automaat stopt.

(b) m_1 en m_2 liggen niet op hetzelfde punt op de rand.
In dit geval gaan we terug naar Stap 2 en voeren we de stappen van het algoritme nogmaals uit.

□

Op de volgende pagina in Figuur 24 is bovenstaand algoritme afgebeeld.



Figuur 24: het verplaatsen van kiezelsteentjes m_1 en m_2 langs een muur:

i. we beginnen op punt x op de rand van de muur en leggen hier m_1 en m_2 neer (Stap 1. van het algoritme).

ii. we verplaatsen m_1 een afstand 1 over de rand van de muur, in dit geval met de muur aan onze rechterkant (Stap 2. van het algoritme).

iii. we hebben m_1 zojuist een afstand 1 naar het noorden verplaatst. We verplaatsen m_2 dus een afstand 2 langs de rand met de muur aan onze rechterkant (Stap 3.(b) van het algoritme.

iv. m_1 en m_2 liggen niet op hetzelfde punt op de rand. We moeten dus nog doorgaan met het algoritme en gaan terug naar Stap 2 (Stap 4.(b) van het algoritme).

v. we blijven de stappen van het algoritme volgen, en verplaatsen de steentjes langs de muur zoals de stappen aangeven.

vi. m_1 en m_2 liggen op hetzelfde punt op de rand liggen. Het punt waar we nu zijn is punt y met dezelfde verticale hoogte als ons beginpunt x. De automaat stopt nu (Stap 4.(a) van het algoritme).

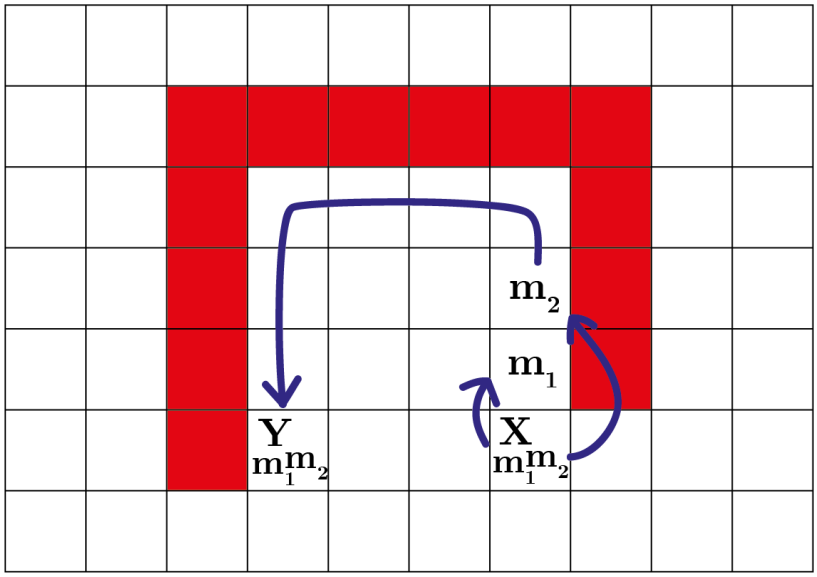
Opmerking: Omdat we punt x niet gemarkeerd hebben, kunnen we niet zomaar teruglopen naar punt x. Om punt x terug te vinden, kunnen we hetzelfde algoritme weer gebruiken vanuit punt y, maar dan beginnen we met verplaatsen in tegengestelde richting van hoe we ons eerst verplaatst hebben (dus als we eerst met de muur aan de rechterkant zijn gaan lopen vanuit x, dan lopen we nu met de muur aan de linkerkant terug vanuit y). Zodra we weer in de situatie zijn dat de twee kiezelstenen op dezelfde plek liggen, zijn we terug bij ons startpunt x.

Opmerking 2: In het algoritme in het bewijs van Stelling 7.4 verplaatsen we de kiezelstenen steeds over de rand van een muur, waardoor de kiezelstenen op de hoekpunten van vakjes komen te liggen. In Figuur 24 zien we een voorbeeld van hoe kiezelstenen m_1 en m_2 over de rand van een muur verplaatst worden. Echter is hier één probleem: we hebben in Subparagraaf 2.2.4 gezegd dat we een kiezelsteentje alleen in een wit vakje kunnen neerleggen, en dus niet op de rand. We moeten dit dus oplossen, wat op het eerste gezicht niet heel lastig lijkt (maar stiekem toch iets lastiger is). Op het eerste

gezicht zouden we denken dat we gewoon het algoritme uit het bewijs van Stelling 7.4 kunnen gebruiken, maar met een paar aanpassingen. Die aanpassingen zijn als volgt:

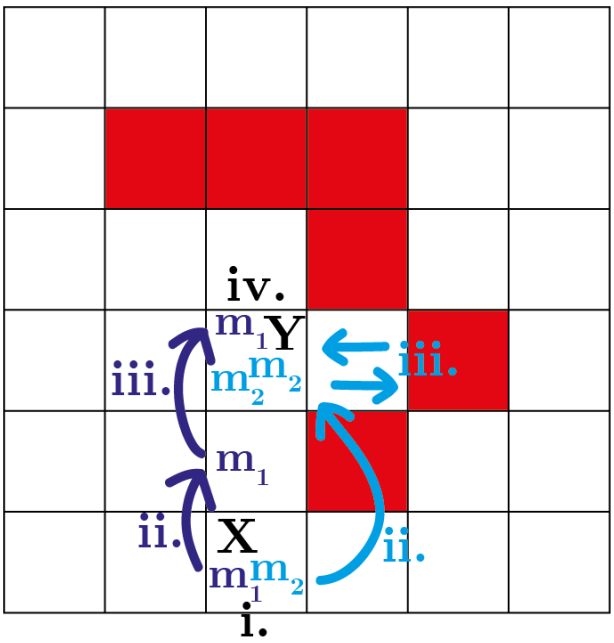
- in plaats van dat we de kiezelsteentjes neerleggen op punt x op de rand van de muur, leggen we de kiezelsteentjes neer in wit vakje X, wat het vakje is waar we zouden staan als we muur W aan het volgen zijn bij punt x.
- in plaats van dat we een kiezelsteentje een afstand 1 verplaatsen over de rand van de muur W, verplaatsen we nu een kiezelsteentje gewoon één vakje langs de rand van de muur W (en respectievelijk verplaatsen we bij een afstand 2 het kiezelsteentje 2 vakjes, en als we een kiezelsteentje zouden laten liggen op de rand van de muur, dan laten we een kiezelsteentje nu in het vakje liggen).

Verder zouden we de kiezelsteentjes dan kunnen verplaatsen zoals in het algoritme van het bewijs van Stelling 7.4 wordt gedaan. In de situatie van Figuur 24 gaat het goed als we de kiezelsteentjes gewoon in de witte vakjes leggen neerleggen, en de genoemde aanpassingen doen. Dit is te zien in Figuur 25.



Figuur 25: het verplaatsen van kiezelsteentjes m_1 en m_2 op dezelfde manier als in Figuur 24, maar in plaats van het neerleggen van de steentjes op de rand van de muur, leggen we de steentjes in de witte vakjes zelf, beginnend in wit vakje X en stopt in wit vakje Y op dezelfde verticale hoogte als X.

Echter werkt bovenstaande methode niet altijd. Soms kan het namelijk zo zijn dat er één wit vakje is die aan twee verschillende delen van de muur grenst. In deze situatie is het van belang dat we weten welke deel van de muur we aan het volgen zijn. Een voorbeeld van een situatie waarin het niet werkt, zie je op de volgende pagina in Figuur 26.



Figuur 26: het verplaatsen van kiezelsteentjes m_1 en m_2 over de witte vakjes waarbij we de muur volgen. In het plaatje worden de verplaatsingen van kiezelsteentje m_1 in het donkerblauw aangegeven, de verplaatsingen van kiezelsteentje m_2 in het lichtblauw aangegeven en een argument dat voor beide kiezelsteentjes geldt in het zwart aangegeven:

i. we beginnen in vakje X en leggen hier m_1 en m_2 neer.

ii. we verplaatsen m_1 één vakje langs de muur met de muur aan onze rechterkant. We hebben m_1 hiermee naar het noorden verplaatst. Hierdoor moeten we m_2 (volgens het algoritme uit het bewijs van Stelling 7.4 twee vakjes langs de muur verplaatsen. Dit doen we dus ook.

iii. m_1 en m_2 liggen niet op hetzelfde punt op de rand. We moeten dus nog doorgaan met het algoritme en gaan terug naar Stap 2 van het algoritme uit het bewijs van Stelling 7.4. We verplaatsen $m-1$ weer één vakje langs de muur, waarmee we m_1 naar het noorden verplaatsen. Nu moeten we m_2 dus twee vakjes langs de muur verplaatsen. Dit doen we dus ook.

iv. m_1 en m_2 liggen nu in hetzelfde vakje, dus de automaat stopt. Echter is het vakje Y waar we staan niet op dezelfde verticale hoogte als dat ons beginvakje X was. Het gaat hier mis doordat we met kiezelsteen m_1 de muur ten zuidoosten van vakje Y aan het volgen waren, terwijl we met kiezelsteen m_2 de muur ten noordoosten van het vakje Y aan het volgen waren.

Het is tijdens het plaatsen van de kiezelstenen dus van belang om te onthouden welk deel van de muur we aan het volgen zijn. We zouden dit op de volgende twee manieren kunnen bijhouden:

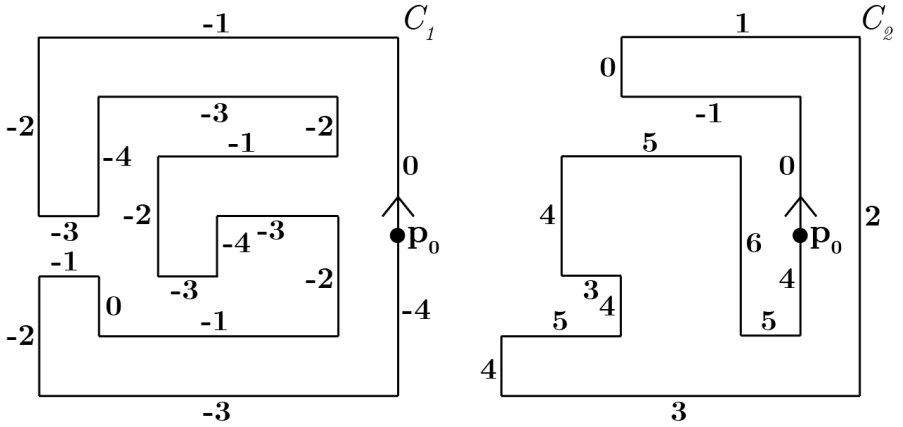
- we kunnen een kiezelsteentje in plaats van midden in een wit vakje ook neerleggen in ditzelfde witte vakje tegen de rand van de muur die we op dat moment aan het volgen zijn (net zoals dat we in Trémaux's algoritme in Paragraaf 3.2 markeringen aanbrengen in een vakje tegen de rand aan). Een nadeel hiervan is dat we kiezelsteentjes op een bepaalde positie in een vakje moeten kunnen neerleggen.

- we kunnen een kiezelsteentje wel gewoon in het midden (of waar we willen) van het witte vakje neerleggen, en ondertussen in de interne toestanden van de automaat bijhouden in welke richting (noord, oost, zuid of west) we de kiezelsteentjes voor het laatst verplaatst hebben. Dit is de manier waarop we het hebben aangepakt in het uiteindelijke algoritme voor het controleren van het unieke punt, waarvan de JavaScript-code in Appendix A.3 staat. We hebben dit gedaan door hierin de twee variabelen g en h toe te voegen, waarmee we bijhouden in welke richting we kiezelsteentje m_1 (dit houden we bij in variabele g) en kiezelsteentje m_2 (dit houden we bij in variabele h) hebben verplaatst de laatste keer dat we m_1 of m_2 verplaatst hebben. Hierdoor kunnen we bijhouden over welke rand we m_1 en m_2 zouden hebben moeten verplaatst. We zijn voor deze manier gegaan, omdat we dan de kiezelsteentjes gewoon in het midden van een vakje kunnen neerleggen, en dat er dus voor zorgt dat we zo min mogelijk eisen nodig hebben om een doolhof volledig te kunnen doorzoeken.

We kunnen dus door middel van twee kiezelsteentjes (of één teller) bijhouden wanneer we weer op dezelfde verticale hoogte zijn als vakje X in een doolhof D. Hieruit kunnen we uiteindelijk gaan bepalen of vakje X $uniek_{NO}$ of $uniek_{NW}$ is, of allebei niet. Om dit uiteindelijk te kunnen doen, geven we eerst nog wat definities en stellingen.

Definitie 7.5 (indicator v van enkelvoudig gesloten kromme C). Laat C een enkelvoudig gesloten kromme in het 2-dimensionale Euclidische vlak zijn, volledig samengesteld uit een eindig aantal horizontale en verticale lijnsegmenten. Laat de lengte van $C > 0$ zijn. Laat p_0 een punt zijn op het inwendige van een verticaal lijnsegment van de kromme C. Laat p een willekeurig punt zijn op de kromme C. We kunnen nu een wandeling maken van p_0 naar p door vanuit p_0 naar het noorden te lopen, en vervolgens de kromme C te volgen tot we bij het punt p aankomen. Voor p definiëren we *indicator* v door:

$v(p) =$ [totaal aantal bochten naar rechts dat gemaakt moet worden tijdens een wandeling van p_0 naar p, beginnend in p_0] $-$ [totaal aantal bochten naar links dat gemaakt moet worden tijdens een wandeling van p_0 naar p, beginnend in p_0].

