Kennis en renju

Peter-Paul Kruijsen

Doctoraalscriptie CS 01-01

Scriptie ingediend als gedeeltelijke vervulling
van de vereisten voor de titel van
doctorandus in de kennistechnologie
aan de Faculteit der Algemene Wetenschappen
van de Universiteit Maastricht

afstudeercommissie:
prof. dr. H.J. van den Herik
dr. ir. J.W.H.M. Uiterwijk
dr. E.O. Postma
dr. I.G. Sprinkhuizen-Kuyper

Universiteit Maastricht
Instituut voor Kennis- en Agent Technologie
Department of Computer Science
Maastricht, Nederland

juni 2001

Voorwoord

Voor u ligt de eindscriptie van Peter-Paul Kruijsen, student kennistechnologie, studierichting artificiële intelligentie, aan de Universiteit Maastricht. Deze scriptie is één van de eindresultaten van mijn afstudeerproject. Ik heb van september 1999 tot en met mei 2001 in deeltijd, doch zeer intensief aan dit project gewerkt. De overige resultaten van dit project zijn een computerprogramma waarover in deze scriptie wordt gesproken, en een presentatie die als verdediging geldt voor het onderzoek dat gedaan is tijdens het project.

Ik ben tijdens dit project voortdurend bijgestaan door dr. ir. J.W.H.M. Uiterwijk, mijn dagelijkse begeleider voor dit project. Hij heeft mij in de eerste oriënterende gesprekken gewezen op een mogelijk onderzoek naar het spel *renju*. Gezien mijn interesse voor een afstudeerproject op het gebied van computerspelen leek mij een keuze voor dit onderwerp een juiste. De goede samenwerking die mij voor ogen stond tussen de heer Uiterwijk en mijzelf is daadwerkelijk tot stand gekomen en behouden tijdens de gehele duur van dit project. Mijn dank gaat daarom in sterke mate uit naar de heer Uiterwijk voor alle kennis en inspiratie die hebben bijgedragen aan een goede afloop van dit project. Mijn dank gaat tevens uit naar de coördinator van dit project, prof. dr. H.J. van den Herik en verder naar dr. E.O. Postma en dr. I.G. Sprinkhuizen-Kuyper, die zo vriendelijk waren plaats te nemen in de afstudeercommissie die dit project beoordeelt.

Ik wil graag de volgende mensen uit mijn naaste omgeving bedanken die mij gedurende mijn afstudeerproject hebben bijgestaan en hebben bijgedragen aan het tot stand komen van deze scriptie: Andrea, Andreas, Anneke, Bart, Chris, Diederik, Emmy, Evert-Jan, Felicia, Freek, Gabriel, Govinda, Henk-Jelle, Inge, Jan-Willem, Janneke, Jos, Karlijn, Linda, Marieke, Marijn, Marion, Michiel, Mik, Peter, Ranco, Raymond, Sabine, Sara, Titia en Wouter.

Ik wil mijn afstudeerwerk opdragen aan de drie belangrijkste personen in mijn leven. Allereerst aan mijn oma, die met zeer veel liefde de vorderingen van mijn studie heeft gevolgd en mij altijd heeft gemotiveerd en geïnspireerd om mijn studie te voltooien; ten tweede aan mijn vader, die vol trots zowel op spiritueel als op materieel gebied alles wat in zijn macht ligt heeft gegeven om mijn studie tot een groot succes te maken en tenslotte aan mijn vriendin Mirjam, die mij voortdurend en volledig laat genieten van de liefde en het leven zelf.

Peter-Paul Kruijsen
Juni 2001

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Samenvatting

Door invloeden vanuit de kunstmatige intelligentie wordt steeds vaker gekozen voor een zo transparant mogelijke opslag van kennis in informatiesystemen. In deze traditie wordt hier een opzet voorgesteld waarbij de ontwikkelde kennis en het computerprogramma gescheiden worden. Om de communicatie tussen beide te waarborgen wordt een formele taal voorgesteld die gebaseerd is op de predikaatlogica. Het domein waarin het systeem is opgesteld is het Japanse bordspel renju. De kenniselementen beschrijven winstsituaties binnen renju en worden opgesteld in de formele taal. Experimenten laten zien dat in het ontwikkelde renjuprogramma met behulp van deze kenniselementen het zoekproces verkort wordt. Het resultaat hiervan is zelfs beter dan bij een vergelijkbare opstelling zonder kenniselementen: de speelsterkte neemt toe. Een bijkomend voordeel is dat met de ontwikkelde techniek kenniselementen aangepast of toegevoegd kunnen worden zonder dat hiervoor de code van het programma veranderd dient te worden.

Summary

Through influences from the artificial-intelligence domain, the process of storing knowledge in an information system is increasingly made as transparent as possible. In the context of this new tradition, a scheme is proposed that separates the set of developed knowledge from the actual computer program. To guarantee a proper communication between them, a formal language is proposed, based on first-order logic. The domain of the system is the Japanese board game renju. Won positions in the game are described by knowledge elements that are precisely formalized within the newly-developed formal language. Experiments show that in a renju computer program, the search process can be reduced when knowledge elements are used. The result is even better then in an equipollent trade-off: the playing strength of the program increases. An additional benefit of this approach is that the knowledge elements can be changed or added without changing code of the program itself.

Hoofdstuk 1

Inleiding

Deze scriptie is, zoals de titel ervan doet vermoeden, opgebouwd uit twee hoofdonderwerpen. Allereerst is er *kennis*. Het is het belangrijkste onderwerp binnen de kennistechnologie en daarom een haast verplicht deel van deze scriptie. Het gedeelte dat hier behandeld wordt, is het opslaan van kennis binnen een kennissysteem. Door invloeden uit de kunstmatige intelligentie zijn de opvattingen hierover in de loop der tijd flink veranderd. Waar vroeger de kennis diep in de structuur van een programma verscholen lag, wordt tegenwoordig steeds meer gepoogd de kennis op een overzichtelijke manier op te slaan zodat ze makkelijk bijgehouden kan worden [5]. Hierdoor komt het typisch dynamische karakter van de kennis steeds beter tot haar recht. Een eerste stap in deze richting is de scheiding van kennis en inferentiemechanisme. De kennis wordt hierbij in een aparte verzameling bijgehouden zodat zij makkelijk terug te vinden is en daardoor eenvoudiger gewijzigd kan worden. Er wordt een inferentiemechanisme gebruikt dat in staat is met de kennis te redeneren en er conclusies uit te trekken. De volgende stap die gemaakt kan worden, is om de kennis niet langer als verzameling binnen het programma te plaatsen, maar juist erbuiten. Dit geeft de gebruiker van het programma de mogelijkheid zelf de kennis te onderhouden zonder dat hierbij de hulp van een programmeur nodig is om de code van het programma te wijzigen. Los van een inferentiemechanisme is hierbij tevens een onderdeel nodig dat controleert of het formaat van de kennis overeenkomt met het formaat dat het inferentiemechanisme hanteert. Deze laatste aanpak is in dit project gekozen voor het bijhouden van de kennis.

Het tweede onderwerp, *renju*, komt terug in de toepassing. Renju [7] is een Japans bordspel dat zijn oorsprong vindt rond het begin van onze jaartelling. Het wordt gespeeld door twee opponenten, hier genoemd *Zwart* en *Wit*. Zwart begint het spel. Het doel is om als eerste speler een aaneengesloten rij van vijf stenen van de eigen kleur te plaatsen op een vlak bord met afmetingen 15×15. Dit mag zowel horizontaal, verticaal als diagonaal. Het spel valt in dezelfde klasse van spelen als boter-kaas-en-eieren, namelijk de klasse van mnk-spelen [9, 11]. Het doel van een mnk-spel is om als eerste speler een aaneengesloten rij van k stenen van de eigen soort op een vlak bord met afmetingen $m \times n$ te plaatsen. Deze definitie stelt dat renju een 15-15-5-spel is, net als het spel go-moku. Het verschil tussen renju en go-moku is dat renju enkele extra spelregels hanteert die Zwart in zijn spel beperken. Deze regels zijn in de loop der tijd ontstaan toen bleek dat Zwart bij go-moku

te veel voordeel had. Dit voordeel komt voort uit het gegeven dat Zwart de eerste zet mag doen. Deze spelregels maken het wetenschappelijk interessant en verantwoord om een onderzoek te wijden aan renju. Beide spelen zijn in het verleden opgelost [1, 2, 12].

