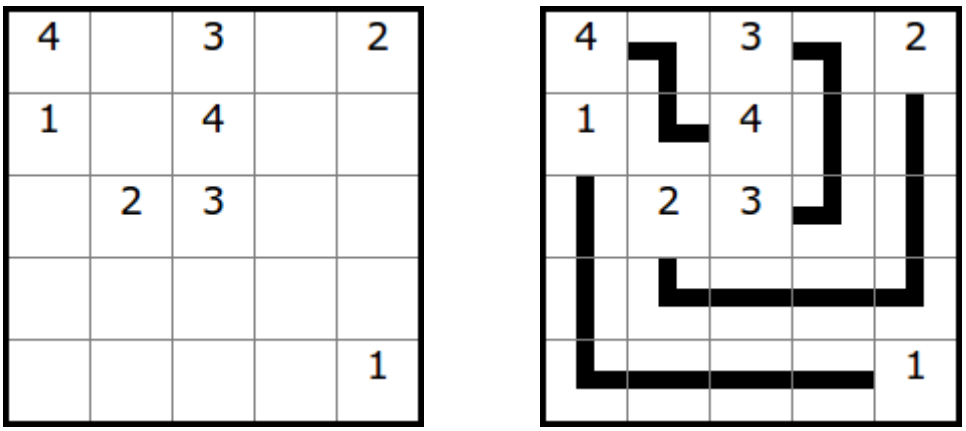
Declaratieve Talen
Prolog-practicum 2012-2013
Arukone

1 Inleiding

Het doel van dit practicum is om een Prolog-programma te implementeren dat automatisch Arukone-puzzels[1] oplost. Een Arukone-puzzel bestaat uit een vierkantig raster waarin paren van cijfers via een aaneengesloten pad met elkaar verbonden moeten worden. Een correcte oplossing bevat tussen iedere twee cijfers die tot hetzelfde paar behoren een doorlopend pad dat geen enkel ander pad kruist en op geen enkel punt splitst in meerdere paden. Figuur 1 toont een voorbeeld van een eenvoudige 5x5-puzzel en de bijhorende oplossing.



(a) Opgave van een 5x5-puzzel (b) Oplossing van een 5x5-puzzel

Figuur 1: De opgave (a) en oplossing (b) van een eenvoudige 5x5-puzzel.

Een Arukone-puzzel is wel ontworpen (*well-designed*) wanneer deze een unieke oplossing heeft en wanneer alle cellen in het raster gevuld zijn. De meegeleverde puzzels voldoen allemaal aan dit principe hoewel er ook Arukone-puzzels bestaan die hier niet aan voldoen. Het volstaat dat je Prolog-programma enkel wel ontworpen Arukone-puzzels op kan lossen.

[1] Raadpleeg `http://en.wikipedia.org/wiki/Numberlink` voor meer informatie.

2 Voorstelling

Deze sectie bevat meer informatie over de opgelegde voorstelling van de opgave en oplossing van een Arukone-puzzel waarmee je Prolog-implementatie overweg moet kunnen.

2.1 Voorstelling van de opgave

We stellen een puzzel voor als een lijst van `link/3`-termen. Het eerste argument van deze termen bevat het kenmerk van het pad (in deze opgave steeds een getal ≥ 0), het tweede argument het beginpunt van het pad en het derde argument het eindpunt van het pad.

```
[
   link(1, pos(2,1), pos(5,5)),
   link(2, pos(1,5), pos(3,2)),
   link(3, pos(1,3), pos(3,3)),
   link(4, pos(1,1), pos(2,3)),
]
```

Bovenstaande lijst stelt de puzzel uit figuur 1 voor. De eerste `link/3`-term in deze lijst stelt dat we een pad zoeken tussen het cijfer één in de cel op de tweede rij in de eerste kolom en het cijfer één in de cel op de vijfde rij in de vijfde kolom. Het eerste argument van een `pos/2`-term bevat dus een rijnummer en het tweede argument een kolomnummer.