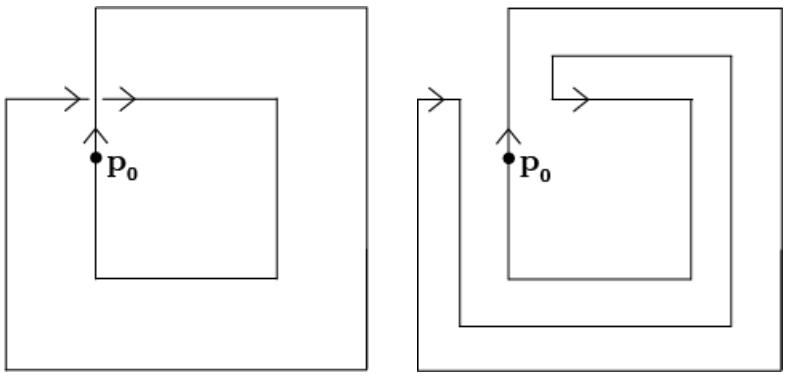


Figuur 27: twee wandelingen langs twee krommen C_1 (in (a)) en C_2 (in (b)) vanuit beginpunt p_0. Langs elk lijnsegment van de kromme wordt de waarde $v(p)$ gegeven, voor wat $v(p)$ zou zijn voor een p die op dit lijnsegment zou liggen. In (a) geldt $v(p_0) = -4$ (een wandeling van p_0 naar p_0) en in (b) geldt $v(p_0) = 4$.

Stelling 7.6. *Laat C een enkelvoudig gesloten kromme in het 2-dimensionale Euclidische vlak zijn, volledig samengesteld uit een eindig aantal horizontale en verticale lijnsegmenten. Laat p_0 een punt zijn op het inwendige van een verticaal lijnsegment van de kromme C. Dan geldt:*

- $v(p_0) = 4$ *als de wandeling van p_0 naar p_0 homotoop is met een lus met de klok mee.*
- $v(p_0) = -4$ *als de wandeling van p_0 naar p_0 homotoop is met een lus tegen de klok in.*

Bewijsschets. We geven een intuïtieve bewijsschets van bovenstaande stelling, waarbij we kromme C zien als een wandeling door een doolhof D, waarbij p_0 een vakje in D is. Stel een wandeling van p_0 naar p_0 is homotoop met een lus met de klok mee. Uit Definitie 2.2 weten we dat een lus van p_0 naar p_0 via vakjes $Y_1, ...Y_n$ loopt, waarbij $n \geq 1$. Als we een wandeling met de klok mee willen maken van p_0 terug naar p_0, waarbij geldt dat we minimaal 1 stap moeten maken naar een vakje dat niet p_0 is (dit volgt uit de definitie van een lus), en waarbij ook geldt dat we p_0 in dezelfde richting verlaten als dat we hierin terugkomen (dit volgt uit de eis uit de stelling waarin gezegd wordt dat het punt p_0 (in deze bewijsschets is p_0 een vakje) altijd een punt is op het inwendige van een verticaal lijnsegment van kromme C). Oftewel, de eerste stap dat we vanuit p_0 zetten is in dezelfde richting (noord, zuid, oost of west) als de laatste stap die we terug naar p_0 zetten. Verder geldt dat kromme C alleen horizontale en verticale lijnsegmenten bevat, en dus alleen bochten van 90° bevat. Uit het feit dat we p_0 in dezelfde richting verlaten als dat we terugkomen, volgt dat we in de lus met de klok mee minimaal 4 bochten van 90° moeten maken, waarbij de uiteindelijke lus die we maken 360°, of een veelvoud hiervan, is. Echter kan het geen veelvoud van 360° zijn. Dit volgt namelijk uit de topologie, omdat we dan in de kromme C sowieso een doorsnijding van de kromme met zichzelf krijgen, wat niet kan in een enkelvoudig gesloten kromme. Een enkelvoudig gesloten kromme verdeelt namelijk het vlak in één binnen- en één buitengebied, waarbij de kromme zelf de grens tussen de twee gebieden is, en waarbij we niet een pad kunnen maken vanuit een punt x in het binnengebied naar een punt y in het buitengebied zonder de kromme te doorsnijden. Dit is bewezen in de Stelling van Jordan [5]. Op de volgende pagina in Figuur 28, zien we een voorbeeld van hoe een wandeling die homotoop is met een lus van 720° zichzelf zou moeten doorsnijden, en hoe een enkelvoudig gesloten kromme (die zichzelf dus niet doorsnijdt) homotoop is met een lus met de klok mee (van 360°).



(a) (b)

Figuur 28: (a): als de wandeling van p_0 naar p_0 homotoop is met een wandeling van 720° om een cirkel met de klok mee, dan moet de lus zichzelf doorsnijden, en is het dus geen enkelvoudig gesloten kromme.
(b): een enkelvoudig gesloten kromme waarin de wandeling van p_0 naar p_0 homotoop is met een lus met de klok mee van 360°.

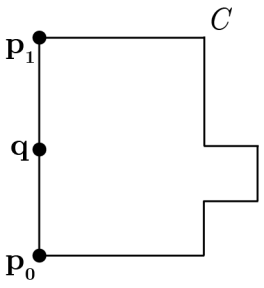
Hieruit volgt dat het aantal bochten dat we moeten maken om een wandeling te maken die homotoop is met een lus met de klok mee, gelijk is aan: $4 + x$ naar rechts en x naar links. Dan volgt dat $v(p)$ = [aantal bochten naar rechts] - [aantal bochten naar links] = $[4 + x] - [x] = 4$.

Andersom, kunnen we op deze manier bewijzen dat als de wandeling van p_0 naar $p = p_0$ homotoop is met een lus tegen de klok in, dat dan $v(p)$ = [aantal bochten naar rechts] - [aantal bochten naar links] = $[x] - [4 + x] = -4$. □

Gevolg 7.6.1. $v(p) \equiv 1 \bmod 3$ *als de wandeling van* p_0 *naar* p *homotoop is met een lus met de klok mee.* $v(p) \equiv 2 \bmod 3$ *als de wandeling van* p_0 *naar* p *homotoop is met een lus tegen de klok in.*

Bewijs. Dit volgt eenvoudig uit Stelling 7.6. Hierin is namelijk bewezen dat $v(p) = 4$ als de wandeling van p_0 naar p homotoop is met een lus met de klok mee. Dus dan $v(p) = 4 \equiv 1 \bmod 3$. Ook is bewezen dat $v(p) = -4$ als de wandeling van p_0 naar p homotoop is met een lus tegen de klok in. Dus dan geldt $v(p) = -4 \equiv 2 \bmod 3$. □

Definitie 7.7. Laat p een hoekpunt zijn op de kromme C, dus een punt waar een verticaal en een horizontaal lijnsegment elkaar raken, en laat q een punt zijn op het verticale lijnsegment van C met als eindpunt p. Dan zeggen we dat $v(p) = v(q)$.



Figuur 29: $v(p_0) = v(q) = v(p_1)$

Nu hebben we genoeg definities en stellingen gehad om een bewijsschets te geven van Stelling 7.3. We geven een bewijsschets met hierin het idee van het algoritme en geven niet een bewijs met het volledige algoritme met stappen en een schema zoals we dit deden in Hoofdstuk 5 en Hoofdstuk 6. We doen dit omdat het algoritme voor het vinden van de unieke punten erg groot is: de volledige JavaScript-code staat in Appendix A.3 en bevat 2914 regels aan code. Dit is veel meer dan de 115 regels aan JavaScript-code voor het doorzoeken van een eenvoudig doolhof (zie Appendix A.1) of de minder dan 349 regels aan JavaScript-code voor het doorzoeken van een doolhof waarin alle unieke punten gemarkeerd zijn (zie Appendix A.2). Hierdoor wordt het volledig uitgetypte algoritme, en het bijbehorende schema te groot om te geven in deze scriptie. Het maximaal aantal toestanden dat de automaat die controleert of iets een uniek punt is, kan bevatten, is het aantal verschillende waardes dat elke variabele in de JavaScript-code mogelijk kan hebben met elkaar vermenigvuldigd. In de JavaScript-code (Appendix A.3) hebben we de volgende variabelen:

- s (4 verschillende mogelijke waardes): dit heeft, net zoals in de eerdere algoritmes, te maken met in welke richting we de laatste stap hebben gezet.
- v (3 verschillende mogelijke waardes): dit houdt de indicator v bij.
- n (2 verschillende mogelijke waardes): dit onderscheidt de situatie waarin we de steentjes klaar leggen van de situatie waarin we de steentjes volgens het algoritme verplaatsen.
- m (3 verschillende mogelijke waardes): dit houdt bij in welke richting we kiezelsteentje m_1 voor het laatst hebben verplaatst (dus hoe veel stappen we kiezelsteentje m_2 moeten verplaatsen).
- r (2 verschillende mogelijke waardes): dit houdt bij of we bezig zijn met kiezelsteentje m_1 te zoeken om vervolgens te verplaatsen, of dat we bezig zijn met kiezelsteentje m_2 te zoeken om eventueel te verplaatsen.
- j (3 verschillende mogelijke waardes): dit is verschillend afhankelijk van hoeveel stappen we moeten zetten voordat we kiezelsteentje m_2 kunnen neerleggen.
- z (2 verschillende mogelijke waardes): dit gebruiken we elke keer nadat we kiezelsteentje m_2 hebben neergelegd om te controleren of we al moeten stoppen met het algoritme. Oftewel: we moeten dan controleren of kiezelsteentje m_1 en kiezelsteentje m_2 in hetzelfde vakje liggen, en ook of de richting waarin we de steentjes voor het laatst verplaatst hebben dezelfde richting is (oftewel of $g = h$, zie voor g en h hieronder).
- b (4 verschillende mogelijke waardes): dit zorgt ervoor dat nadat we hebben gevonden of het vakje X wel of geen uniek punt heeft, we op de juiste manier terug kunnen lopen naar het vakje X. Hoe we teruglopen wordt uitgelegd in de Bewijsschets van Stelling 7.3.
- g (4 verschillende mogelijke waardes): dit houdt bij wat de richting was van de laatste stap voor het neerleggen van kiezelsteentje m_1.
- h (4 verschillende mogelijke waardes): dit houdt bij wat de richting was van de laatste stap voor het neerleggen van kiezelsteentje m_2.
- c (2 verschillende mogelijke waardes): houdt bij wanneer je, nadat je een punt Z_i westelijker dan beginvakje X hebt gevonden, wanneer je weer naar het westen gaat. Hoe dit wordt bijgehouden en waarvoor dit wordt gebruikt, wordt uitgelegd in de Bewijsschets van Stelling 7.3.

Wat alle bovenstaande variabelen precies inhouden, staat uitgebreider uitgelegd in de opmerkingen in de eerste regels van de JavaScript-code in Appendix A.3. Het maximaal aantal toestanden van de automaat is dus de vermenigvuldiging van alle aantallen verschillende mogelijke waardes bij bovenstaande variabelen, oftewel: het maximaal aantal toestanden $= 4 * 3 * 2 * 3 * 2 * 3 * 2 * 4 * 4 * 4 * 2 = 110592$ toestanden. Waarschijnlijk heeft de uiteindelijke automaat minder toestanden, omdat net zoals in Hoofdstuk 5 en Hoofdstuk 6 waarschijnlijk verschillende toestanden hetzelfde gedrag zullen vertonen en dus samengevoegd kunnen worden, maar het zullen er alsnog veel meer zijn dan in de schema's die bij die hoofdstukken staan.

Hieronder volgt de bewijsschets van Stelling 7.3.

Bewijsschets van Stelling 7.3. Laat D een doolhof, laat X een wit vakje in D zijn. We bevinden ons in X.

Te bewijzen: er bestaat een M^2 (een automaat met 2 kiezelstenen) dat vanuit X wat kan rondlopen in D en vervolgens weer terug kan keren naar X wetende of X wel of niet $uniek_{NO}$ of $uniek_{NW}$ is.

Bewijsschets: Dit bewijs vat eigenlijk twee dingen samen, namelijk:

1. We willen controleren of X $uniek_{NO}$ is.
2. We willen controleren of X $uniek_{NW}$ is.

Als 1. geldt, dan willen we allereerst controleren of X aan wat basiseigenschappen voldoet om een $uniek_{NO}$ vakje te kunnen zijn. Dit doen we door de volgende eigenschappen te controleren:

- er bevindt zich een wit vakje ten noorden van X,
- er bevindt zich een wit vakje ten oosten van X,
- als we een stap naar het noorden zetten, dan bevindt zich een muurvakje aan het oosten van ons,
- als we een stap naar het oosten zetten, dan bevindt zich een muurvakje aan het noorden van ons (dit volgt eigenlijk ook uit de vorige drie eisen),
- als we een stap naar het oosten zetten, dan bevindt zich een wit vakje aan het oosten van ons.

Als X niet aan alle bovenstaande voorwaarden voldoet, dan kunnen we gelijk stoppen met het algoritme en teruggeven dat X niet $uniek_{NO}$ is. We hoeven dan niet het algoritme te volgen, en eindigen gelijk in de toestand q_{NO}^{NEE}.

Als 2. geldt, dan willen we dus controleren of X $uniek_{NW}$ is. Hiervoor kunnen we in principe hetzelfde algoritme gebruiken als dat voor het controleren of X $uniek_{NO}$ is, door hier van tevoren nog aan toe te voegen:
Als onderstaande voorwaarden gelden:

- er bevindt zich een wit vakje ten westen van X,
- er bevindt zich een muurvakje ten noorden van X,
- als we een stap naar het westen zetten, dan moeten alle eigenschappen gelden die moeten gelden voor het controleren of X een $uniek_{NO}$ vakje is.

Dan: zet een stap naar het westen, zeg naar vakje Y en volg het algoritme voor het controleren of Y een $uniek_{NO}$ vakje is. Dit algoritme eindigt uiteindelijk weer in vakje Y in de toestand q_{NO}^{JA} of q_{NO}^{NEE}, we gaan dat algoritme straks duidelijker omschrijven. Zet als je in vakje Y bent weer een stap naar het oosten. Nu ben je weer op je beginpunt. Verder geldt:

- als q_{NO}^{JA} in Y dan geldt q_{NW}^{JA} in X,
- als q_{NO}^{NEE} in Y dan geldt q_{NW}^{NEE} in X.

Als X en Y niet aan alle bovenstaande voorwaarden voldoen, dan kunnen we gelijk stoppen met het algoritme en teruggeven dat X niet $uniek_{NW}$ is. We hoeven dan niet door het doolhof te lopen, en eindigen gelijk in de toestand q_{NW}^{NEE}.