Het doel van het combineren van kennis en renju is het ontwikkelen van een programma dat enerzijds in staat is het spel renju te spelen en anderzijds gebaseerd is op een verzameling kennis die onafhankelijk van het programma bijgehouden kan worden. Om een computerprogramma renju te laten spelen, zal deze los van de spelregels van het spel ook kennis moeten hebben over het doen van een sterke zet, gegeven een bepaalde positie. Een dergelijk programma zal vooruit moeten kijken naar vervolgposities om op basis van evaluatie een zo gunstig mogelijk scenario te selecteren. Het zoekproces dat hierbij ontstaat zou ingekort kunnen worden naarmate het programma meer kennis heeft over de eindfasen van een spel. Deze kennis zou moeten blijken uit het herkennen van spelsituaties die voor één van beide spelers een gegarandeerde winst opleveren. Het beschrijven van dergelijke situaties zou moeten resulteren in een aantal kennisregels [10]. De opslag van deze kennisregels zal buiten het systeem liggen, zoals eerder is beschreven. Om het formaat van deze kennisregels te bepalen zal gebruik gemaakt worden van een formele taal, die beschrijft welke elementen gebruikt kunnen worden om de kennisregels op te stellen.

De combinatie van kennis en renju komt duidelijk terug in de probleemstelling die voor dit project als volgt geformuleerd is:

> Is het mogelijk een renjuprogramma te maken dat gebaseerd is op extern opgeslagen kennisregels?

Deze hoofdvraag leidt tot vijf doelstellingen. De eerste hiervan is het ontwikkelen van een formele taal waarbinnen kennisregels geformuleerd kunnen worden die renjusituaties beschrijven. De tweede doelstelling volgt hier direct uit en is het daadwerkelijk opstellen van deze kennisregels. Het derde doel is het ontwikkelen van een inferentiemechanisme dat in staat is de kennisregels te begrijpen en ermee te redeneren. Hierbij sluit het vierde doel aan, namelijk het ontwikkelen van een computerprogramma dat in staat is het spel renju te spelen op basis van de kennisregels en het inferentiemechanisme. De vijfde en laatste doelstelling is het onderzoeken of het zoekproces van het programma ingekort kan worden naarmate meer kennis aan het programma wordt aangeboden en zo ja, met welke resultaten.

Het vervolg van deze scriptie is als volgt opgebouwd. In hoofdstuk 2 wordt een theoretische achtergrond gegeven over concepten die zijn gebruikt bij het ontwikkelen van een formele taal. In hoofdstuk 3 worden de spelregels van renju en de ontwikkelde kenniselementen beschreven. Onder de ontwikkelde kenniselementen vallen ondermeer de kennisregels. Hoofdstuk 4 bevat een beschrijving van het programma. Hier wordt vermeld op welke manier het programma het spel renju speelt en hoe het inferentiemechanisme werkt. In hoofdstuk 5 worden de experimenten beschreven die met het programma zijn uitgevoerd alsmede de resultaten die hierbij zijn behaald. In hoofdstuk 6 wordt bekeken in hoeverre de hierboven genoemde doelstellingen zijn gerealiseerd en worden de conclusies opgesomd die aan de hand van het onderzoek, de experimenten en de resultaten in een eerder stadium zijn getrokken.

Hoofdstuk 2

De theorie

Dit hoofdstuk beschrijft de verschillende concepten die gebruikt zijn voor het opstellen van een formele taal. Een dergelijke taal is nodig om een goede communicatie tussen het programma en de externe kennis te garanderen. De natuurlijke talen zoals mensen die gebruiken bevatten ambiguïteiten waardoor een precieze communicatie verstoord kan worden. Denk hierbij aan het Nederlandse woord *bank*. Mensen zullen uit de context de betekenis van het woord kunnen afleiden, maar voor een computer kan een dergelijke meervoudige betekenis tot ongewenste fouten leiden. Het is om deze reden dat een formele taal noodzakelijk is. Dit is een taal waarbij zowel het formaat als de betekenis van de onderdelen van de taal eenduidig vastgelegd zijn.

2.1 De taal

Voor dit project is een dergelijke formele taal ontwikkeld. Deze taal heeft de naam Blanche meegekregen en is vergelijkbaar met de predikaatlogica [3]. De aard van predikaatlogica is dat eigenschappen van entiteiten door middel van predikaten aan de entiteiten gekoppeld worden. Het predikaat beschrijft hierbij de eigenschap. De entiteiten waar het predikaat iets over zegt, worden als argumenten meegegeven. Op deze manier kunnen feiten binnen een bepaald domein beschreven worden.

> Van een entiteit *deze_zonnebloem* zou kunnen worden aangegeven dat deze een gele kleur heeft. De zin in natuurlijke taal *deze zonnebloem heeft een gele kleur* heeft als mogelijk equivalent in predikaatlogica de zin *gele_kleur(deze_zonnebloem)*. Het predikaat *gele_kleur* heeft één argument. Er kan ook gebruik gemaakt worden van het breder inzetbare predikaat *kleur* dat met twee argumenten gebruikt wordt. Dezelfde zin kan dan beschreven worden met *kleur(deze_zonnebloem, geel)*, waarbij *geel* als een aparte entiteit gezien wordt. Op deze manier kunnen relaties tussen entiteiten weergegeven worden.

Het gebruik van predikaten binnen Blanche gebeurt op analoge wijze. Een verschil tussen Blanche en predikaatlogica is dat in Blanche geen functies gebruikt kunnen worden.

Functies zijn constructies waarbij een combinatie van een predikaat en argumenten gebruikt wordt om entiteiten aan te geven. Een voorbeeld hiervan is de zin *Jan's vader en moeder zijn getrouwd*, die in predikaatlogica beschreven kan worden door *getrouwd(vader(jan), moeder(jan))*. Hierbij is *getrouwd* een predikaat; *vader* en *moeder* zijn twee functies die met betrekking tot *jan* verwijzen naar twee nieuwe entiteiten, namelijk Jan's vader en moeder. Een dergelijke constructie is binnen BLANCHE niet mogelijk. De reden hiervoor is dat in de ontwerpfase werd vermoed dat dit voor het huidig onderzoek overbodig zou zijn. Achteraf is dit vermoeden juist gebleken.

Om een formele taal volledig te definiëren is het noodzakelijk twee zaken duidelijk vast te leggen. Dit zijn de syntax en de semantiek. De syntax van een taal bepaalt het formaat van de afzonderlijke elementen van de taal en hoe deze elementen gecombineerd worden. Het legt vast welke constructies wel en welke niet toegestaan zijn. Een voorbeeld van een syntactische regel in de Nederlandse taal is dat alle eigennamen met een hoofdletter dienen te beginnen. Een goed en overzichtelijk hulpmiddel bij het opstellen van syntactische regels is het *Backus-Naur formalisme* [6]. Deze techniek biedt een notatievorm waarmee onder andere syntactische regels eenvoudig en eenduidig weergegeven kunnen worden. Aan het eind van dit hoofdstuk zullen de syntactische regels van BLANCHE aan de hand van een Backus-Naur formalisme getoond worden. Wanneer eenmaal is vastgelegd welke constructies toegestaan zijn binnen de taal, moet nog vastgelegd worden wat de betekenis ervan is. De semantiek van een taal kent aan ieder element van die taal binnen een context een betekenis toe. Zo ontstaat als het ware een woordenboek van de taal. De semantische regels die voor BLANCHE zijn opgesteld, worden in hoofdstuk 3 behandeld.

2.2 De woorden

In deze sectie zullen de afzonderlijke onderdelen van de taal uitgebreider besproken worden. Deze onderdelen worden hier vergeleken met de woorden in een natuurlijke taal. Deze vergelijking wordt gemaakt voor drie soorten elementen.

Predikaten De eerste soort elementen die besproken worden, zijn de predikaten. Zoals eerder is besproken, worden zij gebruikt om eigenschappen aan entiteiten toe te kennen. In dit opzicht zijn predikaten vergelijkbaar met bijvoeglijke naamwoorden in een natuurlijke taal. Syntactisch is vastgelegd dat ieder predikaat met een kleine letter moet beginnen.

Argumenten Argumenten vormen de tweede soort elementen binnen BLANCHE. De argumenten zijn vergelijkbaar met zelfstandige naamwoorden in een natuurlijke taal. Het zijn woorden die staan voor een bepaalde entiteit. Er is een duidelijk onderscheid te maken tussen twee typen argumenten.

Het eerste type argumenten zijn de constanten. Dit zijn entiteiten die daadwerkelijk bij naam genoemd worden. Zij worden gebruikt om een entiteit direct aan te spreken. Hierdoor kunnen bijvoorbeeld eigenschappen van een entiteit bepaald of gewijzigd worden.

Het tweede type wordt gevormd door de variabelen. Dit type argumenten staat voor een verwijzing naar een niet nader genoemde entiteit binnen een bepaalde klasse. Zij kunnen gekoppeld worden aan een specifieke entiteit door middel van een waardetoekenning of zij kunnen gebuikt worden om iets te zeggen over alle entiteiten binnen een bepaalde klasse.