Het bestand `puzzels.pl` bevat een aantal voorbeeldpuzzels die we voorstellen als `puzzle/3`-termen. Het eerste argument is een uniek volgnummer, het tweede argument geeft de afmetingen van de puzzel en het derde argument bevat een lijst van `link/3`-termen zoals hierboven beschreven. We stellen de afmeting van een puzzel voor als een `grid/2`-term waarbij het eerste argument het aantal rijen bevat en het tweede argument het aantal kolommen. De term `grid(5,4)` definieert dan een puzzel met vijf rijen en vier kolommen.

2.2 Voorstelling van de oplossing

We stellen een oplossing voor als een lijst van `connects/2`-termen die met paden overeenkomen. Het eerste argument van deze termen bevat het kenmerk van het pad en het tweede argument een lijst van de posities van de cellen waardoor het pad loopt. Het aantal `connects/2`-termen in de oplossing is gelijk aan het aantal `link/3`-termen in de opgave.

```
[
   connects(1, [ pos(2,1), pos(3,1), pos(4,1), pos(5,1),
                 pos(5,2), pos(5,3), pos(5,4), pos(5,5) ]),
```

```
    connects(2, [ pos(1,5), pos(2,5), pos(3,5), pos(4,5),
                  pos(4,4), pos(4,3), pos(4,2), pos(3,2) ]),
    connects(3, [ pos(1,3), pos(1,4), pos(2,4), pos(3,4), pos(3,3) ]),
    connects(4, [ pos(1,1), pos(1,2), pos(2,2), pos(2,3) ])
]
```

Bovenstaande lijst stelt de oplossing uit figuur 1 voor. De laatste `connects/2`-term in deze lijst stelt dat we een pad gevonden hebben tussen het cijfer vier in de cel op positie (1,1) en het cijfer vier in de cel op positie (2,3) via de cellen op posities (1,2) en (2,2). De volgorde van de posities van de cellen die tot een oplossing behoren is echter van geen belang.

3 Opgave

Implementeer een predicaat `arukone/3` dat een oplossing bepaalt voor een gegeven Arukone-puzzel. Het eerste argument van dit predicaat bevat de afmeting van de puzzel, het tweede argument bevat de opgave van de puzzel (zie sectie 2.1) en het derde argument bevat de oplossing van de puzzel (zie sectie 2.2). Arukone-puzzels kunnen zowel vierkantig als rechthoekig zijn, maar het volstaat dat je oplossing enkel voor vierkantige puzzels werkt.

Het bijgeleverde bestand `opgave.pl` bevat het geraamte van je oplossing. Bewaar dit bestand als `oplossing.pl` en werk je oplossing hierin verder uit. Dit bestand voegt de bestanden `puzzels.pl` en `visualisatie.pl` in. Deze bestanden bevatten een aantal voorbeeldpuzzels en een aantal predicaten waarmee je een puzzel of oplossing kan visualiseren.

- Het predicaat `solve(PuzzleId,Solution)` lost de puzzel met nummer `PuzzleId` op, indien het predicaat `arukone/3` correct werkt.
- Het predicaat `show_puzzle(PuzzleId)` toont de puzzel met nummer `PuzzleId`.
- Het predicaat `show_solution(PuzzleId)` lost de puzzel met nummer `PuzzleId` eerst op, indien het predicaat `arukone/3` correct werkt, en toont vervolgens de oplossing.

4 Oplossingsstrategie

De oplossingsstrategie van dit practicum is volledig vrij te kiezen zolang de opgave en de oplossing van de puzzels maar aan de vastgelegde voorstelling voldoet. Een mogelijke oplossingsstrategie bestaat er echter in om de puzzel om te vormen tot een graaf en niet-overlappende paden te zoeken in deze graaf. In deze voorstelling komt iedere cel overeen met een knoop en delen twee knopen (of cellen) een boog wanneer zij in horizontale of verticale richting aan elkaar grenzen. De voorwaarde dat twee paden elkaar niet mogen overlappen kan afgedwongen worden door iedere knoop (of cel) slechts één keer te gebruiken.