We weten nu aan welke voorwaarden vakje X moet voldoen waardoor het eventueel een $uniek_{NO}$ vakje kan zijn.

Te bewijzen: We willen nu dus het algoritme geven voor het controleren of vakje X een $uniek_{NO}$ vakje is.

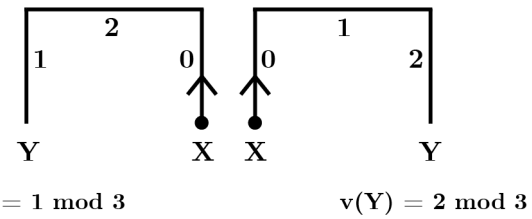
Bewijsschets: Een M^2 kan controleren of vakje X eventueel $uniek_{NO}$ is, door te beginnen vanuit X naar het noorden de muur te volgen en tegelijkertijd de kiezelstenen te verplaatsen zoals in Stelling 7.4, om zo de verticale hoogte bij te houden ten opzichte van ons beginvakje X. We doen dit als volgt:

1. We leggen eerst kiezelsteen m_1 en kiezelsteen m_2 neer in vakje X.
2. We verplaatsen m_1 één vakje langs de rand van de muur, beginnend vanuit X in de richting van het noorden. We volgen nu dus de muur aan onze rechterkant.
3. Vervolgens verplaatsen we m_2 zoals in het bewijs van Stelling 7.4 beschreven langs de muur:
   - als we m_1 zojuist naar het *oosten* of *westen* hebben verplaatst, dan verplaatsen we m_2 ook één vakje langs de muur, ook beginnend vanuit X in de richting van het noorden.
   - als we m_1 zojuist naar het *noorden* hebben verplaatst, dan verplaatsen we m_2 twee vakjes langs de muur, ook beginnend vanuit X in de richting van het noorden.
   - als we m_1 zojuist naar het *zuiden* hebben verplaatst, dan verplaatsen we m_2 niet, en laten we m_2 liggen.
4. We blijven m_1 en m_2 verplaatsen totdat ze weer allebei in hetzelfde vakje liggen (nadat zowel m_1 als m_2 weer verplaatst is). Nu weten we dat we op een vakje Y zijn beland met dezelfde verticale hoogte als X, en onderbreekt de automaat dit proces om te gaan controleren of we op dezelfde verticale hoogte ten westen of ten oosten van X zijn beland. Dit doen we als volgt:

In de interne toestanden van de M^2 houden we tijdens het verplaatsen van m_1 de waarde van indicator v modulo 3 bij (in de JavaScript-code in Appendix A.3 is dit de variabele v). Zodra de automaat het proces van het verplaatsen van kiezelsteentjes m_1 en m_2 heeft onderbroken, kijken we welke waarden indicator v heeft. We weten dat zodra m_1 en m_2 op dezelfde plek liggen (op vakje Y), dat we dan op dezelfde verticale hoogte zijn als ons startvakje X. Uit de waarde van indicator v kunnen we opmaken of we ten westen of ten oosten van X zijn. Namelijk:

- als $v(Y) \equiv 1 \bmod 3$ dan zijn we ten westen van X,
- als $v(Y) \equiv 2 \bmod 3$ dan zijn we ten oosten van X.

Dit volgt uit Gevolg 7.6.1.



Figuur 30: als $v(Y) \equiv 1 \bmod 3$ dan is Y ten westen van X, als $v(Y) \equiv 2 \bmod 3$ dan is Y ten oosten van X

Nu zijn er dus twee verschillende gevallen:

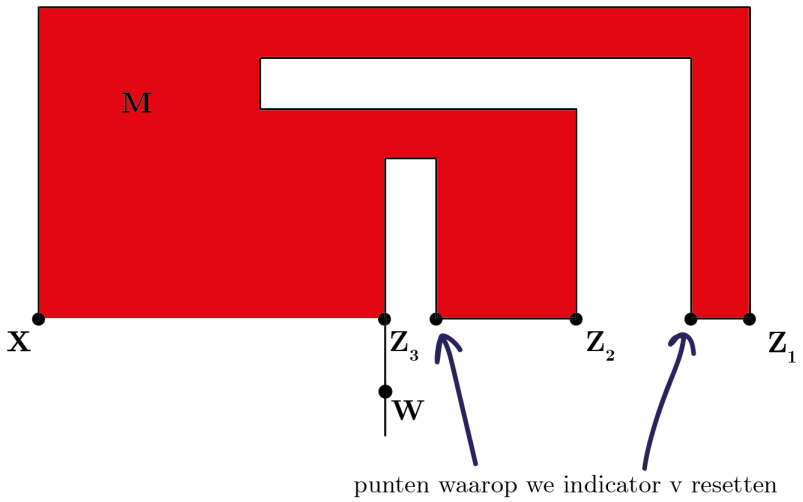
1. Y ligt ten westen van X. In dit geval weten we dat X geen uniek punt is. Het unieke punt is namelijk het meest westelijke van het meest zuidelijke punt van een binnenmuur, en we hebben zojuist een westelijker punt van de muur gevonden. In dit geval zoeken we het pad terug naar X en stoppen de automaat hier in de toestand q_{NO}^{NEE}: er zit geen uniek punt in het noordoosten.

   Het pad terugvinden naar X doen we door vanuit Y weer naar het noorden te lopen (we lopen nu in de richting waar we vandaan kwamen) en het algoritme waarin we de twee kiezelstenen verplaatsen weer te gebruiken. Zodra m_1 en m_2 weer op dezelfde plek liggen, zijn we weer op het eerste punt dat dezelfde verticale hoogte heeft als Y dat we tegen zijn gekomen door terug te lopen langs de muur, dit is ons beginvakje X.

2. Y ligt ten oosten van X. In dit geval kunnen we nog niet zeggen of X het unieke punt wel of niet is. De automaat moet dus nog verder langs de muur lopen, totdat een van de volgende dingen gebeurt:

   (a) We vinden een eindige rij met opeenvolgende vakjes met dezelfde verticale hoogte als vakje X, namelijk de vakjes $Z_1, ..., Z_n$, met de eigenschap dat $Z_1, ..., Z_n$ allemaal ten oosten liggen van X, en de eigenschap dat Z_i westelijker ligt dan Z_{i-1} voor alle $i \in \{1, ..., n\}$. We houden bij of een punt Z_i met $i \in \{1, ..., n\}$ oostelijker of westelijker ligt dan zijn vorige punt, door elke keer als we op dezelfde verticale hoogte zijn de indicator v weer te resetten, en opnieuw beginnen met het bijhouden van het modulo rekenen van indicator v in de interne toestanden als we weer naar het gaan noorden lopen (zo weten we dus elke keer of we oostelijker of westelijker zijn dan het vorige punt). Nu kan er een van de volgende twee situaties plaatsvinden:

      i. We vinden een vakje W dat zuidelijker ligt dan X, doordat we vanuit Z_n (na eventueel een aantal stappen naar het westen te hebben gezet) een stap naar het zuiden moeten zetten om de muur te volgen (en dan dus een stap naar vakje W moeten). In dit geval gaan we naar Stap (b).



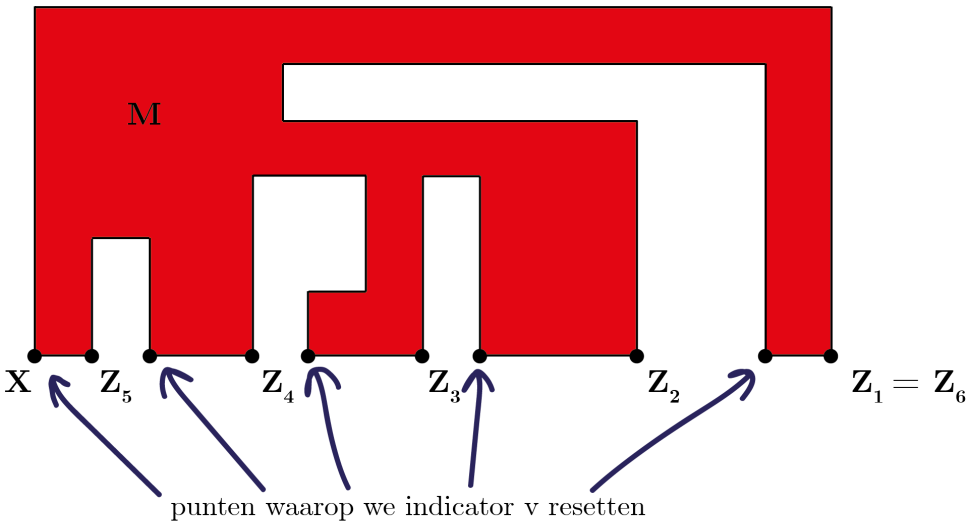
Figuur 31: uitleg van Situatie i: als we vanuit vakje X naar het noorden lopen, om zo muur M te volgen aan de rechterkant, en ondertussen met kiezelsteentjes m_1 en m_2 bijhouden wanneer we weer op dezelfde verticale hoogte als X zijn. Als we dan elke keer stoppen op de vakjes rondom M waarin we op dezelfde verticale hoogte staan als vakje X, namelijk in vakjes Z_i met $i\ in\{1,2,3\}$, waarbij vakje Z_i steeds westelijker ligt dan vakje Z_{i-1} (wat we kunnen weten door de waarde van indicator v in de interne toestanden bij te houden, die we steeds weer resetten zodra we opnieuw naar het noorden gaan lopen nadat we een vakje Z_i gevonden hebben). Dan gaan we door totdat we een vakje W vinden dat zuidelijker is dan een vakje Z_i, dus zuidelijker dan het vakje X waar we begonnen zijn. Dan weten we dat vakje X niet $uniek_{NO}$.

ii. We vinden een vakje Z_{n+1} met dezelfde verticale hoogte als X dat voor het eerst oostelijker ligt dan vakje Z_n. In dit geval weten we dat X is vanuit Z_{n+1} het vorige punt waar we indicator v hebben gereset, en dan ook X is $uniek_{NO}$: er zit een uniek punt in het noordoosten van X. Dit weten we doordat:

   A. we nooit zuidelijker gelopen hebben. X is dus een van de zuidelijkste punten.

   B. zodra we één keer een punt ten oosten van X hebben gevonden, kunnen we niet meer westelijker komen, zonder eerst zuidelijker te zijn geweest. Dit komt doordat het pad dat we lopen (de muur volgen) een enkelvoudig gesloten kromme is waarbij we de kromme zelf niet doorsnijden. Zodra we dus weer een punt ten oosten van ons vorige punt vinden, weten we dat we opnieuw langs ons unieke punt X zijn gelopen.

   We volgen ons pad terug naar X en stoppen daar het algoritme in de toestand q_{NO}^{JA}: er zit een uniek punt in het noordoosten van X. Het pad terug vinden we door zodra we in een vakje Z_{n+1} komen dat oostelijker is dan ons vorige vakje, om te draaien, en met behulp van kiezelsteentjes m_1 en m_2 terug te lopen tot deze beiden weer op dezelfde plek liggen. Dit is dan dus het vorige vakje waar we voor het laatste de indicator v gereset hadden, wat vakje X is.



Figuur 32: uitleg van Situatie ii: als we vanuit vakje X naar het noorden lopen, om zo muur M te volgen aan de rechterkant, en ondertussen met kiezelsteentjes m_1 en m_2 bijhouden wanneer we weer op dezelfde verticale hoogte als X zijn. Als we dan elke keer stoppen in de vakjes rondom M waarin we op dezelfde verticale hoogte staan als vakje X, namelijk in vakjes Z_i met $i \in \{1, ..., 5\}$, waarbij vakje Z_i steeds westelijker ligt dan vakje Z_{i-1} (wat we kunnen weten door de waarde van indicator v in de interne toestanden bij te houden, die we steeds weer resetten zodra we opnieuw naar het noorden gaan lopen nadat we een vakje Z_i gevonden hebben). Dan gaan we door totdat indicator v aantoont dat we een keer oostelijker zijn gewandeld (Z_6 in het figuur) dan ons vorige punt met dezelfde verticale hoogte (Z_5 in het figuur), dan weten we dat X is het vorige punt waarop we indicator v gereset hebben, en X is $uniek_{NO}$. Ondertussen zijn we nooit in een vakje W gekomen dat zuidelijker ligt dan X, dan zouden we immers in Situatie i. zitten.

(b) We hebben een vakje W is gevonden dat zuidelijker ligt dan X. We weten dat we een zuidelijker punt hebben gevonden als we vanuit een punt met dezelfde verticale hoogte als X naar het zuiden een stap naar het zuiden moeten lopen om de muur te volgen. In dit geval weten we dat X niet het unieke punt is. Het unieke punt is namelijk het meest westelijke van het meest zuidelijke punt van een binnenmuur, en we hebben zojuist een zuidelijker punt van de muur gevonden. In dit geval zoeken we het pad terug naar X en stoppen de automaat hier in de toestand q_{NO}^{NEE}: er zit geen uniek punt in het noordoosten van vakje X.

We vinden het pad terug naar X door om te draaien en de muur de andere kant op te volgen. We weten dat we nu een rij van n vakjes gaan vinden met dezelfde verticale hoogte als X, namelijk Z_i met $i \in \{1, ..., n\}$. In de interne toestanden van de automaat houden we indicator v bij, en deze kunnen we gebruiken om X terug te vinden. We draaien om en beginnen te lopen in tegengestelde richting langs de muur dan dat we deden. Elke keer als we een vakje met dezelfde verticale hoogte als X vinden, kijken we of dit vakje oostelijker of westelijker ligt dan het vorige vakje waar we geweest zijn met dezelfde verticale hoogte als X. Telkens resetten we tussendoor indicator v weer. Zodra we het eerste vakje vinden dat westelijker ligt dan het vorige

vakje stoppen we. Dit is vakje X. Dit weten we doordat we vanuit X de eerste keer naar het westen zijn gelopen en daarna lag elk vakje alleen maar oostelijker. Hierdoor lopen we terug eerst steeds naar westelijkere vakjes, en is het eerste vakje dat weer oostelijker ligt vakje X.