Om het verschil tussen constanten en variabelen duidelijk te maken, geldt de syntactische regel dat de eerste letter van een constante een kleine letter of een cijfer moet zijn. De eerste letter van een variabele moet juist altijd een hoofdletter zijn. Om anonieme variabelen te definiëren wordt gebruik gemaakt van de underscore (_) in de naam van de variabele. Met een anonieme variabele wordt een variabele bedoeld die wel nodig is voor een bepaald doel maar die verder niet gebruikt zal worden. Een voorbeeld van een situatie waarin een anonieme variabele gebruikt kan worden is de volgende. Een persoon is moeder indien deze persoon vrouwelijk is en er een persoon bestaat die kind is van de te onderzoeken persoon. Los van een variabele voor de te onderzoeken persoon kan een aparte variabele geïntroduceerd worden voor het kind. Omdat het voor de bewering niet belangrijk is om welk kind het gaat, kan het met een anonieme variabele aangeduid worden. Bij de naamgeving die hier gebruikt wordt, krijgt de te onderzoeken persoon als variabelenaam *Persoon* of *Persoon1* en het kind *Persoon_* of *Persoon_1*. Het gebruik van volgnummers vindt plaats wanneer er meerdere variabelen van een bepaald type beschreven moeten worden.

Operatoren De laatste soort elementen, operatoren genaamd, wordt gebruikt om meer complexe structuren te creëren die in de volgende sectie besproken zullen worden. Om aan te geven dat een predikaat niet geldt in een bepaalde situatie, is een definitie van het woord *niet* opgesteld. Het wordt in Blanche gerepresenteerd door het symbool ¬. Het voegwoord *en* wordt in Blanche weergegeven door een komma. Het wordt gebruikt om predikaten aan elkaar te koppelen. Het precieze gebruik zal duidelijk worden in de volgende sectie. Het voegwoord *indien* kan gebruikt worden om afleidingen te definiëren. Ook hierover volgt meer in de volgende sectie. Het symbool :- (spreek uit: *denotes*) wordt gebruikt om dit woord aan te duiden. Het voegwoord *of* wordt gerepresenteerd door gebruik te maken van twee aparte uitdrukkingen. Het systeem zal door zijn interne opbouw herkennen dat een bewering waar is wanneer ofwel de ene ofwel de andere uitdrukking waar is, of eventueel beide. Er is geen apart symbool gedefinieerd voor het dit woord.

2.3 De zinnen

Door het combineren van predikaten, argumenten en operatoren ontstaan structuren die te vergelijken zijn met zinnen in een natuurlijke taal. Deze structuren zijn er in verschillende complexiteiten. De meest eenvoudige vorm is de *tekenloze literal*. Dit is niets meer dan de combinatie van een predikaat met één of meer argumenten. Een *literal* is een tekenloos literal, eventueel voorafgegaan door een niet-teken. De combinatie van één of meer literals,

gescheiden door komma's, wordt een *sentence* genoemd.

Om het definiëren van nieuwe predikaten aan de hand van andere predikaten mogelijk te maken zijn de *afleidingen* geïntroduceerd. Een afleiding bestaat uit een tekenloos literal, gevolgd door een indien-teken en een sentence. Een afleiding creëert het predikaat dat in de tekenloze literal voorkomt en stelt dat het geldt indien alle literals uit de sentence gelden. In het volgende hoofdstuk zal blijken dat afleidingen een erg krachtig hulpmiddel vormen.

De laatste structuur die gedefiniëerd wordt is die van de *kennisregels*. Het doel van een kennisregel is om gewonnen posities te herkennen. De beschrijving van die situatie is terug te vinden in de *premisse* van de kennisregel. Deze premisse bestaat uit een sentence. Wanneer aan alle literals in deze sentence voldaan wordt, is de kennisregel van toepassing in de onderzochte positie. Een kennisregel leidt in de meeste situaties tot een zet die gespeeld moet worden om het pad naar de winst met een stap te verkorten. In de situaties waarin dit niet gebeurt is een kennisregel van toepassing die een positie beschrijft die reeds gewonnen is. In een dergelijke positie dienen geen zetten meer gedaan te worden. Verder bevat een kennisregel een kleur die aangeeft voor welke van beide spelers de kennisregel geldt en een niveau dat aangeeft in hoeveel zetten de winst in het spel behaald kan worden.

De syntactische regels zijn vastgelegd in figuur 2.1, die gebruik maakt van het Backus-Naur formalisme. De volgende notatievormen worden hierbij gebruikt:

Naam Aan ieder concept dat gedefinieerd wordt, wordt een naam toegekend. Op deze manier kunnen concepten later direct aangesproken worden. Deze naam wordt linksboven de grafische weergave van een concept vermeld.

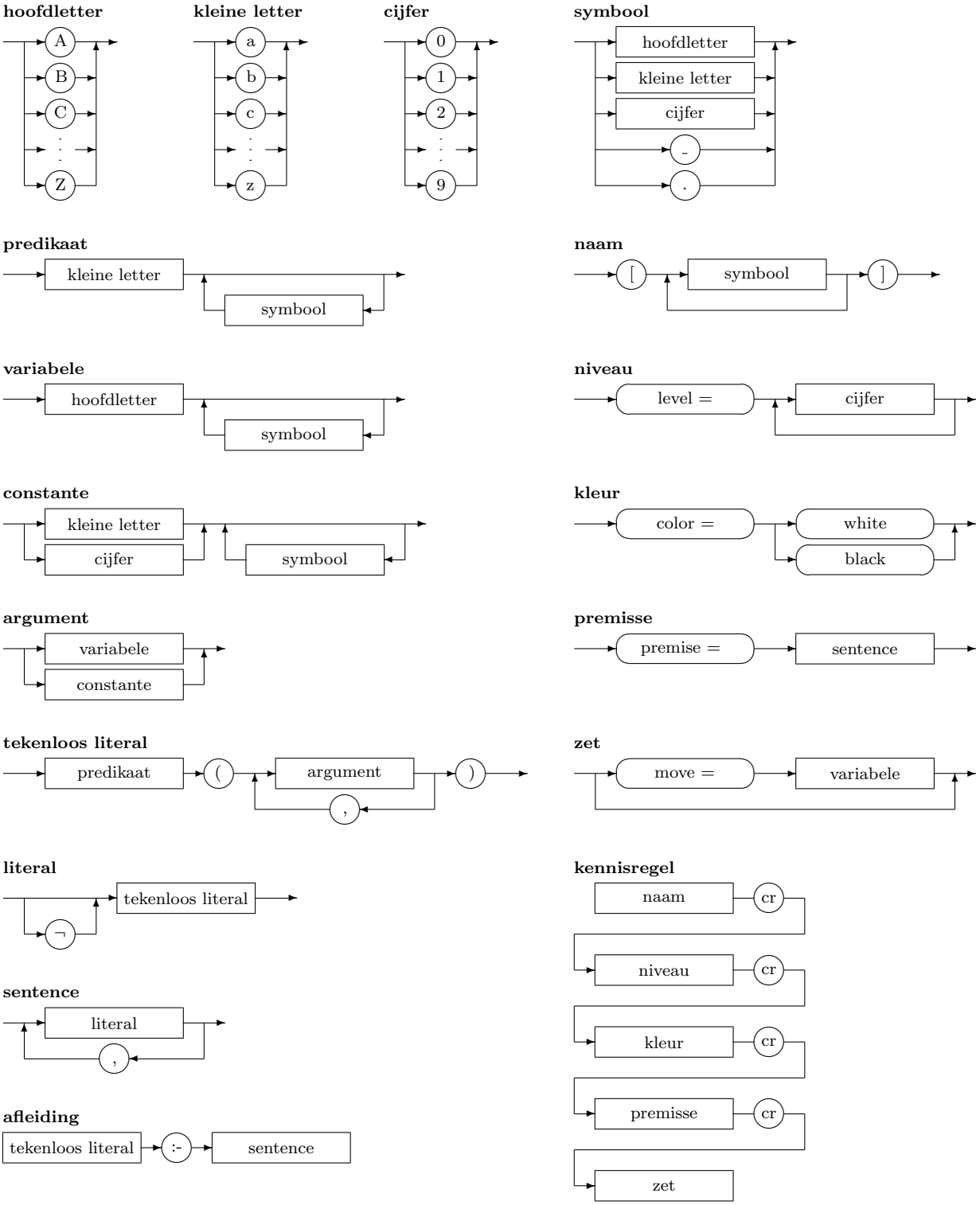
Rechthoeken De namen van eerder gedefinieerde concepten kunnen bij de weergave van een nieuw concept als afkorting gebruikt worden. Dit gebeurt door de naam van het te gebruiken concept in een rechthoek te plaatsen. Dit geeft de mogelijkheid om in plaats van de eerdere definitie van een concept slechts de naam van het concept in een rechthoek te vermelden.

Ovalen en cirkels Om teksten die letterlijk in de kennisregels moet voorkomen weer te geven worden ovalen en cirkels gebruikt. Het verschil tussen beide is dat in een cirkel een enkel karakter wordt aangegeven en in een ovaal een combinatie van karakters. In de ovalen worden begrippen in de Engelse taal vermeld omdat de kennisregels in deze taal opgesteld zijn. In figuur 2.1 wordt een cirkel met het symbool `cr` (carriage return) gebruikt om het einde van een regel in een tekstbestand aan te geven.