5 Tips

Deze sectie bevat een aantal tips die tot een efficiënte oplossing kunnen leiden. We raden aan om verschillende optimalisaties uit te proberen. Bepaalde optimalisaties zullen immers goede resultaten opleveren voor bepaalde puzzels, maar zwak presteren voor andere puzzels.

Probeer Flow Free uit

Spelontwikkelaar Big Duck Games heeft onlangs Flow Free uitgebracht, een populair spel dat erg gelijkaardig is aan Arukone. Het doel van het spel is om paden te bouwen tussen twee stippen van dezelfde kleur in plaats van tussen twee dezelfde cijfers. De oplossingsstrategie van beide spellen is bijgevolg hetzelfde. We raden aan om Flow Free eens uit te proberen zodat je vertrouwd geraakt met het probleem en zelf een aantal interessante tactieken kan ontdekken. Flow Free is gratis beschikbaar voor iOS en Android, maar er bestaan ook afgeleide HTML5-implementaties die je in een moderne browser kan spelen.

- **Android:** `https://play.google.com/store/apps/details?id=com.bigduckgames.flow`
- **iOS:** `https://itunes.apple.com/be/app/id526641427`
- **HTML5:** `http://www.html5games.com/2012/07/flow-free/`

Bouw de paden naargelang de afstand tussen begin- en eindpunt

Een interessante heuristiek is de afstand tussen het begin- en eindpunt van ieder te bouwen pad. Je kan deze heuristiek gebruiken om eerst de vermoedelijk kortste paden te bouwen en daarna pas de langere paden of juist andersom. Onderzoek zelf wat het efficiëntste werkt.

Bouw de paden naargelang de positie van begin- en eindpunt

Een alternatieve heuristiek is de positie van het begin- en eindpunt van ieder te bouwen pad. Je kan deze heuristiek gebruiken om eerst paden te bouwen tussen punten die aan de randen van de puzzel liggen en daarna pas paden te bouwen tussen punten die in het midden van de puzzel liggen of juist andersom. Onderzoek zelf wat het efficiëntste werkt.

Bouw zoveel mogelijk rechte paden

Het bouwen van zoveel mogelijk rechte (deel)paden is een andere tactiek die tot efficiëntere oplossingen kan leiden. Volg tijdens het bouwen van een pad de huidige richting in plaats van voortdurend van richting te veranderen of telkens een willekeurige richting te kiezen.

Sorteren volgens een zelf bepaald criterium

Prolog biedt een predicaat `predsort/3` aan dat een gegeven lijst sorteert met behulp van een gegeven predicaat. Raadpleeg de handleiding[2] en beschouw volgend bondig voorbeeld.

```
sorteer_sleutel(<,(Sleutel1,_),(Sleutel2,_)) :-
    Sleutel1 > Sleutel2.

sorteer_sleutel(>,(Sleutel1,_),(Sleutel2,_)) :-
    Sleutel1 =< Sleutel2.

sorteer_paren(InvoerLijst,UitvoerLijst) :-
    predsort(sorteer_sleutel,InvoerLijst,UitvoerLijst).

?- sorteer_paren([(3,a),(2,b),(4,d),(1,c)],GesorteerdeLijst).
GesorteerdeLijst = [(4,d),(3,a),(2,b),(1,c)].
```

6 Aandachtspunten

De beoordeling van je oplossing is niet alleen afhankelijk van het aantal puzzels dat je programma correct oplost en hoe snel het dit doet. De mate waarin je oplossing de richtlijnen voor het schrijven van Prolog-code[3] volgt, heeft eveneens een significant aandeel in de beoordeling. Deze sectie bevat een bondig overzicht van de belangrijkste aandachtspunten.