□

8 Samenvoegen van de twee automaten om zo elk doolhof te kunnen doorzoeken

We weten nu dus dat er een automaat bestaat die vanuit een vakje X uit doolhof D wat kan rondlopen (met 2 kiezelsteentjes) en vervolgens weer kan terugkeren naar X met de kennis of X wel of niet een vakje is dat een uniek punt in het noordoosten of noordwesten heeft. Dat hebben we zojuist bewezen. We weten ook uit Hoofdstuk 6 dat we een doolhof waarin alle unieke punten gemarkeerd zijn volledig kunnen doorzoeken, oftewel rondlopen zodat we sowieso op elk wit vakje van het doolhof D geweest zijn. Hierdoor weten we dat we dus in elk doolhof D waarin de unieke punten gemarkeerd zijn sowieso de uitgang kunnen vinden, ongeacht welk wit vakje de uitgang is.

Echter willen we natuurlijk de uitgang kunnen vinden voor elk doolhof zoals het gedefinieerd is in Hoofdstuk 2, dus zonder dat de punten gemarkeerd zijn. Dit kunnen we bereiken door de automaten waarin we controleren of een vakje X een uniek punt is, en het doorzoeken van het totale doolhof samen te voegen. Dat doen we op de volgende manier:

We volgen de automaat waarin we het doolhof doorzoeken als de punten gemarkeerd zijn (het schema uit Hoofdstuk 6), maar elke keer als we op een vakje X staan dat eventueel $uniek_{NO}$ of $uniek_{NW}$ is, gaan we over op het algoritme om te controleren of vakje X (het algoritme uit dit hoofdstuk) inderdaad $uniek_{NO}$ of $uniek_{NW}$ is. Een vakje X dat eventueel $uniek_{NO}$ is, voldoet aan de volgende eigenschappen (ook genoemd aan het begin van de Bewijsschets van Stelling 7.3):

- er bevindt zich een wit vakje ten noorden van X,
- er bevindt zich een wit vakje ten oosten van X,
- als we een stap naar het noorden zetten, dan bevindt zich een muurvakje aan het oosten van ons,
- als we een stap naar het oosten zetten, dan bevindt zich een muurvakje aan het noorden van ons (dit volgt eigenlijk ook uit de vorige drie eisen),
- als we een stap naar het oosten zetten, dan bevindt zich een wit vakje aan het oosten van ons.

En een vakje X dat eventueel $uniek_{NO}$ is, voldoet aan de volgende eigenschappen (ook genoemd aan het begin van de Bewijsschets van Stelling 7.3):

- er bevindt zich een wit vakje ten westen van X,
- er bevindt zich een muurvakje ten noorden van X,
- als we een stap naar het westen zetten, dan moeten alle eigenschappen gelden die moeten gelden voor het controleren of X een $uniek_{NO}$ vakje is.

We kunnen de twee automaten uit dit hoofdstuk en uit Hoofdstuk 6 dus samenvoegen tot één grote automaat, met twee kiezelsteentjes, die elk doolhof gedefinieerd zoals in Hoofdstuk 2 volledig kan doorzoeken, en dus de uitgang kan vinden.

9 Dankbetuiging

Graag wil ik mijn dank uitspreken aan dr. Wieb Bosma voor de ondersteuning tijdens mijn scriptie. Zijn geduld, enthousiasme, begrip en waardevolle inhoudelijke input waren erg belangrijk voor mij, en hebben mij vaak weer nieuwe motivatie gegeven om verder te gaan. Ik heb zijn begeleiding dan ook erg gewaardeerd.

Daarnaast wil ik ook Timo Maarse bedanken voor zijn belangrijke bijdrage aan mijn scriptie. Zijn prachtige website `https://robot.maarse.xyz/` bood mij een platform waarop ik eenvoudig JavaScript-code kon schrijven en uittesten. Dit heeft mijn onderzoek aanzienlijk vooruit geholpen en heeft mij vooral ook veel meer plezier gegeven in het proces.

Referenties

[1] https://biologiepagina.nl/Brugklasnieuw/Planten/zoekkaartbomen.jpg.

[2] Blum, Manuel en Dexter Kozen: *On the power of the compass (or, why mazes are easier to search than graphs)*. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pagina's 132–142, 1978.

[3] Lucas, É.: *Le jeu des labyrinthes*, pagina's 41–55. Nummer v.1 in *Récréations mathématiques*. Gauthier-Villars, 1882. https://www.cantab.net/users/michael.behrend/repubs/maze_maths/pages/lucas_ch3_en.html.

[4] Maarse, Timo. https://robot.maarse.xyz/.

[5] Maehara, Ryuji: *The Jordan Curve Theorem Via the Brouwer Fixed Point Theorem*. The American Mathematical Monthly, 91, 1984, ISSN 00029890.

[6] Rouse Ball, W.W.: *Mathematical Recreations and Essays*, hoofdstuk Mazes, pagina's 149–154. 2008. https://www.gutenberg.org/ebooks/26839.

[7] Rozenberg, G., H.J. Hoogeboom, en J. Engelfriet: *Hoofdstuk 4: Talen en Automaten*. 2000. https://liacs.leidenuniv.nl/~graafjmde/FI1/dict-ch4.pdf.

[8] Schrijver, Alexander: *Kleuren en Routeren*.

[9] Shah, Anupam N.: *Pebble automata on arrays*. Computer Graphics and Image Processing, 3, 1974, ISSN 0146664X.

[10] Silva, Alexandra en Jurriaan Rot: *Languages and Automata 2018, Radboud University, Lecture 2*. 2018.

[11] Snapp, Robert R. en Maureen D. Neumann: *An Amazing Algorithm*, volume 20, pagina's 540–547. National Council of Teachers of Mathematics, 2015. https://www.jstor.org/stable/10.5951/mathteacmiddscho.20.9.0540.

[12] Steffen, Bernhard, Falk Howar, en Maik Merten: *Introduction to Active Automata Learning from a Practical Perspective*, pagina's 256–296. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, ISBN 978-3-642-21455-4. https://doi.org/10.1007/978-3-642-21455-4_8.

A JavaScript-code

A.1 Code voor het doorzoeken van een eenvoudig doolhof

```
1
2  let s=0; // s=0 betekent laatste stap was noord; s=1 zuid; s=2 oost; s=3
       west
3  let k=0; // k=0 betekent dat je vanaf een zuidermuur naar het noorden aan
       het lopen bent en terug; k=2 betekent dat je de muur aan het volgen
       bent
4
5  return ({north, south, west, east, hasStone1, hasStone2, seeStone1,
       seeStone2, uniqueNW, uniqueNE, uniqueSW, uniqueSE}) => {
6      // Available actions: north, south, west, east, place1, place2,
           pickup1, pickup2
7      if (s===0) {
8          if (k===0) {
9              if (!north) {
10                 return "north";
11             }
12             if (north && !south) {
13                 s = 1;
14                 return "south";
15             }
16             if (north && south && !west) {
17                 s = 3;
18                 return "west";
19             }
20             if (north && south && west && !east) {
21                 s = 2;
22                 k = 2;
23                 return "east";
24             }
25         }
26         if (k===2) {
27             if (!north && west) {
28                 p = 1;
29                 return "north";
30             }
31             if (!west) {
32                 s = 3;
33                 return "west";
34             }
35             if (north && !east && west) {
36                 s = 2;
37                 return "east";
38             }
39             if (north && east && west && !south) {
40                 s = 1;
41                 return "south";
42             }
43         }
44     }
45     if (s===1) {
46         if (k===2) {
47             if (!south && east) {
48                 return "south";
49             }
50             if (!east) {
51                 s = 2;
52                 return "east";
53             }
54             if (east && south && !north) {
55                 s = 0;
56                 k = 0;
```

```
                return "north";
            }
        }
        if (k===0) {
            if (!south) {
                return "south";
            }
            if (south && !west) {
                k = 0;
                s = 3;
                return "west";
            }
            if (south && west && !north) {
                k = 2;
                s = 0;
                return "north";
            }
        }
    }
    if (s===2) {
        if (!east && north) {
            k = 2;
            return "east";
        }
        if (!north) {
            s = 0;
            return "north";
        }
        if (north && east && !south) {
            s = 1;
            return "south";
        }
        if (north && east && south && !west) {
            s = 3;
            return "west";
        }
    }
    if (s===3) {
        if (south && !north) {
            k = 0;
            s = 0;
            return "north";
        }
        if (south && north && !west) {
            k = 2;
            return "west";
        }
        if (south && north && west) {
            k = 2;
            s = 2;
            return "east";
        }
        if (!south) {
            s = 1;
            k = 2;
            return "south";
        }
    }
}
```

A.2 Code voor doorzoeken van een doolhof als de unieke punten gemarkeerd zijn

```
let s=0; // s=0 betekent laatste stap was noord; s=1 zuid; s=2 oost; s=3 west
let k=0; // k=0 betekent dat je vanaf een zuidermuur naar het noorden aan het lopen bent en terug; k=2 betekent dat je de muur aan het volgen bent
let x=0; // x=0 is de normale situatie; x=1 zodra je een eventuele "extra muur" ontdekt in het uniqueNE en je komt vanuit het zuiden, en je was de muur aan het volgen, namelijk dan: volg die extra muur. x=3: volgen extra muur vanaf NW kant
let i=0; // i wordt alleen gebruikt in geval dat je uniqueNW hebt gezien en terug bent gelopen naar het zuiden. i=0 wordt gebruikt als je voor het eerst geen muur aan de west-kant hebt terwijl je weer naar het noorden loopt; i=1 staat aan zolang daar nog wel een muur zit (die grenst aan de zuidmuur vanaf waar je loopt)
let p=0; // p wordt alleen gebruikt in geval dat je naar het noorden loopt en een muur langs het westen volgt, zodra dit het geval is wordt p=0 (normale situatie) veranderd in p=1 (toestand voor als je langs een westmuur loopt naar het noorden). Deze toestand wordt gebruikt om te controleren waar je het eerst weer moet afslaan.
let a=0; // a wordt in de zelfde situatie als p gebruikt. a=0 zolang je een muur aan het westen hebt. Als je geen muur aan het westen hebt dan a=1, dit wordt gebruikt om te onderzoeken wanneer je weer moet afslaan (dit omdat er een uniqueNW en dus een "extra muur" kan zijn en dat moet je dus controleren)

return ({north, south, west, east, hasStone1, hasStone2, seeStone1, seeStone2, uniqueNW, uniqueNE, uniqueSW, uniqueSE}) => {
    // Available actions: north, south, west, east, place1, place2, pickup1, pickup2
    if (p===0) {
        if (x===0) {
            if (!uniqueNW && !uniqueNE) {
                if (s===0) {
                    if (k===0) {
                        if (!north) {
                            return "north";
                        }
                        if (north && !south) {
                            s = 1;
                            return "south";
                        }
                        if (north && south && !west) {
                            s = 3;
                            return "west";
                        }
                        if (north && south && west && !east) {
                            s = 2;
                            k = 2;
                            return "east";
                        }
                    }
                    if (k===2) {
                        if (!north && west) {
                            p = 1;
                            return "north";
                        }
                        if (!west) {
                            s = 3;
                            return "west";
                        }
                        if (north && !east && west) {
                            s = 2;
```

```
                            return "east";
                        }
                        if (north && east && west && !south) {
                            s = 1;
                            return "south";
                        }
                    }
                }
                if (s===1) {
                    if (k===2) {
                        if (!south && east) {
                            return "south";
                        }
                        if (!east) {
                            s = 2;
                            return "east";
                        }
                        if (east && south && !north) {
                            s = 0;
                            k = 0;
                            return "north";
                        }
                    }
                    if (k===0) {
                        if (!south) {
                            return "south";
                        }
                        if (south && !west) {
                            k = 0;
                            s = 3;
                            return "west";
                        }
                        if (south && west && !north) {
                            k = 2;
                            s = 0;
                            return "north";
                        }
                    }
                }
                if (s===2) {
                    if (!east && north) {
                        k = 2;
                        return "east";
                    }
                    if (!north) {
                        s = 0;
                        return "north";
                    }
                    if (north && east && !south) {
                        s = 1;
                        return "south";
                    }
                    if (north && east && south && !west) {
                        s = 3;
                        return "west";
                    }
                }
                if (s===3) {
                    if (south && !north) {
                        k = 0;
                        s = 0;
                        return "north";
                    }
                    if (south && north && !west) {
                        k = 2;
                        return "west";
```

```
109                     }
110                     if (south && north && west) {
111                         k = 2;
112                         s = 2;
113                         return "east";
114                     }
115                     if (!south) {
116                         s = 1;
117                         k = 2;
118                         return "south";
119                     }
120                 }
121             }
122             if (uniqueNE) { //kan niet s===3 want als je vanuit westen zou
                      komen dan zou je door een   extra   muur  heen gelopen
                     zijn
123                 if (s===1) {
124                     if (k===2) {
125                         if (!south) {
126                             x = 1;
127                             return "south";
128                         }
129                         if (south) {
130                             s = 0;
131                             k = 0;
132                             return "north";
133                         }
134                     }
135                     if (k===0) {
136                         if (!south) {
137                             return "south";
138                         }
139                         if (south && !west) {
140                             k = 0;
141                             s = 3;
142                             return "west";
143                         }
144                         if (south && west && !north) { //let op er is
                              nooit een muur noord omdat je uniqueNE is
145                             k = 2;
146                             s = 0;
147                             return "north";
148                         }
149                     }
150                 }
151                 if (s===0) {
152                     if (k===0) {
153                         if (!north) { // altijd de situatie dat er geen
                              muur noord is
154                             x = 0;
155                             return "north";
156                         }
157                     }
158                     if (k===2) {
159                         if (!west) {
160                             s = 3;
161                             return "west";
162                         }
163                         if (west && !north){ //nooit muur noord want
                              uniqueNE
164                             p = 1;
165                             return "north";
166                         }
167                     }
168                 }
169                 if (s===2) {
```

```
170                     if (!north) { //nooit noord want uniqueNE
171                         s = 0;
172                         return "north";
173                     }
174                 }
175             }
176             if (uniqueNW) { //geen s=2 want je kan nooit vanuit het oosten
                     komen (dan loop je namelijk door een "extra muur" heen),
                    ook geen s=1 want je kan nooit uit het zuiden komen sinds
                    ten noorden een muur zit
177                 if (s===0) {
178                     if (k===0) { //je komt niet in situatie uniqueNW, s
                        ===0 en k===2 omdat je dan al in p===1 bent gegaan
                        .
179                         if (north && !south) {
180                             s = 1;
181                             x = 3;      //naar x = 3 om langs de "extra
                                muur" te lopen
182                             return "south";
183                         }
184                         if (north && south && !east) {
185                             s = 2;
186                             k = 2;
187                             return "east";
188                         }
189                     }
190                 }
191                 if (s===3) { //er is altijd een muur noord want uniqueNW,
                    je wilt nooit west want daar extra muur.
192                     if (!south) {
193                         s = 1;
194                         k = 2;
195                         return "south";
196                     }
197                     if (south && !east) {
198                         k = 2;
199                         s = 2;
200                         return "east";
201                     }
202                 }
203             }
204         }
205         if (x===1) {
206             if (k===2) { // je komt alleen in situatie x===1 als k===2
207                 if (!south && !east) {
208                     s = 1;
209                     return "south";
210                 }
211                 if (!south && east) {
212                     x = 0;
213                     s = 1;
214                     k = 2;
215                     return "south";
216                 }
217                 if (south && !north) { // nooit een muur noord want dan
                        niet uniqueNE
218                     s = 0;
219                     x = 0;
220                     k = 0;
221                     return "north";
222                 }
223             }
224         }
225         if (x===3) {
226             if (k===0) { // altijd k===0 want enige situatie dat je in x
                    ===3 komt
```

```
227                 if (s===1) {
228                     if (!south) {
229                         return "south";
230                     }
231                     if (south && !north && !west) { //er is nooit een muur
                            in het noorden want daar kom je net vandaan als s
                        =1
232                         s = 0;
233                         i = 0;
234                         return "north";
235                     }
236                     if (south && !north && west) {
237                         s = 0;
238                         i = 1;
239                         return "north";
240                     }
241                 }
242                 if (s===0) {
243                     if (i===0) {
244                         if (!west && !north) {
245                             return "north";
246                         }
247                         if (west && !north) {
248                             x = 0;
249                             k = 0;
250                             s = 1;
251                             return "south";
252                         }
253                         if (north && !east) {
254                             x = 0;
255                             k = 2;
256                             s = 2;
257                             return "east";
258                         }
259                     }
260                     if (i===1) {
261                         if (west && !north) {
262                             return "north";
263                         }
264                         if (!west && !north) {
265                             i = 0;
266                             return "north";
267                         }
268                         if (north) {
269                             s = 2;
270                             x = 0;
271                             k = 2;
272                             return "east";
273                         }
274                     }
275                 }
276             }
277         }
278     }
279     if (p===1) {
280         if (a===0) {
281             if (west && !north) {
282                 return "north";
283             }
284                if (!west && !north) {
285                 a = 1;
286                 return "north";
287             }
288             if (north && !uniqueNW && !west) {
289                 s = 3;
290                 p = 0;
```

```
                x = 0;
                return "west";
            }
            if (north && uniqueNW && !east) {
                s = 2;
                p = 0;
                x = 0;
                return "east";
            }
            if (north && !uniqueNW && west && !east) {
                s = 2;
                p = 0;
                x = 0;
                return "east";
            }
            if (north && !uniqueNW && west && east) {
                s = 1;
                p = 0;
                x = 0;
                return "south";
            }
        }
        if (a===1) {
            if (s===0) {
                if (!west && !north) {
                    return "north";
                }
                if (west && !north) {
                    s = 1;
                    return "south";
                }
                if (north && !uniqueNW) {
                    s = 1;
                    return "south";
                }
                if (north && uniqueNW) {
                    s = 2;
                    a = 0;
                    p = 0;
                    x = 0;
                    i = 0;
                    k = 2;
                    return "east";
                }
            }
            if (s===1) {
                if (!west && !south) {
                    return "south";
                }
                if (west && !south && !north) {
                    s = 0;
                    a = 0;
                    p = 0;
                    return "north"
                }
            }
        }
    }
}
```