Pijlen Deze worden gebruikt om sequenties van rechthoeken, ovalen en cirkels te vormen. Een splitsing van pijlen betekent een keuzemoment waarbij één van de pijlen na de splitsing gekozen dient te worden.

In de figuur wordt duidelijk hoe de besproken elementen opgebouwd zijn en hoe ze gebruikt worden in een groter geheel. De eindconcepten die geconstrueerd zijn, zijn de afleidingen en de kennisregels. Enkel bij deze twee concepten worden geen pijlen bij start en einde gebruikt. Het is belangrijk om in te zien dat de syntactische regels slechts het formaat

van de besproken onderdelen vastleggen. Aan de hand van deze syntactische regels zijn afleidingen en kennisregels te formuleren die wellicht het goede formaat hebben, maar absoluut niets betekenen. De betekenis van de geformuleerde afleidingen en kennisregels wordt niet door de syntax gegeven, maar is onderdeel van de semantiek, die in het volgende hoofdstuk behandeld zal worden.



Figuur 2.1: De syntactische regels van Blanche.

Hoofdstuk 3

De toepassing

In dit hoofdstuk wordt besproken hoe BLANCHE een invulling kan krijgen binnen het domein van het spel renju. Het hoofdstuk begint met een uitgebreide uitleg van de spelregels die binnen renju gelden. Vervolgens worden verschillende kenniselementen besproken die ontwikkeld zijn binnen BLANCHE. De belangrijkste elementen zijn hierbij de afleidingen en kennisregels.

3.1 Het spel renju

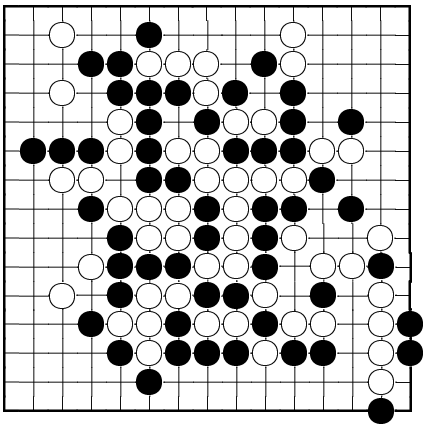
Zoals in hoofdstuk 1 reeds beschreven staat, behoort renju tot de klasse van *mnk*-spelen. Het wordt gespeeld tussen twee opponenten, hier *Zwart* en *Wit* genoemd, door beurtelings zetten te doen met zwarte en witte stenen op een plat speelbord met een rooster, gevormd door 15 horizontale en 15 verticale lijnen. In tegenstelling tot schaken worden de stukken, in het geval van renju de stenen, niet geplaatst op de vlakken die door de lijnen worden begrensd, maar op de snijpunten van de lijnen. Deze snijpunten worden in het vervolg *velden* genoemd. Het speelbord bestaat hierdoor uit 225 velden. In figuur 3.1 wordt een positie getoond waaruit duidelijk blijkt hoe de velden gevormd worden. De horizontale lijnen worden van onder naar boven gelabeld van 1 tot en met 15. De verticale lijnen worden van links naar rechts gelabeld van *a* tot en met *o*. De velden worden gelabeld volgens de snijpunten van de lijnen waarop zij liggen. Het veld linksonder is *a1*.

Om de spelregels van renju te kunnen beschrijven wordt eerst een aantal begrippen geïntroduceerd.

Overline Een overline is een ononderbroken groep van zes of meer stenen van dezelfde kleur op een rij. Een rij kan zowel horizontaal, verticaal als diagonaal zijn.

Vijf-groep Een vijf-groep is een ononderbroken groep van vijf stenen van dezelfde kleur op een rij.

Vier-groep Een vier-groep is een groep van vier stenen van dezelfde kleur op een rij, die met één zet uitgebreid kan worden tot een vijf-groep.



Figuur 3.1: Voorbeeld van een renjupositie (Sinov vs. Soosyrv, Peking, 1999).

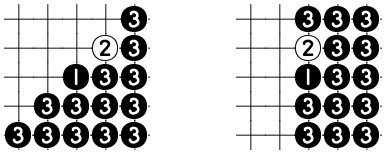
Open vier-groep Een open vier-groep is een ononderbroken groep van vier stenen van dezelfde kleur op een rij met aan beide kanten een open veld.

Drie-groep Een drie-groep is een groep van drie stenen van dezelfde kleur op een rij, die met één zet uitgebreid kan worden tot een open vier-groep.

Dubbel-vier Een zet die leidt tot de vorming van meer dan één vier-groep.

Dubbel-drie Een zet die leidt tot de vorming van meer dan één drie-groep.

De spelregels van renju zijn als volgt [13]. Zwart begint het spel door een zwarte steen te plaatsen op het middelste veld van het bord (*h8*). Zwart doet ook de tweede en derde zet, de tweede met een witte steen, de derde weer met een zwarte. De tweede steen moet exact één veld verwijderd zijn van de eerste steen. De derde steen moet één of twee velden verwijderd zijn van de eerste steen. Deze afstanden mogen zowel horizontaal, verticaal als diagonaal zijn. Vanwege symmetrie en spiegeling zijn er twee verschillende mogelijkheden voor de tweede zet en voor ieder van die zetten weer 13 mogelijkheden voor de derde zet. In totaal zijn er hierdoor 26 openingspatronen die zijn opgenomen in figuur 3.2. Deze figuur is in twee delen opgesplitst. De linker helft toont de eerste zwarte steen op het middelste veld van het bord, gemarkeerd met een 1. De witte steen daar schuin boven geeft één van de twee varianten voor de tweede zet. Rondom deze twee stenen zijn de 13 mogelijkheden



Figuur 3.2: De 26 openingspatronen van renju.

voor de derde zet weergegeven, alle gemarkeerd met een 3. Uit deze 13 zal één mogelijkheid door Zwart gekozen moeten worden als derde zet wanneer hij kiest voor een tweede zet schuin ten opzichte van de eerste zet. Wanneer hij de tweede zet direct naast de eerste zet plaatst zijn er ook 13 verschillende mogelijkheden voor de derde zet. Deze mogelijkheden worden volgens dezelfde methode in de rechterhelft van figuur 3.2 weergegeven.

Wit mag na de derde zet bepalen wie met de witte stenen gaat spelen en wie met de zwarte en heeft daarmee het recht op dit punt in het spel van kleur te wisselen. Dit is de zogenaamde *swap-rule*. Na deze keuze doet de speler die nu met de witte stenen speelt een zet op een willekeurig onbezet veld. Na deze zet doet Zwart twee voldoende verschillende[1] voorstellen voor de vijfde zet. Wit kiest een van deze twee zetten die Zwart vervolgens uitvoert als vijfde zet. Na de vijfde zet eindigen de openingsregels.

Het doel van het spel is, net als dat van go-moku, om als eerste speler een vijf-groep te creëren. De speler die hierin slaagt, wint het spel. In tegenstelling tot go-moku gelden er bij renju voor Zwart enkele beperkingen. Bepaalde zetten zetten zijn voor Zwart niet toegestaan, namelijk de zetten die, zonder dat tegelijk een vijf-groep gecreëerd wordt, leiden tot een overline, een dubbel-vier of een dubbel-vier. Dergelijke zetten leiden tot winst voor Wit. Wit heeft geen beperkingen en wint ook wanneer hij een overline creëert.

In de volgende twee gevallen is een dubbel-drie voor Zwart wel toegestaan:

1. Een dubbel-drie is toegestaan wanneer niet meer dan één van de drie-groepen op een punt uitgebreid kan worden tot een open vier-groep zonder dat op dit punt een overline of een dubbel-vier ontstaat. Wanneer minstens één van de groepen enkel uitgebreid kan worden ten koste van een overline of dubbel-vier, is de dubbel-drie van geen enkel nut voor Zwart en geldt deze niet als verboden situatie.

2. De tweede situatie waarin een dubbel-drie is toegestaan, is wanneer niet meer dan één van de drie-groepen op een punt kan worden uitgebreid tot een open vier-groep zonder dat op dit punt een verboden dubbel-drie wordt geceëerd. Ook hier geldt dat één of beide drie-groepen geen dreiging vormen wanneer ze slechts ten koste van een verboden dubbel-drie kunnen worden uitgebreid. Hiertoe zal recursief bekeken worden of de ontstane dubbel-drieën toegestaan zijn.

De basisregel om te achterhalen of een dubbel-drie toegestaan is, is om na te gaan of er dreiging van de beide groepen uitgaat. Indien dit voor minstens één van de groepen niet het geval is, omdat bij uitbreiding een verloren situatie zou ontstaan, is de dubbel-drie toegestaan en leidt de situatie niet tot winst voor Wit. Met een verboden zwarte dubbel-drie wordt een dubbel-drie voor Zwart aangeduid die niet is toegestaan.