- **Correctheid:** Zorg ervoor dat je code compileert op de departementale computers. Vermijd waarschuwingen over variabelen die in het hoofd van een clause voorkomen, maar niet in het lichaam ervan (*singleton variables*). Plaats alle code in één bestand.
- **Documentatie:** Schrijf voldoende documentatie. Beschrijf zowel de algemene oplossingsstrategie die je volgt als de heuristieken en optimalisaties die je toepast bovenaan het bestand meteen na de hoofding met je naam en studierichting. Indien je oplossing niet (volledig) werkt, beschrijf je hier ook de problemen waarmee je te kampen hebt. Beschrijf de werking van ieder afzonderlijk predicaat vlak boven de verschillende clauses van dat predicaat. Hou documentatie en code zoveel mogelijk gescheiden en vermijd daarom ook code en documentatie op één en dezelfde regel.
- **Elegantie:** Plaats ieder doel in het lichaam van een clause op een aparte regel, voorafgegaan door een vast aantal spaties (bijvoorbeeld vier) of één tab. Wees consistent en gebruik dus nooit spaties en tabs door elkaar. Plaats de verschillende clauses van een predicaat steeds vlak bij elkaar. Vermijd witregels in het lichaam van een clause.

[2] Raadpleeg `http://www.swi-prolog.org/pldoc/man?predicate=predsort/3` voor meer informatie.
[3] Richtlijnen om Prolog-code te schrijven zijn beschikbaar via het onderdeel “Documenten” op Toledo.

- **Leesbaarheid:** Geef ieder predicaat een duidelijke en ondubbelzinnige naam die zijn functie beschrijft (bijvoorbeeld `bereken_kortste_pad`). Geef iedere variabele een duidelijke en ondubbelzinnige naam die zijn inhoud beschrijft (bijvoorbeeld `KortstePad`).
- **Compactheid:** Vermijd erg lange predicaten door ze in verschillende kleinere predicaten op te splitsen die elk een eigen taak uitvoeren. Verwijder ongebruikte predicaten of plaats ze in commentaar en documenteer hun beoogde functie in je oplossing.
- **Efficiëntie:** Zorg er eerst voor dat je programma aan alle bovenstaande richtlijnen voldoet alvorens je de efficiëntie ervan probeert te verbeteren. Appendix A bevat de uitvoeringstijden voor een aantal voorbeeldpuzzels van een niet-geoptimaliseerde implementatie die de voorgestelde oplossingsstrategie uit sectie 4 volgt. We verwachten dat je oplossing puzzels 0 tot en met 9 in één minuut op kan lossen. We gebruiken de moeilijkere puzzels 10 en 11 vooral om de efficiëntie van je oplossing te beoordelen.

7 Praktische informatie

Los deze opgave zelfstandig op en dien je oplossing uiterlijk op **donderdag 8 november om 18u00** in via Toledo. Het practicum maakt deel uit van het examen dus samenwerken is strikt verboden! Het practicum is bovendien een uitstekende voorbereiding op het examen.

Zorg ervoor dat je programma compileert en werkt op de computers van het Departement Computerwetenschappen[4]. Indien je oplossing niet (helemaal) werkt, vermeld dan duidelijk in commentaar wat er precies foutloopt. Zorg ervoor dat je oplossing strikt de opgelegde specificatie volgt en dus niet afwijkt van de opgelegde predicaatnamen.

Vermeld helemaal bovenaan het bestand `oplossing.pl`, op de eerste twee regels, de lijnen

```
% Voornaam Naam
% Studierichting
```

waarbij je uiteraard jouw eigen naam en studierichting invult. Schrijf als studierichting *Bachelor informatica*, *Schakelprogramma toegepaste informatica*, *Master computerwetenschappen* of *Andere* indien je geen enkele van de drie voorgaande studierichtingen volgt.

Dien je oplossing in via het onderdeel "Toetsen en opdrachten" op Toledo. Wees aandachtig, want je kan je oplossing slechts één keer indienen. Het volstaat om het bestand `oplossing.pl` in te dienen. De meegeleverde bestanden `puzzels.pl` en `visualisatie.pl` dien je dus niet mee in. Laat de oproepen die deze twee bestanden invoegen, namelijk `ensure_loaded(puzzels)` en `ensure_loaded(visualisatie)`, wel in je oplossing staan.

Richt je vragen bij deze opgave aan Jan Van Haaren via `jan.vanhaaren@kuleuven.be`.

Veel succes!

[4]Instructies om thuis of op kot de departementale computers te gebruiken zijn beschikbaar op Toledo.