A.3 Code voor het vinden van een uniek punt

```
let s=0; // s=0 betekent laatste stap was noord; s=1 zuid; s=2 oost; s=3 west
let v=0; // v is modulo rekenen
let n=0; // n=0 situatie begin (klaarleggen steentjes), n=1 echte situatie
let m=0; // richting m1 heen verplaatst, m=0 is noord, m=1 is zuid, m=2 is oost of west
let r=0; // r=0 bezig met m1 zoeken en verplaatsen, r=1 bezig met m2 zoeken en verplaatsen.
let j=0; // j=0 doe 1 stap dan j=1 dan steen neerleggen, j=2 als twee stappen voor neerleggen.
let z=0; // elke keer z=1 nadat je steentje 2 hebt verplaatst.
let b=0; // b=0 beginsituatie, b=1 situatie terugbrengen v=1, b=2 situatie v=2 klaarleggen voor volgende rondloop ronde, b=3 terugbrengen c=1 en je komt zuidelijker
let g=0; // laatste stap voor neerleggen steen 1: g=0 noord, g=1 zuid, g=2 oost, g=3 west
let h=0; // laatste stap voor neerleggen steen 2: h=0 noord, h=1 zuid, h=2 oost, h=3 west
let c=0; // c=0 normaal; c=1 je bent al een keer naar het westen gelopen en kijkt nu wanneer je als eerste weer naar het westen gaat.


return ({north, south, west, east, hasStone1, hasStone2, seeStone1, seeStone2, uniqueNW, uniqueNE, uniqueSW, uniqueSE}) => {
    // Available actions: north, south, west, east, place1, place2, pickup1, pickup2
    if (b===0) {
        if (n===0) {
            if (!north && !east && hasStone1 && hasStone2) {
                return "north"; //je bezit altijd steen 1 en 2 maar je kan ook altijd noord omdat dat een voorwaarde is van wanneer je uberhaupt aan deze code begint (als je niet noord kan dan weet je al dat het geen uniek punt NE is dus dan hoef je geen eens te beginnen)
            }
            if (east && hasStone1) { //je ziet als je naar het noorden gelopen bent sowieso een muur in het oosten, je hebt ook sowieso steen 1, je kan deze dus plaatsen.
                return "place1";
            }
            if (east && !hasStone1 && seeStone1) {
                if (!north) {
                   return "north";
               }
                if (north && !west) {
                   s = 3;
                   return "west";
               }
                if (north && west && !south) { //algoritme eindigt na een stap naar het zuiden met de kennis dat er geen uniek punt is.
                   return "east"; //return east gezegd omdat hij dan eindigt in een error, omdat we dat het algoritme hier eindigt (maar eigenlijk nog return south en dan stop)
               }
            }
            if (!seeStone1 && hasStone2 && !hasStone1) {
                n = 1;
                if (s===0) { //s kan alleen 0 of 3 zijn omdat het nog niet 1 of 2 gemaakt kan zijn geworden hierboven.
                    h = 0;
                }
```

```
                if (s===3) {
                    h = 3;
                }
                return "place2";
            }
        }
        if (n===1) {
            if (z===1) {
                if (!(seeStone1 && seeStone2)) {
                    z = 0;
                }
                if (!(g===h)) {
                    z = 0;
                }
                if (seeStone1 && seeStone2 && g===h) { //mag pas nadat
                    stone2 ook verplaatst is!! Daarom z toegevoegd.
                    if (c===0) {
                        if (v===0) {
                            // hier doen we volgens mij precies niks mee,
                                want dit komt niet voor
                        }
                        if (v===1) {
                            // we zijn ten westen van eerst, dus we weten
                                niet uniek punt
                            // we willen steentjes weer mee terug nemen
                                naar vorige punt
                            b = 1;
                            n = 0;
                            return "pickup1";
                        }
                        if (v===2) {
                            // we zijn ten oosten van eerst, dus we weten
                                niet zoveel
                            if (west) { // er is een zuidelijker muur, dus
                                 geen uniek punt, keer terug
                                b = 1;
                                n = 0;
                                j = 0;
                                return "pickup1";
                            }
                            if (!west) {
                                b = 2;
                                n = 0;
                                j = 0;
                                return "pickup1";
                            }
                        }
                    }
                    if (c===1) {
                        // we zijn ten westen van net, maar c===1 dus we
                            zijn al een keer oostelijker geweest
                        if (v===1) {
                            if (west) { // er is een zuidelijker muur, dus
                                 geen uniek punt, keer terug: nog stukkie
                                ingewikkeld, want moeten omgekeerd gaan
                                lopen en modulo bijhouden om te kijken
                                wanneer we voor het eerst een keer naar
                                het westen gaan.
                                b = 3;
                                n = 0;
                                j = 0;
                                return "pickup1";
                            }
                            if (!west) {
                                b = 2;
                                n = 0;
```

```
95                                  j = 0;
96                                  return "pickup1";
97                              }
98                          }
99                          if (v===2) {
100                             // we zijn weer naar het westen gegaan, vorige
                                    punt is ons startpunt en het unieke punt,
                                    we gaan terug.
101                             b = 1;
102                             n = 0;
103                             j = 0;
104                             return "pickup1";
105                         }
106                     }
107                 }
108             }
109             if (z===0) {
110                 if (r===0) {
111                     if (!hasStone1 && !seeStone1){
112                         if (s===0) {
113                            if (!east) {
114                                s = 2;
115                                return "east";
116                            }
117                             if (east && !north) {
118                                return "north";
119                            }
120                             if (east && north && !west) {
121                                s = 3;
122                                return "west";
123                            }
124                             if (east && north && west && !south) {
125                                s = 1;
126                                return "south";
127                            }
128                         }
129                         if (s===1) {
130                              if (!west) {
131                                s = 3;
132                                return "west";
133                            }
134                            if (west && !south) {
135                                return "south";
136                            }
137                             if (west && south && !east) {
138                                s = 2;
139                                return "east";
140                            }
141                             if (west && south && east && !north) {
142                                s = 0;
143                                return "north";
144                            }
145                         }
146                         if (s===2) {
147                             if (!south) {
148                                s = 1;
149                                return "south";
150                            }
151                            if (south && !east) {
152                                return "east";
153                            }
154                             if (south && east && !north) {
155                                s = 0;
156                                return "north";
157                            }
158                             if (south && east && north && !west) {
```

```
                                s = 3;
                                return "west";
                            }
                        }
                        if (s===3) {
                            if (!north) {
                                s = 0;
                                return "north";
                            }
                            if (north && !west) {
                                return "west";
                            }
                            if (north && west && !south) {
                                s = 1;
                                return "south";
                            }
                            if (north && west && south && !east) {
                                s = 2;
                                return "east";
                            }
                        }
                    }
                    if (!hasStone1 && seeStone1){
                        if (s===0 && g===0) {
                            return "pickup1";
                        }
                        else if (s===1 && g===1) {
                            return "pickup1";
                        }
                        else if (s===2 && g===2) {
                            return "pickup1";
                        }
                        else if (s===3 && g===3) {
                            return "pickup1";
                        }
                        else {
                            if (s===0) {
                                if (!east) {
                                    s = 2;
                                    return "east";
                                }
                                if (east && !north) {
                                    return "north";
                                }
                                if (east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                                if (east && north && west && !south) {
                                    s = 1;
                                    return "south";
                                }
                            }
                            if (s===1) {
                                if (!west) {
                                    s = 3;
                                    return "west";
                                }
                                if (west && !south) {
                                    return "south";
                                }
                                if (west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                                if (west && south && east && !north) {
```

```
                                    s = 0;
                                    return "north";
                                }
                            }
                            if (s===2) {
                                if (!south) {
                                    s = 1;
                                    return "south";
                                }
                                if (south && !east) {
                                    return "east";
                                }
                                if (south && east && !north) {
                                    s = 0;
                                    return "north";
                                }
                                if (south && east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                if (!north) {
                                    s = 0;
                                    return "north";
                                }
                                if (north && !west) {
                                    return "west";
                                }
                                if (north && west && !south) {
                                    s = 1;
                                    return "south";
                                }
                                if (north && west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (hasStone1) {
                        if (j===0) {
                            j = 1;
                            if (v===0) {
                                if (s===0) {
                                    if (!east) {
                                        s = 2;
                                        m = 2;
                                        v = 1;
                                        return "east";
                                    }
                                    if (east && !north) {
                                        m = 0;
                                        return "north";
                                    }
                                    if (east && north && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 2;
                                        return "west";
                                    }
                                    if (east && north && west && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 1;
                                        return "south";
```

```
                                    }
                                }
                                if (s===1) {
                                     if (!west) {
                                        s = 3;
                                        m = 2;
                                        v = 1;
                                        return "west";
                                    }
                                    if (west && !south) {
                                        m = 1;
                                        return "south";
                                    }
                                     if (west && south && !east) {
                                        s = 2;
                                        m = 2;
                                        v = 2;
                                        return "east";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        m = 0;
                                        v = 1;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!south) {
                                        s = 1;
                                        m = 1;
                                        v = 1;
                                        return "south";
                                    }
                                    if (south && !east) {
                                        m = 2;
                                        return "east";
                                    }
                                     if (south && east && !north) {
                                        s = 0;
                                        m = 0;
                                        v = 2;
                                        return "north";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 1;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                     if (!north) {
                                        s = 0;
                                        m = 0;
                                        v = 1;
                                        return "north";
                                    }
                                    if (north && !west) {
                                        m = 2;
                                        return "west";
                                    }
                                     if (north && west && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 2;
                                        return "south";
```

```
                                    }
                                     if (north && west && south && !east) {
                                        s = 2;
                                        m = 2;
                                        v = 1;
                                        return "east";
                                    }
                                }
                            }
                            if (v===1) {
                                if (s===0) {
                                    if (!east) {
                                        s = 2;
                                        m = 2;
                                        v = 2;
                                        return "east";
                                    }
                                     if (east && !north) {
                                         m = 0;
                                        return "north";
                                    }
                                     if (east && north && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 0;
                                        return "west";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 2;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                      if (!west) {
                                        s = 3;
                                        m = 2;
                                        v = 2;
                                        return "west";
                                    }
                                    if (west && !south) {
                                        m = 1;
                                        return "south";
                                    }
                                     if (west && south && !east) {
                                        s = 2;
                                        m = 2;
                                        v = 0;
                                        return "east";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        m = 0;
                                        v = 2;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!south) {
                                        s = 1;
                                        m = 1;
                                        v = 2;
                                        return "south";
                                    }
                                    if (south && !east) {
```

```
                                    m = 2;
                                    return "east";
                                }
                                 if (south && east && !north) {
                                    s = 0;
                                    m = 0;
                                    v = 0;
                                    return "north";
                                }
                                 if (south && east && north && !west) {
                                    s = 3;
                                    m = 2;
                                    v = 2;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                 if (!north) {
                                    s = 0;
                                    m = 0;
                                    v = 2;
                                    return "north";
                                }
                                if (north && !west) {
                                    m = 2;
                                    return "west";
                                }
                                 if (north && west && !south) {
                                    s = 1;
                                    m = 1;
                                    v = 0;
                                    return "south";
                                }
                                 if (north && west && south && !east) {
                                    s = 2;
                                    m = 2;
                                    v = 2;
                                    return "east";
                                }
                            }
                        }
                        if (v===2) {
                            if (s===0) {
                                if (!east) {
                                    s = 2;
                                    m = 2;
                                    v = 0;
                                    return "east";
                                }
                                 if (east && !north) {
                                     m = 0;
                                    return "north";
                                }
                                 if (east && north && !west) {
                                    s = 3;
                                    m = 2;
                                    v = 1;
                                    return "west";
                                }
                                 if (east && north && west && !south) {
                                    s = 1;
                                    m = 1;
                                    v = 0;
                                    j = 1;
                                    return "south";
                                }
```

```
                                    }
                                    if (s===1) {
                                         if (!west) {
                                            s = 3;
                                            m = 2;
                                            v = 0;
                                            j = 1;
                                            return "west";
                                        }
                                        if (west && !south) {
                                            m = 1;
                                            j = 1;
                                            return "south";
                                        }
                                         if (west && south && !east) {
                                            s = 2;
                                            m = 2;
                                            v = 1;
                                            j = 1;
                                            return "east";
                                        }
                                         if (west && south && east && !north) {
                                            s = 0;
                                            m = 0;
                                            v = 0;
                                            j = 1;
                                            return "north";
                                        }
                                    }
                                    if (s===2) {
                                         if (!south) {
                                            s = 1;
                                            m = 1;
                                            v = 0;
                                            j = 1;
                                            return "south";
                                        }
                                        if (south && !east) {
                                            m = 2;
                                            j = 1;
                                            return "east";
                                        }
                                         if (south && east && !north) {
                                            s = 0;
                                            m = 0;
                                            v = 1;
                                            j = 1;
                                            return "north";
                                        }
                                         if (south && east && north && !west) {
                                            s = 3;
                                            m = 2;
                                            v = 0;
                                            j = 1;
                                            return "west";
                                        }
                                    }
                                    if (s===3) {
                                         if (!north) {
                                            s = 0;
                                            m = 0;
                                            v = 0;
                                            j = 1;
                                            return "north";
                                        }
                                        if (north && !west) {
```

```
                                    m = 2;
                                    j = 1;
                                    return "west";
                                }
                                 if (north && west && !south) {
                                    s = 1;
                                    m = 1;
                                    v = 1;
                                    j = 1;
                                    return "south";
                                }
                                 if (north && west && south && !east) {
                                    s = 2;
                                    m = 2;
                                    v = 0;
                                    j = 1;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (j===1) {
                        j = 0;
                        r = 1;
                        if (s===0) {
                            g = 0;
                        }
                        if (s===1) {
                            g = 1;
                        }
                        if (s===2) {
                            g = 2;
                        }
                        if (s===3) {
                            g = 3;
                        }
                        return "place1";
                    }
                }
            }
            if (r===1) {
                if (!hasStone2 && !seeStone2) {
                    if (s===0) {
                       if (!east) {
                           s = 2;