Een spel eindigt ook in winst voor een speler wanneer zijn tegenstander opgeeft. Een spel eindigt in remise wanneer alle velden van het bord bezet zijn zonder dat één van beide spelers heeft gewonnen of wanneer beide spelers besluiten tot remise. In officiële toernooien wordt met een tijdslimiet gespeeld. De spelers moeten beide binnen een bepaalde

[1] De officiële spelregels zijn over dit begrip vrij onduidelijk. Gedoeld wordt op het niet toestaan van bijvoorbeeld posities die op een spiegeling na aan elkaar gelijk zijn.

tijdsspanne ofwel een vooraf overeengekomen aantal ofwel alle zetten doen. Wanneer een speler deze tijdslimiet overschrijdt, verliest hij het spel. Tabel 3.1 geeft een overzicht van uitkomsten van het spel en mogelijkheden die tot deze uitkomsten leiden.

Winst voor Zwart	Zwarte vijf-groep
	Wit geeft op
	Wit overschrijdt tijdslimiet
Winst voor Wit	Witte vijf-groep
	Witte overline
	Zwarte overline
	Zwarte dubbel-vier
	Verboden zwarte dubbel-drie
	Zwart geeft op
	Zwart overschrijdt tijdslimiet
Remise	Alle velden bezet
	Spelers besluiten remise

Tabel 3.1: Overzicht van de uitkomsten van een spel.

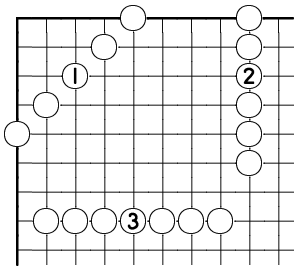
De genoemde spelregels over opgeven, remise en tijdslimieten zijn niet in dit onderzoek opgenomen. Van de openingsregels zijn enkel de 26 openingspatronen opgenomen. Het feit dat Zwart de eerste drie zetten doet, de mogelijkheid voor Wit om te wisselen na de derde beurt en de voorstellen van Zwart in de vijfde beurt zijn in het onderzoek niet meegenomen. Een compleet overzicht van de officiële spelregels is te vinden op de webpagina van de internationale renju-federatie [13].

3.2 De entiteiten

Om een computerprogramma het spel renju te laten spelen zal het kennis moeten hebben over enkele fundamentele begrippen uit het renjuspel. Deze begrippen worden hier besproken. Zij zijn in het programma zelf opgenomen en dienen als basis voor de afleidingen en kennisregels die buiten het programma worden opgeslagen.

Veld Voor ieder van de 225 velden op het speelbord is binnen het programma een constante gedefiniëerd. Een veld heeft een aantal eigenschappen waaronder een positie en een kleur. Wanneer een veld leeg is, heeft het de kleur *none*. Wanneer zich een steen op het veld bevindt, heeft het de kleur *white* of *black*, afhankelijk van de kleur van de steen. De constanten binnen het programma kunnen niet als zodanig worden aangeroepen. Zij worden toegekend aan de waarde van een variabele. Deze variabele krijgt een naam die opgebouwd is aan de hand van de tekst *Field*. Mogelijke namen voor variabelen zijn onder andere *Field*, *Field1* en *Field_2*.

Groep Bepaalde verzamelingen van velden worden gecombineerd tot een groep. Primair bestaat een groep uit vijf basisvelden. Deze basisvelden zijn vijf aangrenzende velden op een horizontale, verticale of diagonale[2] lijn. Het aantal vijf is gekozen omdat dit precies het aantal stenen op een rij is dat nodig is om het spel te winnen. Los van deze basisvelden worden, wanneer deze bestaan, ook de twee velden die in de lengterichting aan de groep grenzen tot de groep gerekend. Dit zijn de zogenaamde randvelden. Het nut van deze randvelden is onder andere om na te gaan of Zwart een overline maakt. Figuur 3.3 geeft een voorbeeld van een groep met enkel de vijf basisvelden (1), een groep met slechts één randveld (2) en een groep met twee randvelden (3).



Figuur 3.3: Voorbeeld van verschillende soorten groepen.

In tabel 3.2 worden de aantallen groepen getoond van de drie typen zoals ze in figuur 3.3 met de cijfers 1, 2 en 3 zijn aangeduid. Uit tabel 3.2 blijkt dat verreweg de meeste groepen twee randvelden bevatten.

	type 1	type 2	type 3	totaal
horizontaal	0	30	135	165
verticaal	0	30	135	165
diagonaal	4	76	162	242
totaal	4	136	432	572

Tabel 3.2: Aantallen groepen van drie verschillende typen.

Net als velden hebben ook groepen een kleur. Deze kleur wordt bepaald aan de hand van de stenen die zich op de vijf basisvelden bevinden. Een waarde *none* wil zeggen dat de vijf basisvelden van de groep alle leeg zijn. De waarden *white* en *black* geven aan dat zich enkel witte, respectievelijk enkel zwarte stenen in de basisvelden van de groep bevinden. Een kleurwaarde *both* geeft aan dat de basisvelden van de groep zowel witte als zwarte stenen bevatten.

Om iets te kunnen zeggen over de bezettingsgraad van een groep wordt het begrip *cardinaliteit* gebruikt. De cardinaliteit van een groep is nul wanneer de kleur van

[2]Voor ieder van deze richtingen is in het programma een constante gedefinieerd.

de groep *none* of *both* is. Wanneer de kleur van de groep *white* of *black* is, geeft de cardinaliteit aan hoeveel van de vijf basisvelden bezet zijn. Een groep met kleur *white* en cardinaliteit 3 wordt ook wel een witte drie-groep genoemd. Voor andere kleuren en cardinaliteiten wordt een soortgelijke naamgeving gehanteerd.

Groep-variabelen krijgen namen die opgebouwd zijn uit het woord *Group*. Mogelijke namen voor deze variabelen zijn onder andere *Group*, *Group_1* en *Group4*.

3.3 De predikaten

Er is een aantal predikaten ontwikkeld waarmee onderzocht kan worden of er velden en groepen bestaan die aan een bepaalde voorwaarde voldoen. Aan de hand van deze predikaten worden verderop afleidingen en kennisregels geformuleerd. De predikaten die gedefinieerd zijn, hebben op één na een vast aantal argumenten. Dit aantal wordt middels een getal na een schuine streep vermeld na de predikaatnaam. Het zal later blijken dat het mogelijk is via afleidingen predikaten te definiëren met dezelfde naam als hier vermeld, maar met een ander aantal argumenten. In het programma zijn semantische regels opgenomen die nagaan of de argumenten die worden meegegeven van het goede type zijn. Wanneer dit niet zo is, zal het predikaat niet toegepast kunnen worden. De volgende zeven predikaten zijn ontwikkeld:

group/3 Het predikaat *group* heeft drie argumenten. Als eerste een groep, als tweede een kleur en als derde een cardinaliteit. Het predikaat geeft aan of de groep de gevraagde kleur en cardinaliteit heeft. Een voorbeeld van het gebruik van dit predikaat is *group(Group1, black, 5)*.

in/4 Het predikaat *in* heeft vier argumenten. Als eerste een veld, als tweede een groep en als derde en vierde twee getallen die als i_1 en i_2 aangeduid worden. Het predikaat geeft aan of het veld op één van de posities tussen i_1 en i_2 van de groep ligt. De basisvelden hebben hierbij posities 1 tot en met 5. Eventuele randvelden bevinden zich op de posities 0 en/of 6. Een voorbeeld hiervan is *in(Field, Group1, 1, 1)*, dat aangeeft dat *Field* zich op het eerste basisveld bevindt van *Group1*.

color/2 Dit predikaat gaat na of het veld van het eerste argument de kleur van het tweede argument heeft. Als kleurwaarden kunnen enkel de constanten *none*, *white* en *black* gebruikt worden.

direction/2 Aan de hand van dit predikaat kan bepaald worden welke richting een bepaalde groep heeft. Het wordt gebruikt met een groep als eerste argument en een richting als tweede argument. Deze richting is een van de constanten zoals ze in het programma zijn opgenomen. De waarden van deze constanten zijn niet van belang omdat dit predikaat enkel gebruikt zal worden in combinatie met het hierna te definiëren predikaat *equal*.

equal/2 Dit predikaat geeft aan of de twee argumenten die het meekrijgt aan elkaar gelijk zijn. De argumenten kunnen zowel velden, groepen als bijvoorbeeld richtingen of cardinaliteiten zijn.

last.move/1 Dit predikaat geeft aan of het veld dat als argument wordt meegegeven het veld is waarop de laatste zet is gedaan.

lead.to.black.loss Dit predikaat heeft als enige geen voorgedefinieerd aantal argumenten. De argumenten die meegegeven worden, moeten lege velden zijn. Er wordt gesimuleerd dat op ieder van deze velden een zwarte steen geplaatst wordt in de volgorde van de argumenten. Na ieder van deze zetten wordt bekeken of zich een verliessituatie voordoet voor Zwart door een overline, dubbel-vier of verboden dubbel-drie. Wanneer dit ergens in het proces voorkomt, zal worden geconcludeerd dat de zetten van de argumenten inderdaad tot verlies voor Zwart leiden. Wanneer alle stenen geplaatst kunnen worden zonder een verliessituatie te creëren, wordt dit uiteraard niet gedaan. In ieder geval worden de gesimuleerde zwarte stenen van het speelbord verwijderd nadat dit predikaat behandeld is.