A Oplossingen van voorbeeldpuzzels

Deze appendix bevat de oplossingen van de meegeleverde puzzels en de uitvoeringstijden van een implementatie die de voorgestelde oplossingsstrategie toepast. Deze uitvoeringstijden zijn bekomen op `brugge` in het softwareontwerplabo noord. De uitvoeringstijden zijn slechts indicatief en kunnen nog verder gereduceerd worden via gerichte optimalisaties.

Puzzel 0 (5x5-puzzel met 4 paden, voorbeeld uit figuur 1)

```
4 _ 3 _ 2                   4 4 3 3 2
1 _ 4 _ _                   1 4 4 3 2
_ 2 3 _ _                   1 2 3 3 2
_ _ _ _ _                   1 2 2 2 2
_ _ _ _ 1                   1 1 1 1 1

% 45,933 inferences, 0.008 CPU in 0.008 seconds
```

Puzzel 1 (5x5-puzzel met 4 paden)

```
2 _ _ _ _                   2 2 4 4 4
1 2 _ 3 4                   1 2 4 3 4
_ _ _ _ _                   1 4 4 3 3
_ 4 _ 1 _                   1 4 1 1 3
_ _ _ 3 _                   1 1 1 3 3

% 8,659 inferences, 0.001 CPU in 0.001 seconds
```

Puzzel 2 (5x5-puzzel met 3 paden)

```
_ _ _ 3 _                   2 2 2 3 3
_ 3 _ 2 _                   2 3 2 2 3
_ _ _ _ _                   2 3 3 3 3
_ _ _ _ _                   2 2 2 2 2
1 _ _ 1 2                   1 1 1 1 2

% 362,537 inferences, 0.049 CPU in 0.049 seconds
```

Puzzel 3 (6x6-puzzel met 5 paden)

```
_ _ 1 _ _ _                 1 1 1 3 3 3
_ 2 3 _ 4 _                 1 2 3 3 4 3
1 _ _ _ _ 3                 1 2 2 2 4 3
_ _ _ _ _ _                 4 4 4 2 4 4
4 5 _ _ 2 _                 4 5 4 2 2 4
5 _ _ _ _ _                 5 5 4 4 4 4

% 468,456 inferences, 0.052 CPU in 0.052 seconds
```

Puzzel 4 (7x7-puzzel met 7 paden)

```
1 _ 2 _ 2 3 _               1 1 2 2 2 3 3
4 _ 5 _ _ 6 _               4 1 5 5 5 6 3
_ _ _ _ 5 _ _               4 1 1 1 5 6 3
_ _ _ _ 6 _ 3               4 4 4 1 6 6 3
1 7 4 _ _ _ _               1 7 4 1 1 1 1
_ _ _ _ _ 7 _               1 7 7 7 7 7 1
_ _ _ _ _ _ _               1 1 1 1 1 1 1

% 11,524,951 inferences, 1.261 CPU in 1.261 seconds
```

Puzzel 5 (7x7-puzzel met 6 paden)

```
_ _ _ _ _ _ 1               1 1 1 1 1 1 1
_ _ _ _ _ 2 3               1 2 2 2 2 2 3
_ 2 _ _ _ _ _               1 2 3 3 3 3 3
_ _ _ 4 5 _ _               1 3 3 4 5 5 5
_ _ 4 _ 6 _ _               1 3 4 4 6 6 5
_ _ _ _ 3 6 _               1 3 3 3 3 6 5
_ _ _ _ _ 1 5               1 1 1 1 1 1 5

% 8,721,292 inferences, 0.967 CPU in 0.967 seconds
```

Puzzel 6 (7x7-puzzel met 5 paden)

```
_ _ _ _ _ _ _                1 1 1 1 1 1 1
1 _ _ _ _ _ _                1 5 5 5 5 5 1
_ _ 2 _ _ _ _                5 5 2 2 2 5 1
_ _ _ _ 2 _ _                5 4 4 4 2 5 1
_ _ 3 _ 4 _ _                5 4 3 4 4 5 1
5 4 _ _ 3 5 _                5 4 3 3 3 5 1
1 _ _ _ _ _ _                1 1 1 1 1 1 1

% 78,675,602 inferences, 8.667 CPU in 8.670 seconds
```