                           return "east";
                       }
                        if (east && !north) {
                           return "north";
                       }
                        if (east && north && !west) {
                           s = 3;
                           return "west";
                       }
                        if (east && north && west && !south) {
                           s = 1;
                           return "south";
                       }
                    }
                    if (s===1) {
                         if (!west) {
                           s = 3;
                           return "west";
                       }
                       if (west && !south) {
                           return "south";
```

```
                        }
                         if (west && south && !east) {
                            s = 2;
                            return "east";
                        }
                         if (west && south && east && !north) {
                            s = 0;
                            return "north";
                        }
                    }
                    if (s===2) {
                         if (!south) {
                            s = 1;
                            return "south";
                        }
                        if (south && !east) {
                            return "east";
                        }
                         if (south && east && !north) {
                            s = 0;
                            return "north";
                        }
                         if (south && east && north && !west) {
                            s = 3;
                            return "west";
                        }
                    }
                    if (s===3) {
                         if (!north) {
                            s = 0;
                            return "north";
                        }
                        if (north && !west) {
                            return "west";
                        }
                         if (north && west && !south) {
                            s = 1;
                            return "south";
                        }
                         if (north && west && south && !east) {
                            s = 2;
                            return "east";
                        }
                    }
                }
                if (!hasStone2 && seeStone2) {
                    if (s===0 && h===0) {
                        return "pickup2";
                    }
                    else if (s===1 && h===1) {
                        return "pickup2";
                    }
                    else if (s===2 && h===2) {
                        return "pickup2";
                    }
                    else if (s===3 && h===3) {
                        return "pickup2";
                    }
                    else {
                        if (s===0) {
                            if (!east) {
                                s = 2;
                                return "east";
                            }
                             if (east && !north) {
                                return "north";
```

```
                                }
                                 if (east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                                 if (east && north && west && !south) {
                                    s = 1;
                                    return "south";
                                }
                            }
                            if (s===1) {
                                 if (!west) {
                                    s = 3;
                                    return "west";
                                }
                                if (west && !south) {
                                    return "south";
                                }
                                 if (west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                                 if (west && south && east && !north) {
                                    s = 0;
                                    return "north";
                                }
                            }
                            if (s===2) {
                                 if (!south) {
                                    s = 1;
                                    return "south";
                                }
                                if (south && !east) {
                                    return "east";
                                }
                                 if (south && east && !north) {
                                    s = 0;
                                    return "north";
                                }
                                 if (south && east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                 if (!north) {
                                    s = 0;
                                    return "north";
                                }
                                if (north && !west) {
                                    return "west";
                                }
                                 if (north && west && !south) {
                                    s = 1;
                                    return "south";
                                }
                                 if (north && west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (hasStone2) {
                        if (m===0) {
                            if (j===0) {
```

```
                                    j = 1;
                                    if (s===0) {
                                       if (!east) {
                                           s = 2;
                                           return "east";
                                       }
                                        if (east && !north) {
                                           return "north";
                                       }
                                        if (east && north && !west) {
                                           s = 3;
                                           return "west";
                                       }
                                        if (east && north && west && !south) {
                                           s = 1;
                                           return "south";
                                       }
                                    }
                                    if (s===1) {
                                         if (!west) {
                                           s = 3;
                                           return "west";
                                       }
                                       if (west && !south) {
                                           return "south";
                                       }
                                        if (west && south && !east) {
                                           s = 2;
                                           return "east";
                                       }
                                        if (west && south && east && !north) {
                                           s = 0;
                                           return "north";
                                       }
                                    }
                                    if (s===2) {
                                        if (!south) {
                                           s = 1;
                                           return "south";
                                       }
                                       if (south && !east) {
                                           j = 1;
                                           return "east";
                                       }
                                        if (south && east && !north) {
                                           s = 0;
                                           return "north";
                                       }
                                        if (south && east && north && !west) {
                                           s = 3;
                                           return "west";
                                       }
                                    }
                                    if (s===3) {
                                        if (!north) {
                                           s = 0;
                                           return "north";
                                       }
                                       if (north && !west) {
                                           return "west";
                                       }
                                        if (north && west && !south) {
                                           s = 1;
                                           return "south";
                                       }
                                        if (north && west && south && !east) {
```

```
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===1) {
                                j = 2;
                                if (s===0) {
                                    if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                     if (east && !north) {
                                        return "north";
                                    }
                                     if (east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                      if (!west) {
                                        s = 3;
                                        return "west";
                                    }
                                    if (west && !south) {
                                        return "south";
                                    }
                                     if (west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !east) {
                                        return "east";
                                    }
                                     if (south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                     if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !west) {
                                        return "west";
                                    }
```

```
                                    if (north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===2) {
                                j = 0;
                                r = 0;
                                z = 1;
                                if (s===0) {
                                    h = 0;
                                }
                                if (s===1) {
                                    h = 1;
                                }
                                if (s===2) {
                                    h = 2;
                                }
                                if (s===3) {
                                    h = 3;
                                }
                                return "place2";
                            }
                        }
                        if (m===1) {
                            r = 0;
                            z = 1;
                            if (s===0) {
                                h = 0;
                            }
                            if (s===1) {
                                h = 1;
                            }
                            if (s===2) {
                                h = 2;
                            }
                            if (s===3) {
                                h = 3;
                            }
                            return "place2";
                        }
                        if (m===2) {
                            if (j===0) {
                                j = 1;
                                if (s===0) {
                                    if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                    if (east && !north) {
                                        return "north";
                                    }
                                    if (east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                    if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                }
```

```
                                if (s===1) {
                                     if (!west) {
                                       s = 3;
                                       return "west";
                                    }
                                    if (west && !south) {
                                       return "south";
                                    }
                                     if (west && south && !east) {
                                       s = 2;
                                       return "east";
                                    }
                                     if (west && south && east && !north) {
                                       s = 0;
                                       return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!south) {
                                       s = 1;
                                       return "south";
                                    }
                                    if (south && !east) {
                                       return "east";
                                    }
                                     if (south && east && !north) {
                                       s = 0;
                                       return "north";
                                    }
                                     if (south && east && north && !west) {
                                       s = 3;
                                       return "west";
                                    }
                                }
                                if (s===3) {
                                     if (!north) {
                                       s = 0;
                                       return "north";
                                    }
                                    if (north && !west) {
                                       return "west";
                                    }
                                     if (north && west && !south) {
                                       s = 1;
                                       return "south";
                                    }
                                     if (north && west && south && !east) {
                                       s = 2;
                                       return "east";
                                    }
                                }
                            }
                            if (j===1) {
                                j = 0;
                                r = 0;
                                z = 1;
                                if (s===0) {
                                    h = 0;
                                }
                                if (s===1) {
                                    h = 1;
                                }
                                if (s===2) {
                                    h = 2;
                                }
                                if (s===3) {
```

```
                                    h = 3;
                                }
                                return "place2";
                            }
                        }
                    }
                }
            }
        }
    }
    if (b===1) {
        if (n===0) {
            if (j===0) {
                if (hasStone1 && seeStone2) {
                    return "pickup2";
                }
                if (!north && hasStone1 && hasStone2 && !seeStone1 && !
                    seeStone2) {
                    j = 1;
                    s = 0;
                    return "north";
                }
            }
            if (j===1) {
                if (!seeStone1 && !seeStone2 && hasStone1) {
                    if (s===0) {
                        g = 0;
                    }
                    if (s===1) {
                        g = 1;
                    }
                    if (s===2) {
                        g = 2;
                    }
                    if (s===3) {
                        g = 3;
                    }
                    return "place1";
                }
                if (!hasStone1 && seeStone1) {
                   if (!west) {
                       s = 3;
                       return "west";
                   }
                    if (west && !north) {
                        s = 0;
                       return "north";
                   }
                    if (west && north && !east) {
                       s = 2;
                       return "east";
                   }
                    if (east && north && west && !south) {
                       s = 1;
                       return "south";
                   }
                }
                if (!seeStone1 && hasStone2 && !hasStone1) {
                    n = 1;
                    j = 0;
                    if (s===0) {
                        h = 0;
                    }
                    if (s===1) {
                        h = 1;
                    }
```

```
                    if (s===2) {
                        h = 2;
                    }
                    if (s===3) {
                        h = 3;
                    }
                    return "place2";
                }
            }
        }
        if (n===1) {
            if (z===1) {
                if (!(seeStone1 && seeStone2)) {
                    z = 0;
                }
                if (!(g===h)) {
                    z = 0;
                }
                if (seeStone1 && seeStone2 && g===h) {
                    //hier stopt het algoritme: WEL UNIEK?
                    b = 5;
                    return "pickup1";
                }
            }
            if (z===0) {
                if (r===0) {
                    if (!hasStone1 && !seeStone1) {
                        if (s===0) {
                           if (!west) {
                               s = 3;
                               return "west";
                           }
                            if (west && !north) {
                               return "north";
                           }
                            if (west && north && !east) {
                               s = 2;
                               return "east";
                           }
                            if (east && north && west && !south) {
                               s = 1;
                               return "south";
                           }
                        }
                        if (s===1) {
                            if (!east) {
                               s = 2;
                               return "east";
                           }
                           if (east && !south) {
                               return "south";
                           }
                            if (east && south && !west) {
                               s = 3;
                               return "west";
                           }
                            if (west && south && east && !north) {
                               s = 0;
                               return "north";
                           }
                        }
                        if (s===2) {
                            if (!north) {
                               s = 0;
                               return "north";
                           }
```

```
                        if (north && !east) {
                            return "east";
                        }
                         if (north && east && !south) {
                            s = 1;
                            return "south";
                        }
                         if (south && east && north && !west) {
                            s = 3;
                            return "west";
                        }
                    }
                    if (s===3) {
                         if (!south) {
                            s = 1;
                            return "south";
                        }
                        if (south && !west) {
                            return "west";
                        }
                         if (south && west && !north) {
                            s = 0;
                            return "north";
                        }
                         if (north && west && south && !east) {
                            s = 2;
                            return "east";
                        }
                    }
                }
                if (!hasStone1 && seeStone1){
                    if (s===0 && g===0) {
                        return "pickup1";
                    }
                    else if (s===1 && g===1) {
                        return "pickup1";
                    }
                    else if (s===2 && g===2) {
                        return "pickup1";
                    }
                    else if (s===3 && g===3) {
                        return "pickup1";
                    }
                    else {
                        if (s===0) {
                            if (!west) {
                                s = 3;
                                return "west";
                            }
                             if (west && !north) {
                                return "north";
                            }
                             if (west && north && !east) {
                                s = 2;
                                return "east";
                            }
                             if (east && north && west && !south) {
                                s = 1;
                                return "south";
                            }
                        }
                        if (s===1) {
                             if (!east) {
                                s = 2;
                                return "east";
                            }
```

```
                                if (east && !south) {
                                    return "south";
                                }
                                 if (east && south && !west) {
                                    s = 3;
                                    return "west";
                                }
                                 if (west && south && east && !north) {
                                    s = 0;
                                    return "north";
                                }
                            }
                            if (s===2) {
                                 if (!north) {
                                    s = 0;
                                    return "north";
                                }
                                if (north && !east) {
                                    return "east";
                                }
                                 if (north && east && !south) {
                                    s = 1;
                                    return "south";
                                }
                                 if (south && east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                 if (!south) {
                                    s = 1;
                                    return "south";
                                }
                                if (south && !west) {
                                    return "west";
                                }
                                 if (south && west && !north) {
                                    s = 0;
                                    return "north";
                                }
                                 if (north && west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (hasStone1) {
                        if (j===0) {
                            j = 1;
                            if (s===0) {
                                if (!west) {
                                    s = 3;
                                    m = 2;
                                    return "west";
                                }
                                 if (west && !north) {
                                     m = 0;
                                    return "north";
                                }
                                 if (west && north && !east) {
                                    s = 2;
                                    m = 2;
                                    return "east";
                                }
```

```
                if (east && north && west && !south) {
                    s = 1;
                    m = 1;
                    return "south";
                }
            }
            if (s===1) {
                if (!east) {
                    s = 2;
                    m = 2;
                    return "east";
                }
                if (east && !south) {
                    m = 1;
                    return "south";
                }
                if (east && south && !west) {
                    s = 3;
                    m = 2;
                    return "west";
                }
                if (west && south && east && !north) {
                    s = 0;
                    m = 0;
                    return "north";
                }
            }
            if (s===2) {
                if (!north) {
                    s = 0;
                    m = 0;
                    return "north";
                }
                if (north && !east) {
                    m = 2;
                    return "east";
                }
                if (north && east && !south) {
                    s = 1;
                    m = 1;
                    return "south";
                }
                if (south && east && north && !west) {
                    s = 3;
                    m = 2;