Een voorbeeld van het gebruik van dit predikaat is de bewering *lead.to.black.loss (Field1, Field2, Field3)*, dat nagaat of het doen van zetten voor Zwart op achtereenvolgens *Field1*, *Field2* en *Field3* leidt tot een niet toegestane situatie en daarmee tot verlies voor Zwart. De werking van dit predikaat is als volgt: Het programma simuleert een zwarte steen op *Field1*. Wanneer deze steen leidt tot een niet toegestane situatie, is de bewering waar. In dit geval wordt de steen op *Field1* weggehaald om terug te keren naar de oorspronkelijke positie en wordt niet verder gekeken naar *Field2* en *Field3*. Wanneer de gesimuleerde zet niet tot een niet toegestane situatie leidt, wordt een zwarte steen op *Field2* gesimuleerd. Indien deze zet tot een niet toegestane situatie leidt, worden de stenen op *Field1* en *Field2* weggehaald en is de bewering waar. Als dit niet het geval is wordt een zwarte steen gesimuleerd op *Field3* en wordt wederom bekeken of de ontstane situatie toegelaten is. Als dit niet het geval is, is de bewering waar en worden de stenen op *Field1*, *Field2* en *Field3* verwijderd. Als dit wel het geval is, is de bewering onwaar. Ook nu worden de drie gesimuleerde stenen verwijderd.

3.4 De afleidingen

Met behulp van de besproken predikaten zouden direct kennisregels opgesteld kunnen worden. Er wordt echter nog een extra stap gemaakt, namelijk het definiëren van afleidingen. Met behulp van de afleidingen worden nieuwe predikaten gedefinieerd, die slechts waar zijn indien andere predikaten waar zijn. Een afleiding bestaat uit twee onderdelen. Het gedeelte voor het *denotes*-teken wordt de *premisse* genoemd en bevat de naam van het nieuwe predikaat met de bijbehorende argumenten. Het gedeelte achter het *denotes*-teken heet de *body* en bevat één of meer literals die moeten gelden om de premisse waar te maken. Deze

literals kunnen de eerder besproken predikaten bevatten, maar ook predikaten die op hun beurt weer de premisse zijn van een afleiding.

Groepen voor Wit Omdat in de kennisregels erg veel gebruik gemaakt wordt van bijvoorbeeld witte vier-groepen zijn enkele predikaten gedefiniëerd die een verkorte notatievorm bieden. Deze predikaten gaan uit van het *group*-predikaat.

```
white.one(Group)   :- group(Group, white, 1)
white.two(Group)   :- group(Group, white, 2)
white.three(Group) :- group(Group, white, 3)
white.four(Group)  :- group(Group, white, 4)
white.five(Group)  :- group(Group, white, 5)
```

Groepen voor Zwart Ook voor Zwart zijn predikaten gedefinieerd die het *group*-predikaat vervangen. Er is hier toegevoegd dat zwarte groepen geen zwarte stenen in de randvelden mogen hebben. Dit wordt nagegaan door het predikaat *black.border*.

```
black.border(Group) :- in(Field_, Group, 0, 0), color(Field_, black)
black.border(Group) :- in(Field_, Group, 6, 6), color(Field_, black)
```

Er is één soort zwarte groepen dat wel zwarte stenen in de randvelden bevat, namelijk de zwarte overline. Voor dit begrip is een apart predikaat gedefinieerd.

```
black.one(Group)     :- group(Group, black, 1), ¬black.border(Group)
black.two(Group)     :- group(Group, black, 2), ¬black.border(Group)
black.three(Group)   :- group(Group, black, 3), ¬black.border(Group)
black.four(Group)    :- group(Group, black, 4), ¬black.border(Group)
black.five(Group)    :- group(Group, black, 5), ¬black.border(Group)
black.overline(Group):- group(Group, black, 5),  black.border(Group)
```

Open groepen Deze predikaten definiëren open groepen aan de hand van de hierboven beschreven predikaten en het predikaat *open*. Het predikaat *open* maakt op zijn beurt weer gebruik van het predikaat *empty*. Deze predikaten worden hier eerst besproken.

```
empty(Field) :- color(Field, none)
open(Group)  :- in(Field_1, Group, 0, 0), empty(Field_1),
                in(Field_2, Group, 5, 5), empty(Field_2)
open(Group)  :- in(Field_1, Group, 1, 1), empty(Field_1),
                in(Field_2, Group, 6, 6), empty(Field_2)
```

Het *empty*-predikaat geeft aan dat een veld leeg is als het kleur *none* heeft. Het *open*-predikaat stelt dat een groep open is wanneer een buitenste basisveld en het tegenoverliggende randveld leeg zijn. De predikaten die witte open groepen definiëren zien er als volgt uit:

```
white.open.one(Group)   :- white.one(Group),   open(Group)
white.open.two(Group)   :- white.two(Group),   open(Group)
```

```
white.open.three(Group) :- white.three(Group), open(Group)
white.open.four(Group)  :- white.four(Group),  open(Group)
```

De predikaten voor de zwarte open groepen zien er vergelijkbaar uit:

```
black.open.one(Group)   :- black.one(Group),   open(Group)
black.open.two(Group)   :- black.two(Group),   open(Group)
black.open.three(Group) :- black.three(Group), open(Group)
black.open.four(Group)  :- black.four(Group),  open(Group)
```

Het predikaat voor een open vijf-groep is niet gedefinieerd omdat deze niet kan bestaan. Omdat alle basisvelden van een vijf-groep gevuld zijn, kan het predikaat *open* nooit gelden.

Herdefinitie van het *in*-predikaat Het predikaat *in* is met vier argumenten erg omslachtig. Daarom zijn er herdefinities gemaakt voor de meest gebruikte varianten. Het predikaat *in* met twee argumenten kijkt of een veld binnen één van de basisvelden van een groep valt. Het predikaat *extended.in* doet hetzelfde maar neemt ook de randvelden mee. Het derde predikaat *centered.in* kijkt of een veld voorkomt in de middelste drie velden van een groep

```
in(Field, Group)          :- in(Field, Group, 1, 5)
extended.in(Field, Group) :- in(Field, Group, 0, 6)
centered.in(Field, Group) :- in(Field, Group, 2, 4)
```

Overlap predikaten Bij de kennisregels worden situaties behandeld waarin moet worden nagegaan of twee groepen geen overlappend veld hebben dat anders is dan een reeds bekend veld. Om dit te onderzoeken worden hier drie predikaten gedefinieerd.

Het eerste is het *overlap/3*-predikaat. Dit predikaat stelt dat twee verschillende groepen een overlappend veld hebben, anders dan een meegegeven veld, als er zich een leeg veld in beide groepen bevindt, dat verschilt van het meegegeven veld. De afleiding hiervoor luidt als volgt:

```
overlap(Group1, Group2, Field) :-
        ¬equal(Group1, Group2), extended.in(Field_, Group1),
        in(Field_, Group2), empty(Field_), ¬equal(Field, Field_)
```

Het tweede is het *overlap/4*-predikaat. Dit predikaat gaat na of een groep een overlap heeft, anders dan het meegegeven veld, met een willekeurige andere groep die een bepaalde kleur en cardinaliteit heeft.

```
overlap(Group, Color, N, Field) :-
        group(Group_, Color, N), overlap(Group, Group_, Field)
```

Het *overlap/6*-predikaat is een uitbreiding van het *overlap/4*-predikaat en kijkt of twee groepen beide een overlap hebben met een groep met een bepaalde kleur en

cardinaliteit. Deze overlappende velden moeten hierbij verschillend zijn van de velden die als laatste twee argumenten worden meegegeven.

```
overlap(Group1, Group2, Color, N, Field1, Field2) :-
        group(Group_, Color, N), overlap(Group1, Group_, Field1),
                                 overlap(Group2, Group_, Field2)
```

Speciale zwarte drie-groep Om na te gaan of Zwart een verboden dubbel-drie maakt is het noodzakelijk te kijken of er twee zwarte drie-groepen bestaan die beide, zonder dat hierbij een zwarte vijf-groep ontstaat, met één steen uit te breiden zijn tot zwarte open vier-groepen. Een dergelijke zwarte drie-groep is als volgt gedefiniëerd:

```
black.special.three(Group) :- black.three(Group),
                    in(Field_1, Group, 2, 5), empty(Field_1),
                    in(Field_2, Group, 1, 1), empty(Field_2),
                    in(Field_3, Group, 6, 6), empty(Field_3),
                    ¬overlap(Group, black, 4, Field_2),
                    ¬lead.to.black.loss(Field_1, Field_2),
                    ¬lead.to.black.loss(Field_1, Field_3)
black.special.three(Group) :- black.three(Group),
                    in(Field_1, Group, 1, 4), empty(Field_1),
                    in(Field_2, Group, 0, 0), empty(Field_2),
                    in(Field_3, Group, 5, 5), empty(Field_3),
                    ¬overlap(Group, black, 4, Field_3),
                    ¬lead.to.black.loss(Field_1, Field_2),
                    ¬lead.to.black.loss(Field_1, Field_3)
```