Puzzel 7 (7x7-puzzel met 5 paden)

```
_ _ _ 1 2 _ _                1 1 1 1 2 2 2
_ _ _ _ 3 4 _                1 5 5 5 3 4 2
_ 5 3 _ _ _ _                1 5 3 5 3 4 2
_ 2 _ 5 _ _ _                1 2 3 5 3 4 2
_ _ _ _ _ 4 _                1 2 3 3 3 4 2
_ _ _ _ _ _ _                1 2 2 2 2 2 2
_ _ _ _ _ _ 1                1 1 1 1 1 1 1

% 161,090,139 inferences, 17.754 CPU in 17.760 seconds
```

Puzzel 8 (7x7-puzzel met 5 paden)

```
_ _ _ _ _ _ _                1 1 1 1 1 1 1
1 2 _ _ _ 2 _                1 2 2 2 2 2 1
3 1 _ _ _ _ _                3 1 1 1 1 1 1
_ 4 _ 4 _ _ _                3 4 4 4 3 3 3
_ 5 3 _ _ 5 _                3 5 3 3 3 5 3
_ _ _ _ _ _ _                3 5 5 5 5 5 3
_ _ _ _ _ _ _                3 3 3 3 3 3 3

% 186,585,748 inferences, 20.496 CPU in 20.503 seconds
```

Puzzel 9 (8x8-puzzel met 7 paden)

```
_ _ _ _ 1 2 1 _          1 1 1 1 1 2 1 1
_ _ _ _ 3 _ 4 _          1 6 6 6 3 2 4 1
_ _ 5 _ _ _ _ _          1 6 5 6 3 2 4 1
_ _ _ _ _ 2 4 _          1 6 5 6 3 2 4 1
_ _ 5 6 _ _ _ _          1 6 5 6 3 3 3 1
_ _ _ _ _ 7 _ _          1 6 7 7 7 7 3 1
_ 6 7 3 _ _ _ _          1 6 7 3 3 3 3 1
_ _ _ _ _ _ _ _          1 1 1 1 1 1 1 1
```

```
% 94,563,161 inferences, 10.275 CPU in 10.279 seconds
```

Puzzel 10 (8x8-puzzel met 7 paden)

```
1 _ _ _ _ _ _ _          1 3 3 3 3 3 3 3
_ _ _ _ _ _ 2 3          1 3 2 2 2 2 2 3
_ _ _ 4 _ _ _ _          1 3 2 4 4 4 4 4
_ 3 _ _ _ _ 2 4          1 3 2 2 2 2 2 4
_ _ _ _ 5 _ _ 6          1 5 5 5 5 6 6 6
_ _ 6 _ _ _ _ 7          1 5 6 6 6 6 7 7
_ _ 5 7 _ _ _ 1          1 5 5 7 7 7 7 1
_ _ _ _ _ _ _ _          1 1 1 1 1 1 1 1
```

```
% 8,674,432,717 inferences, 945.054 CPU in 945.384 seconds
```

Puzzel 11 (9x9-bonuspuzzel met 9 paden)

```
_ _ _ _ 4 _ _ 6 _        9 9 9 9 4 4 4 6 6
9 7 5 _ _ _ _ 2 _        9 7 5 9 9 9 4 2 6
7 _ _ _ 3 _ _ _ 6        7 7 5 5 3 9 4 2 6
_ _ 8 _ _ _ _ _ _        8 8 8 5 3 9 4 2 2
_ _ _ _ _ _ _ 1 _        8 5 5 5 3 9 4 1 2
_ 5 _ _ _ _ _ _ _        8 5 3 3 3 9 4 1 2
_ _ _ 8 _ _ _ _ _        8 3 3 8 9 9 4 1 2
_ 3 _ _ _ 1 4 _ _        8 3 8 8 9 1 4 1 2
_ _ _ 9 _ _ _ _ 2        8 8 8 9 9 1 1 1 2
```

```
% 127,067,721,919 inferences, 13726.197 CPU in 13730.988 seconds
```