                    return "west";
                }
            }
            if (s===3) {
                if (!south) {
                    s = 1;
                    m = 1;
                    return "south";
                }
                if (south && !west) {
                    m = 2;
                    return "west";
                }
                if (south && west && !north) {
                    s = 0;
                    m = 0;
                    return "north";
                }
                if (north && west && south && !east) {
                    s = 2;
                    m = 2;
```

```
                                    return "east";
                                }
                            }
                        }
                        if (j===1) {
                            j = 0;
                            r = 1;
                            if (s===0) {
                                g = 0;
                            }
                            if (s===1) {
                                g = 1;
                            }
                            if (s===2) {
                                g = 2;
                            }
                            if (s===3) {
                                g = 3;
                            }
                            return "place1";
                        }
                    }
                }
                if (r===1) {
                    if (!hasStone2 && !seeStone2) {
                        if (s===0) {
                           if (!west) {
                               s = 3;
                               return "west";
                           }
                            if (west && !north) {
                               return "north";
                           }
                            if (west && north && !east) {
                               s = 2;
                               return "east";
                           }
                            if (east && north && west && !south) {
                               s = 1;
                               return "south";
                           }
                        }
                        if (s===1) {
                            if (!east) {
                               s = 2;
                               return "east";
                           }
                           if (east && !south) {
                               return "south";
                           }
                            if (east && south && !west) {
                               s = 3;
                               return "west";
                           }
                            if (west && south && east && !north) {
                               s = 0;
                               return "north";
                           }
                        }
                        if (s===2) {
                            if (!north) {
                               s = 0;
                               return "north";
                           }
                           if (north && !east) {
                               return "east";
```

```
                        }
                         if (north && east && !south) {
                            s = 1;
                            return "south";
                        }
                         if (south && east && north && !west) {
                            s = 3;
                            return "west";
                        }
                    }
                    if (s===3) {
                         if (!south) {
                            s = 1;
                            return "south";
                        }
                        if (south && !west) {
                            return "west";
                        }
                         if (south && west && !north) {
                            s = 0;
                            return "north";
                        }
                         if (north && west && south && !east) {
                            s = 2;
                            return "east";
                        }
                    }
                }
                if (!hasStone2 && seeStone2) {
                    if (s===0 && h===0) {
                        return "pickup2";
                    }
                    else if (s===1 && h===1) {
                        return "pickup2";
                    }
                    else if (s===2 && h===2) {
                        return "pickup2";
                    }
                    else if (s===3 && h===3) {
                        return "pickup2";
                    }
                    else {
                        if (s===0) {
                            if (!west) {
                                s = 3;
                                return "west";
                            }
                             if (west && !north) {
                                return "north";
                            }
                             if (west && north && !east) {
                                s = 2;
                                return "east";
                            }
                             if (east && north && west && !south) {
                                s = 1;
                                return "south";
                            }
                        }
                        if (s===1) {
                             if (!east) {
                                s = 2;
                                return "east";
                            }
                            if (east && !south) {
                                return "south";
```

```
                                }
                                 if (east && south && !west) {
                                    s = 3;
                                    return "west";
                                }
                                 if (west && south && east && !north) {
                                    s = 0;
                                    return "north";
                                }
                            }
                            if (s===2) {
                                 if (!north) {
                                    s = 0;
                                    return "north";
                                }
                                if (north && !east) {
                                    return "east";
                                }
                                 if (north && east && !south) {
                                    s = 1;
                                    return "south";
                                }
                                 if (south && east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                 if (!south) {
                                    s = 1;
                                    return "south";
                                }
                                if (south && !west) {
                                    return "west";
                                }
                                 if (south && west && !north) {
                                    s = 0;
                                    return "north";
                                }
                                 if (north && west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (hasStone2) {
                        if (m===0) {
                            if (j===0) {
                                j = 1;
                                if (s===0) {
                                    if (!west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && !north) {
                                        return "north";
                                    }
                                     if (west && north && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
```

```
                                }
                                if (s===1) {
                                    if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        return "south";
                                    }
                                    if (east && south && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                    if (west && south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                    if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        return "east";
                                    }
                                    if (north && east && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                    if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !west) {
                                        return "west";
                                    }
                                    if (south && west && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===1) {
                                j = 2;
                                if (s===0) {
                                    if (!west) {
                                        s = 3;
                                        return "west";
                                    }
                                    if (west && !north) {
                                        return "north";
                                    }
                                    if (west && north && !east) {
                                        s = 2;
                                        return "east";
```

```
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                     if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        return "south";
                                    }
                                     if (east && south && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        return "east";
                                    }
                                     if (north && east && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                     if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !west) {
                                        return "west";
                                    }
                                     if (south && west && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                     if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===2) {
                                j = 0;
                                r = 0;
                                z = 1;
                                if (s===0) {
                                    h = 0;
                                }
                                if (s===1) {
```

```
                                    h = 1;
                                }
                                if (s===2) {
                                    h = 2;
                                }
                                if (s===3) {
                                    h = 3;
                                }
                                return "place2";
                            }
                        }
                        if (m===1) {
                            r = 0;
                            z = 1;
                            if (s===0) {
                                h = 0;
                            }
                            if (s===1) {
                                h = 1;
                            }
                            if (s===2) {
                                h = 2;
                            }
                            if (s===3) {
                                h = 3;
                            }
                            return "place2";
                        }
                        if (m===2) {
                            if (j===0) {
                                j = 1;
                                if (s===0) {
                                   if (!west) {
                                       s = 3;
                                       return "west";
                                   }
                                    if (west && !north) {
                                       return "north";
                                   }
                                    if (west && north && !east) {
                                       s = 2;
                                       return "east";
                                   }
                                    if (east && north && west && !south) {
                                       s = 1;
                                       return "south";
                                   }
                                }
                                if (s===1) {
                                    if (!east) {
                                       s = 2;
                                       return "east";
                                   }
                                   if (east && !south) {
                                       return "south";
                                   }
                                    if (east && south && !west) {
                                       s = 3;
                                       return "west";
                                   }
                                    if (west && south && east && !north) {
                                       s = 0;
                                       return "north";
                                   }
                                }
                                if (s===2) {
```

```
                                    if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        return "east";
                                    }
                                    if (north && east && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                    if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !west) {
                                        return "west";
                                    }
                                    if (south && west && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===1) {
                                j = 0;
                                r = 0;
                                z = 1;
                                if (s===0) {
                                    h = 0;
                                }
                                if (s===1) {
                                    h = 1;
                                }
                                if (s===2) {
                                    h = 2;
                                }
                                if (s===3) {
                                    h = 3;
                                }
                                return "place2";
                            }
                        }
                    }
                }
            }
        }
    }
    if (b===2) {
        if (n===0) {
            if (j===0) {
                if (seeStone2 && hasStone1 && !hasStone2) {
                    return "pickup2";
                }
                if (!west && hasStone1 && hasStone2) {
                    j = 1;
```

```
                    s = 3;
                    return "west";
                }
            }
            if (j===1) {
                if (!west && north) {
                    s = 3;
                    return "west";
                }
                if (west && north) { // er is een zuidelijker punt dus we
                    moeten terug
                    j = 2;
                    s = 2;
                    return "east";
                }
                if (!north) {
                    n = 1;
                    j = 0;
                    v = 0;
                    g = 0;
                    s = 0;
                    return "north";
                }
            }
            if (j===2) {
                if (!east && north){
                    return "east";
                }
                if (!north) {
                    j = 0;
                    b = 1;
                    n = 0;
                    return "place2";
                }
            }
        }
        if (n===1) {
            if (east && !seeStone1 && !seeStone2 && hasStone1) {
                return "place1";
            }
            if (east && !hasStone1 && seeStone1) {
               if (!east) {
                   s = 2;
                   return "east";
               }
                if (east && !north) {
                   return "north";
               }
                if (east && north && !west) {
                   s = 3;
                   return "west";
               }
                if (east && north && west && !south) {
                   s = 1;
                   return "south";
               }
            }
            if (!seeStone1 && hasStone2 && !hasStone1) {
                b = 0;
                c = 1;
                if (s===0) {
                    h = 0;
                }
                if (s===1) {
                    h = 1;
                }
```

```
                if (s===2) {
                    h = 2;
                }
                if (s===3) {
                    h = 3;
                }
                return "place2";
            }
        }
    }
    if (b===3) {
        if (n===0) {
            if (j===0) {
                if (hasStone1 && seeStone2) {
                    return "pickup2";
                }
                if (!north && hasStone1 && hasStone2 && !seeStone1 && !
                    seeStone2) {
                    j = 1;
                    s = 0;
                    v = 0;
                    return "north";
                }
            }
            if (j===1) {
                if (!seeStone1 && !seeStone2 && hasStone1) {
                    if (s===0) {
                        g = 0;
                    }
                    if (s===1) {
                        g = 1;
                    }
                    if (s===2) {
                        g = 2;
                    }
                    if (s===3) {
                        g = 3;
                    }
                    return "place1";
                }
                if (!hasStone1 && seeStone1) {
                   if (!west) {
                       s = 3;
                       return "west";
                   }
                    if (west && !north) {
                        s = 0;
                       return "north";
                   }
                    if (west && north && !east) {
                       s = 2;
                       return "east";
                   }
                    if (east && north && west && !south) {
                       s = 1;
                       return "south";
                   }
                }
                if (!seeStone1 && hasStone2 && !hasStone1) {
                    n = 1;
                    j = 0;
                    if (s===0) {
                        h = 0;
                    }
                    if (s===1) {
                        h = 1;
```

```
                    }
                    if (s===2) {
                        h = 2;
                    }
                    if (s===3) {
                        h = 3;
                    }
                    return "place2";
                }
            }
        }
        if (n===1) {
            if (z===1) {
                if (!(seeStone1 && seeStone2)) {
                    z = 0;
                }
                if (!(g===h)) {
                    z = 0;
                }
                if (seeStone1 && seeStone2 && g===h) {
                    if (v===1) {
                        // hier eindigt het algoritme
                        if (seeStone1) {
                            b = 5;
                            return "pickup1";
                        }
                    }
                    if (v===2){
                        //moeten weer gaan klaarleggen voor volgende
                            loopactie
                        b = 4;
                        n = 0;
                        j = 0;
                        return "pickup1";
                    }
                }
            }
            if (z===0) {
                if (r===0) {
                    if (!hasStone1 && !seeStone1) {
                        if (s===0) {
                           if (!west) {
                               s = 3;
                               return "west";
                           }
                            if (west && !north) {
                               return "north";
                           }
                            if (west && north && !east) {
                               s = 2;
                               return "east";
                           }
                            if (east && north && west && !south) {
                               s = 1;
                               return "south";
                           }
                        }
                        if (s===1) {
                            if (!east) {
                               s = 2;
                               return "east";
                           }
                           if (east && !south) {
                               return "south";
                           }
                            if (east && south && !west) {
```

```
                    s = 3;
                    return "west";
                }
                 if (west && south && east && !north) {
                    s = 0;
                    return "north";
                }
            }
            if (s===2) {
                 if (!north) {
                    s = 0;
                    return "north";
                }
                if (north && !east) {
                    return "east";
                }
                 if (north && east && !south) {
                    s = 1;
                    return "south";
                }
                 if (south && east && north && !west) {
                    s = 3;
                    return "west";
                }
            }
            if (s===3) {
                 if (!south) {
                    s = 1;
                    return "south";
                }
                if (south && !west) {
                    return "west";
                }
                 if (south && west && !north) {
                    s = 0;
                    return "north";
                }
                 if (north && west && south && !east) {
                    s = 2;
                    return "east";
                }
            }
        }
        if (!hasStone1 && seeStone1){
            if (s===0 && g===0) {
                return "pickup1";
            }
            else if (s===1 && g===1) {
                return "pickup1";
            }
            else if (s===2 && g===2) {
                return "pickup1";
            }
            else if (s===3 && g===3) {
                return "pickup1";
            }
            else {
                if (s===0) {
                    if (!west) {
                        s = 3;
                        return "west";
                    }
                     if (west && !north) {
                        return "north";
                    }
                     if (west && north && !east) {
```

```
                                    s = 2;
                                    return "east";
                                }
                                 if (east && north && west && !south) {
                                    s = 1;
                                    return "south";
                                }
                            }
                            if (s===1) {
                                 if (!east) {
                                    s = 2;
                                    return "east";
                                }
                                if (east && !south) {
                                    return "south";
                                }
                                 if (east && south && !west) {
                                    s = 3;
                                    return "west";
                                }
                                 if (west && south && east && !north) {
                                    s = 0;
                                    return "north";
                                }
                            }
                            if (s===2) {
                                 if (!north) {
                                    s = 0;
                                    return "north";
                                }
                                if (north && !east) {
                                    return "east";
                                }
                                 if (north && east && !south) {
                                    s = 1;
                                    return "south";
                                }
                                 if (south && east && north && !west) {
                                    s = 3;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                 if (!south) {
                                    s = 1;
                                    return "south";
                                }
                                if (south && !west) {
                                    return "west";
                                }
                                 if (south && west && !north) {
                                    s = 0;
                                    return "north";
                                }
                                 if (north && west && south && !east) {
                                    s = 2;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (hasStone1) {
                         if (j===0) {
                            j = 1;
                            if (v===0) {
                                 if (s===0) {
```

```
                                    if (!west) {
                                        s = 3;
                                        m = 2;
                                        v = 2;
                                        return "west";
                                    }
                                     if (west && !north) {