Extra richtingspredikaat In navolging van het *direction*-predikaat dat de richting van één groep aangeeft, is dit predikaat gedefiniëerd dat nagaat of twee groepen dezelfde richting hebben.

```
same.direction(Group1, Group2) :- direction(Group1, Direction_1),
                                  direction(Group2, Direction_2),
                                  equal(Direction_1, Direction_2)
```

3.5 De kennisregels

Alle bouwstenen voor de constructie van kennisregels zijn nu besproken. Zoals eerder is genoemd, dienen kennisregels om tijdig gewonnen situaties te herkennen. De regels zijn hiervoor ingedeeld in verschillende niveau's. Per niveau worden kennisregels verzameld die winst opleveren binnen een bepaald aantal zetten. Ook zijn er kennisregels op niveau 0 gedefinieerd die directe winst beschrijven. Het hoogste niveau waarop in dit onderzoek kennisregels beschreven zijn, is niveau 7. Het is mogelijk kennisregels te ontwikkelen op niveau 9 en hoger. Los van het aantal velden op het bord bestaat er geen bovengrens voor

het niveau waarop kennisregels opgesteld kunnen worden[3]. Om het spel renju op te lossen volstaat een serie kennisregels die vanuit een leeg speelbord leiden tot een winstpositie. De kennisregels zijn in types ingedeeld om de overzichtelijkheid te verhogen. Per type bevinden zich één of meerdere kennisregels. In tabel 3.5 zijn de ontwikkelde typen kennisregels weergegeven op de verschillende niveaus.

Niveau 0	Directe winst
Niveau 1	Directe dreiging
Niveau 3	T-dreiging
	Renju-dreiging
Niveau 5	Z-dreiging
	Kruis van type 1
	Kruis van type 2
Niveau 7	Dubbel kruis van type 1
	Dubbel kruis van type 2
	Dubbel kruis van type 3

Tabel 3.3: Overzicht van de ontwikkelde typen kennisregels.

De meeste typen kennisregels zijn overgenomen uit Uiterwijk's onderzoek naar go-moku [10]. Vele regels die voor go-moku gelden, gelden namelijk ook voor renju. In tegenstelling tot go-moku moeten bij renju aparte regels opgesteld worden voor Zwart en Wit. De reden hiervoor wordt gevormd door de regels die Zwart belemmeren in zijn spel zoals het feit dat Zwart verliest door een overline. Los van Uiterwijk's regels zijn zowel extra typen als ook extra kennisregels binnen bestaande typen ontwikkeld die specifiek voor renju gelden.

De kennisregels worden per niveau en per type behandeld en worden vaak verduidelijkt aan de hand van voorbeelden.

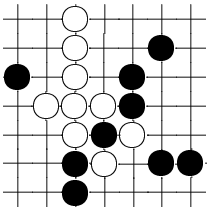
Kennisregels op niveau 0

Kennisregels op niveau 0 beschrijven situaties waarbij de winst reeds behaald is door één van beide spelers. Deze regels fungeren hiermee als stopcriteria voor het spel. Door het aanpassen van deze regels kan de essentie van het spel gewijzigd worden. De kennisregels op niveau 0 vallen onder één type en beschrijven zowel winsten door middel van vijf-groepen als winsten door middel van niet toegestane situaties voor Zwart.

Directe winst Een situatie levert directe winst op wanneer zich ergens op het bord een vijf-groep bevindt. De definitie van een vijf-groep is verschillend voor beide spelers.

[3]Een bestaande kennisregel op niveau n resulteert in een kennisregel op niveau $n+2$ door de premisse aan te vullen met een drie-groep voor de speler aan zet. Het uitbreiden van deze drie-groep dwingt de tegenstander tot verdediging. Hierna is de originele premisse van toepassing. Dit verhoogt het niveau van de originele kennisregel met 2.

Voor Zwart is een groep slechts een vijf-groep wanneer zich in de randvelden van de groep geen zwarte stenen bevinden. Voor Wit is een vijf-groep iedere groep waarin zich vijf witte stenen op de basisvelden bevinden. Het is voor Wit niet van belang welke stenen zich op de randvelden van een dergelijke groep bevinden. Deze definities zijn reeds in de eerder vermelde predikaten *black.five* en *white.five* opgenomen. In figuur 3.4 wordt een situatie getoond die gewonnen is voor Wit.



Figuur 3.4: Voorbeeld van directe winst voor Wit.

Door de hiervoor genoemde predikaten zijn de kennisregels voor directe winst door een vijf-groep erg eenvoudig weer te geven. De premisse van de kennisregel voor Wit luidt:

```
white.five(Group1)
```

Deze premisse geldt indien een groep gevonden kan worden die voldoet aan het predikaat *white.five*. Wanneer een dergelijke groep gevonden kan worden, krijgt de variabele *Group1* de waarde van deze groep.

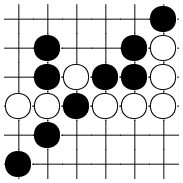
De premisse van de kennisregel voor Zwart is vergelijkbaar:

```
black.five(Group1)
```

Indien één van deze premissen van toepassing is, kan geconcludeerd worden dat de partij gewonnen is door één van beide spelers.

Los van de winst door een vijf-groep kan Wit ook winnen in typische renju situaties waarin Zwart een verboden situatie creëert. Er zijn drie van deze situaties mogelijk. De eerste hiervan is wanneer Zwart een overline creëert. Een overline is een situatie met meer dan vijf zwarte stenen op een rij. Een voorbeeld van directe winst door een zwarte overline staat afgebeeld in figuur 3.5.

Zwart zal in een dergelijke situatie de partij zelf uit handen gegeven hebben omdat hij degene is die de zet gedaan heeft die leidde tot de overline. Een situatie zoals de hier afgebeelde zal dan ook niet voorkomen zolang Zwart geen blunders begaat. Wanneer Zwart gedwongen wordt te kiezen tussen het verdedigen van een witte vier-groep ten koste van een eigen overline of het niet verdedigen van deze vier-groep in de hoop dat Wit zijn dreiging niet zal benutten, zal hij altijd voor de tweede optie kiezen. Wanneer zich een situatie voordoet waarbij door het doen van een zet zowel



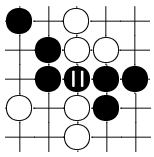
Figuur 3.5: Voorbeeld van directe winst voor Wit door een zwarte overline.

een zwarte vijf-groep als een zwarte overline ontstaat, krijgt de vijf-groep voorrang waardoor Zwart de partij wint. Bij het toepassen van deze kennisregel moet dan ook worden nagegaan of met het creëren van de overline niet tevens een vijf-groep ontstaat.

De premisse van de kennisregel luidt als volgt:

```
¬black.five(Group_), black.overline(Group1)
```

Een tweede situatie die verboden is voor Zwart, is de zogenaamde dubbel-vier. Onder een dubbel-vier wordt verstaan de situatie volgend op een zet waardoor uit twee drie-groepen twee vier-groepen ontstaan. Een voorbeeld van een dergelijke situatie staat afgebeeld in figuur 3.6, waarbij het laatst gespeelde veld gemarkeerd is. De vier-groepen zullen groepen moeten zijn zonder zwarte randvelden omdat er anders geen dreiging van Zwart uitgaat. Deze conditie is wederom reeds in de gebruikte predikaten opgenomen.

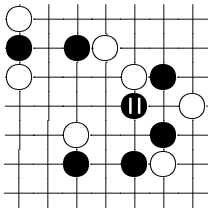


Figuur 3.6: Voorbeeld van directe winst voor Wit door een zwarte dubbel-vier.

Om deze situatie in een kennisregel te vangen moet worden nagegaan of er twee zwarte vier-groepen met verschillende richtingen bestaan. Deze twee groepen dienen als gemeenschappelijk veld het laatstgespeelde veld te hebben. De regel dat Zwart verliest wanneer hij een dubbel-vier maakt, gaat niet op wanneer hij tegelijkertijd een vijf-groep vormt.

De premisse van de kennisregel die is opgesteld luidt als volgt:

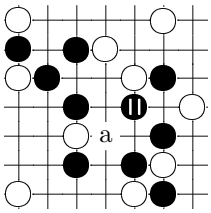
```
¬black.five(Group_),
black.four(Group1), black.four(Group2),
¬same.direction(Group1, Group2),
in(Field1, Group1), in(Field1, Group2), last.move(Field1)
```



Figuur 3.7: Voorbeeld van directe winst voor Wit door een verboden zwarte dubbel-drie.