                                         m = 0;
                                        return "north";
                                    }
                                     if (west && north && !east) {
                                        s = 2;
                                        m = 2;
                                        v = 1;
                                        return "east";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 2;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                     if (!east) {
                                        s = 2;
                                        m = 2;
                                        v = 2;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        m = 1;
                                        return "south";
                                    }
                                     if (east && south && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 1;
                                        return "west";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        m = 0;
                                        v = 2;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!north) {
                                        s = 0;
                                        m = 0;
                                        v = 2;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        m = 2;
                                        return "east";
                                    }
                                     if (north && east && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 1;
                                        return "south";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
```

```
                                    m = 2;
                                    v = 2;
                                    return "west";
                                }
                            }
                            if (s===3) {
                                if (!south) {
                                    s = 1;
                                    m = 1;
                                    v = 2;
                                    return "south";
                                }
                                if (south && !west) {
                                    m = 2;
                                    return "west";
                                }
                                if (south && west && !north) {
                                    s = 0;
                                    m = 0;
                                    v = 1;
                                    return "north";
                                }
                                if (north && west && south && !east) {
                                    s = 2;
                                    m = 2;
                                    v = 2;
                                    return "east";
                                }
                            }
                        }
                        if (v===1) {
                            if (s===0) {
                                if (!west) {
                                    s = 3;
                                    m = 2;
                                    v = 0;
                                    return "west";
                                }
                                if (west && !north) {
                                    m = 0;
                                    return "north";
                                }
                                if (west && north && !east) {
                                    s = 2;
                                    m = 2;
                                    v = 2;
                                    return "east";
                                }
                                if (east && north && west && !south) {
                                    s = 1;
                                    m = 1;
                                    v = 0;
                                    return "south";
                                }
                            }
                            if (s===1) {
                                if (!east) {
                                    s = 2;
                                    m = 2;
                                    v = 0;
                                    return "east";
                                }
                                if (east && !south) {
                                    m = 1;
                                    return "south";
                                }
```

```
                                    if (east && south && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 2;
                                        return "west";
                                    }
                                    if (west && south && east && !north) {
                                        s = 0;
                                        m = 0;
                                         v = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                    if (!north) {
                                        s = 0;
                                        m = 0;
                                        v = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        m = 2;
                                        return "east";
                                    }
                                    if (north && east && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 2;
                                        return "south";
                                    }
                                    if (south && east && north && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 0;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                    if (!south) {
                                        s = 1;
                                        m = 1;
                                        v = 0;
                                        return "south";
                                    }
                                    if (south && !west) {
                                        m = 2;
                                        return "west";
                                    }
                                    if (south && west && !north) {
                                        s = 0;
                                        m = 0;
                                        v = 2;
                                        return "north";
                                    }
                                    if (north && west && south && !east) {
                                        s = 2;
                                        m = 2;
                                        v = 0;
                                        return "east";
                                    }
                                }
                            }
                            if (v===2) {
                                if (s===0) {
                                    if (!west) {
                                        s = 3;
```

```
                                        m = 2;
                                        v = 1;
                                        return "west";
                                    }
                                     if (west && !north) {
                                         m = 0;
                                        return "north";
                                    }
                                     if (west && north && !east) {
                                        s = 2;
                                        m = 2;
                                        v = 0;
                                        return "east";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 1;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                     if (!east) {
                                        s = 2;
                                        m = 2;
                                        v = 1;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        m = 1;
                                        return "south";
                                    }
                                     if (east && south && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 0;
                                        return "west";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        m = 0;
                                        v = 1;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!north) {
                                        s = 0;
                                        m = 0;
                                        v = 1;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        m = 2;
                                        return "east";
                                    }
                                     if (north && east && !south) {
                                        s = 1;
                                        m = 1;
                                        v = 0;
                                        return "south";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        m = 2;
                                        v = 1;
```

```
                                    return "west";
                                }
                            }
                            if (s===3) {
                                if (!south) {
                                    s = 1;
                                    m = 1;
                                    v = 1;
                                    return "south";
                                }
                                if (south && !west) {
                                    m = 2;
                                    return "west";
                                }
                                if (south && west && !north) {
                                    s = 0;
                                    m = 0;
                                    v = 0;
                                    return "north";
                                }
                                if (north && west && south && !east) {
                                    s = 2;
                                    m = 2;
                                    v = 1;
                                    return "east";
                                }
                            }
                        }
                    }
                    if (j===1) {
                        j = 0;
                        r = 1;
                        if (s===0) {
                            g = 0;
                        }
                        if (s===1) {
                            g = 1;
                        }
                        if (s===2) {
                            g = 2;
                        }
                        if (s===3) {
                            g = 3;
                        }
                        return "place1";
                    }
                }
            }
            if (r===1) {
                if (!hasStone2 && !seeStone2) {
                    if (s===0) {
                        if (!west) {
                            s = 3;
                            return "west";
                        }
                        if (west && !north) {
                            return "north";
                        }
                        if (west && north && !east) {
                            s = 2;
                            return "east";
                        }
                        if (east && north && west && !south) {
                            s = 1;
                            return "south";
                        }
```

```
                        }
                        if (s===1) {
                            if (!east) {
                                s = 2;
                                return "east";
                            }
                            if (east && !south) {
                                return "south";
                            }
                            if (east && south && !west) {
                                s = 3;
                                return "west";
                            }
                            if (west && south && east && !north) {
                                s = 0;
                                return "north";
                            }
                        }
                        if (s===2) {
                            if (!north) {
                                s = 0;
                                return "north";
                            }
                            if (north && !east) {
                                return "east";
                            }
                            if (north && east && !south) {
                                s = 1;
                                return "south";
                            }
                            if (south && east && north && !west) {
                                s = 3;
                                return "west";
                            }
                        }
                        if (s===3) {
                            if (!south) {
                                s = 1;
                                return "south";
                            }
                            if (south && !west) {
                                return "west";
                            }
                            if (south && west && !north) {
                                s = 0;
                                return "north";
                            }
                            if (north && west && south && !east) {
                                s = 2;
                                return "east";
                            }
                        }
                    }
                    if (!hasStone2 && seeStone2) {
                        if (s===0 && h===0) {
                            return "pickup2";
                        }
                        else if (s===1 && h===1) {
                            return "pickup2";
                        }
                        else if (s===2 && h===2) {
                            return "pickup2";
                        }
                        else if (s===3 && h===3) {
                            return "pickup2";
                        }
```

```
                        else {
                            if (s===0) {
                               if (!west) {
                                   s = 3;
                                   return "west";
                               }
                                if (west && !north) {
                                   return "north";
                               }
                                if (west && north && !east) {
                                   s = 2;
                                   return "east";
                               }
                                if (east && north && west && !south) {
                                   s = 1;
                                   return "south";
                               }
                            }
                            if (s===1) {
                                if (!east) {
                                   s = 2;
                                   return "east";
                               }
                               if (east && !south) {
                                   return "south";
                               }
                                if (east && south && !west) {
                                   s = 3;
                                   return "west";
                               }
                                if (west && south && east && !north) {
                                   s = 0;
                                   return "north";
                               }
                            }
                            if (s===2) {
                                if (!north) {
                                   s = 0;
                                   return "north";
                               }
                               if (north && !east) {
                                   return "east";
                               }
                                if (north && east && !south) {
                                   s = 1;
                                   return "south";
                               }
                                if (south && east && north && !west) {
                                   s = 3;
                                   return "west";
                               }
                            }
                            if (s===3) {
                                if (!south) {
                                   s = 1;
                                   return "south";
                               }
                               if (south && !west) {
                                   return "west";
                               }
                                if (south && west && !north) {
                                   s = 0;
                                   return "north";
                               }
                                if (north && west && south && !east) {
                                   s = 2;
```

```
                                    return "east";
                                }
                            }
                        }
                    }
                    if (hasStone2) {
                        if (m===0) {
                            if (j===0) {
                                j = 1;
                                if (s===0) {
                                    if (!west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && !north) {
                                        return "north";
                                    }
                                     if (west && north && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                     if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        return "south";
                                    }
                                     if (east && south && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        return "east";
                                    }
                                     if (north && east && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                     if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !west) {
```

```
                                        return "west";
                                    }
                                     if (south && west && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                     if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===1) {
                                j = 2;
                                if (s===0) {
                                    if (!west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && !north) {
                                        return "north";
                                    }
                                     if (west && north && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                     if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        return "south";
                                    }
                                     if (east && south && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        return "east";
                                    }
                                     if (north && east && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
```

```
                                    if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !west) {
                                        return "west";
                                    }
                                    if (south && west && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===2) {
                                j = 0;
                                r = 0;
                                z = 1;
                                if (s===0) {
                                    h = 0;
                                }
                                if (s===1) {
                                    h = 1;
                                }
                                if (s===2) {
                                    h = 2;
                                }
                                if (s===3) {
                                    h = 3;
                                }
                                return "place2";
                            }
                        }
                        if (m===1) {
                            r = 0;
                            z = 1;
                            if (s===0) {
                                h = 0;
                            }
                            if (s===1) {
                                h = 1;
                            }
                            if (s===2) {
                                h = 2;
                            }
                            if (s===3) {
                                h = 3;
                            }
                            return "place2";
                        }
                        if (m===2) {
                            if (j===0) {
                                j = 1;
                                if (s===0) {
                                    if (!west) {
                                        s = 3;
                                        return "west";
                                    }
                                    if (west && !north) {
                                        return "north";
                                    }
                                    if (west && north && !east) {
                                        s = 2;
```

```
                                        return "east";
                                    }
                                     if (east && north && west && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                }
                                if (s===1) {
                                     if (!east) {
                                        s = 2;
                                        return "east";
                                    }
                                    if (east && !south) {
                                        return "south";
                                    }
                                     if (east && south && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                     if (west && south && east && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                }
                                if (s===2) {
                                     if (!north) {
                                        s = 0;
                                        return "north";
                                    }
                                    if (north && !east) {
                                        return "east";
                                    }
                                     if (north && east && !south) {
                                        s = 1;
                                        return "south";
                                    }
                                     if (south && east && north && !west) {
                                        s = 3;
                                        return "west";
                                    }
                                }
                                if (s===3) {
                                     if (!south) {
                                        s = 1;
                                        return "south";
                                    }
                                    if (south && !west) {
                                        return "west";
                                    }
                                     if (south && west && !north) {
                                        s = 0;
                                        return "north";
                                    }
                                     if (north && west && south && !east) {
                                        s = 2;
                                        return "east";
                                    }
                                }
                            }
                            if (j===1) {
                                j = 0;
                                r = 0;
                                z = 1;
                                if (s===0) {
                                    h = 0;
                                }
```

```
                                if (s===1) {
                                    h = 1;
                                }
                                if (s===2) {
                                    h = 2;
                                }
                                if (s===3) {
                                    h = 3;
                                }
                                return "place2";
                            }
                        }
                    }
                }
            }
        }
    }
    if (b===4) {
        if (n===0) {
            if (j===0) {
                if (seeStone2 && hasStone1 && !hasStone2) {
                    return "pickup2";
                }
                if (!east && hasStone1 && hasStone2) {
                    j = 1;
                    s = 2;
                    return "east";
                }
            }
            if (j===1) {
                if (!east && north) {
                    s = 2;
                    return "east";
                }
                if (!north) {
                    n = 1;
                    j = 0;
                    v = 0;
                    g = 0;
                    s = 0;
                    b = 3;
                    return "north";
                }
            }
        }
    }
    if (b===5) {
        if (seeStone2 && hasStone1) {
            return "pickup2";
            // algoritme eindigt in "WEL/GEEN UNIEK"
        }
    }
}
```