De derde en laatste situatie waarin Zwart door een eigen zet de partij uit handen kan geven is wanneer hij een verboden dubbel-drie maakt. Dit komt voor wanneer hij twee drie-groepen creëert die samenkomen op het punt van de laatste zet. Een voorbeeld van een verboden dubbel-drie staat in figuur 3.7. In deze figuur verliest Zwart de partij nadat hij het gemarkeerde veld heeft gespeeld. Hierdoor ontstaan twee diagonale drie-groepen die beide uitbreidbaar zijn: een verboden dubbel-drie.

In figuur 3.8 is de figuur uitgebreid tot een situatie die wel is toegestaan. Het spelen van veld *a* heeft hier namelijk een overline tot gevolg. Daardoor valt de dreiging van één van beide drie-groepen weg en is de situatie niet langer verboden.



Figuur 3.8: Voorbeeld van een toegestane zwarte dubbel-drie.

Ook hier is de kennisregel niet van toepassing wanneer Zwart gelijk met een eventueel verboden dubbel-drie een vijf-groep creëert. Om na te gaan of een drie-groep in een dubbel-drie dreigend is, wordt gebruik gemaakt van het predikaat *black.special.three*, dat eerder bij de afleidingen is gedefinieerd.

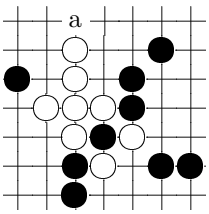
De premisse van de kennisregel die de verboden dubbel-drie beschrijft, luidt:

```
¬black.five(Group_),
black.three(Group1), black.three(Group2),
¬same.direction(Group1, Group2),
in(Field1, Group1), in(Field1, Group2), last.move(Field1),
black.special.three(Group1),
black.special.three(Group2)
```

Kennisregels op niveau 1

Kennisregels op niveau 1 beschrijven situaties die met één zet uit te breiden zijn tot een winstsituatie. Ook hier is slechts één type kennisregels. Vanaf dit niveau wordt bij de kennisregels naast de premisse een zet gegeven die gedaan moet worden om de winst te bereiken. Deze zet wordt bij de beschrijving van de kennisregel genoemd.

Directe dreiging Een speler kan de winst naar zich toetrekken wanneer zich in zijn beurt een vier-groep op het bord bevindt met stenen van zijn eigen kleur. Een dergelijke groep kan namelijk met één zet uitgebreid worden tot een vijf-groep die winst oplevert. De winst bij dit type kennisregels kan behaald worden door het laatste lege basisveld van de groep te spelen. In figuur 3.9 staat een dergelijke situatie waarin Wit kan winnen door het veld *a* te spelen.



Figuur 3.9: Voorbeeld van een directe dreiging door Wit.

Om de hier beschreven situatie te beschrijven zou een premisse in de vorm van *black.four(Group1)* of *white.four(Group1)* voldoende zijn. Om echter te bepalen welk veld gespeeld moet worden, zijn de premissen uitgebreid.

De volledige premisse van de kennisregel voor Wit is als volgt:

```
white.four(Group1), in(Field1, Group1), empty(Field1)
```

De premisse van de kennisregel voor Zwart luidt:

```
black.four(Group1), in(Field1, Group1), empty(Field1)
```

Wanneer de premisse van toepassing is, zal de waarde van de variabele *Field1* het veld zijn dat gespeeld moet worden om tot winst te komen.

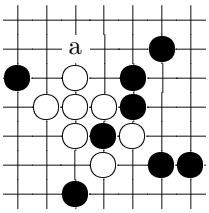
Veel van de kennisregels die volgen zullen eisen dat de tegenstander van de speler aan zet geen vier-groepen heeft. De methode die gebruikt wordt om de kennisregels te doorzoeken zal in §4.1 besproken worden. Door de gebruikte methode zal door de hier besproken kennisregel gestopt worden met het doorzoeken van de kennisregels wanneer een van beide spelers een vier-groep heeft. Dit is de reden dat bij de komende regels niet opgenomen hoeft te worden dat de tegenstander geen vier-groepen mag hebben.

Kennisregels op niveau 3

Kennisregels op niveau 3 beschrijven situaties die binnen 3 zetten tot gegarandeerde winst leiden. Dergelijke regels komen voor wanneer de speler aan zet na een goede eerste zet de winst na een derde zet naar zich toe kan trekken, ongeacht de tweede zet van de tegenstander. Kennisregels van niveau 3 beschrijven in welke situaties dit voorkomt en wat in dergelijke situaties een goede eerste zet is. Er zijn twee typen kennisregels onderscheiden op dit niveau, namelijk de T-dreiging en de renju-dreiging.

T-dreiging Dit type kennisregels beschrijft een situatie die winst in drie zetten oplevert. De situatie die hier besproken wordt, komt voor wanneer een speler twee drie-groepen heeft die met één steen beide kunnen worden uitgebreid tot vier-groepen. Hiervoor is los van twee drie-groepen een leeg veld nodig dat deel uitmaakt van beide drie-groepen. Door het spelen van dit lege veld ontstaan twee vier-groepen, waarvan de tegenstander er slechts één kan verdedigen. De overgebleven vier-groep kan vervolgens uitgebreid worden tot een gewonnen situatie. De naamgeving van dit type kennisregels is ontstaan uit een situatie die dit type beschrijft waarbij de beide drie-groepen haaks op elkaar staan en samen met het te spelen veld de letter T vormen.

Een speciaal geval van de T-dreiging is de open drie-groep. In dit geval hebben de twee benodigde drie-groepen dezelfde richting. Een open drie-groep is niets anders dan een drie-groep met minstens één leeg randveld. In figuur 3.10 is een situatie afgebeeld met een open drie-groep. Wanneer Wit het veld *a* speelt, ontstaat een open vier-groep. Zwart kan deze groep slechts aan één zijde verdedigen. Het andere uiteinde kan door Wit gespeeld worden voor de winst.



Figuur 3.10: Voorbeeld van een T-dreiging voor Wit.

De premisse van de kennisregel die de T-dreiging voor Wit beschrijft is:

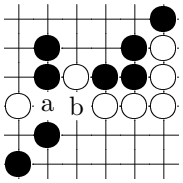
```
white.three(Group1), white.three(Group2), ¬equal(Group1, Group2),
in(Field1, Group1), in(Field1, Group2), empty(Field1)
```

Omdat Zwart geen dubbel-vier mag maken, leiden bepaalde T-dreigingen voor Zwart niet tot winst. Om na te gaan welke dreigingen gelden, wordt nagegaan of het spelen van het overlappende veld leidt tot een situatie die verlies voor Zwart inleidt. Indien dit niet het geval is, is de dreiging geldig en kan geconcludeerd worden dat de kennisregel van toepassing is. De premisse van de kennisregel voor Zwart verschilt hierdoor van die voor Wit en luidt als volgt:

```
black.three(Group1), black.three(Group2), ¬equal(Group1, Group2),
in(Field1, Group1), in(Field1, Group2), empty(Field1),
¬lead.to.black.loss(Field1)
```

In beide situaties moet het veld *Field1* gespeeld worden om tot de winst te komen.

Renju-dreiging Deze kennisregel is een typische renju-regel en geldt alleen voor Wit. Zij maakt gebruik van het feit dat bepaalde situaties niet zijn toegestaan voor Zwart. In de situatie die hier beschreven wordt, heeft Wit een drie-groep die hij uit kan breiden, maar die Zwart vervolgens niet kan verdedigen omdat hij er een niet toegestane situatie mee zou creëren. Een dergelijke situatie staat afgebeeld in figuur 3.11. Wanneer Wit in deze situatie het veld *a* speelt, zou Zwart moeten verdedigen via veld *b*. Omdat dit tot een overline leidt, zal hij dit niet doen. Hierna is voor Wit de weg vrij om zelf veld *b* te spelen en de partij te winnen.



Figuur 3.11: Voorbeeld van een renju-dreiging.

De premisse van de kennisregel voor deze situatie luidt:

```
white.three(Group1),
in(Field1, Group1), empty(Field1),
in(Field2, Group1), ¬equal(Field1, Field2), empty(Field2),
lead.to.black.loss(Field2)
```

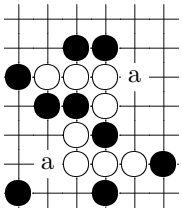
Het veld dat gespeeld moet worden, is *Field1*. Omdat Zwart *Field2* niet kan verdedigen, kan Wit in de derde beurt de winst naar zich toetrekken door wel *Field2* te spelen.

Kennisregels op niveau 5

Kennisregels op niveau 5 beschrijven situaties die binnen vijf zetten omgezet kunnen worden in een winstsituatie voor de speler aan zet, ongeacht de zetten van de tegenstander. Op dit niveau zijn drie typen kennisregels onderscheiden, namelijk de Z-dreiging, het kruis van type 1 en het kruis van type 2.

Z-dreiging Een manier om in vijf zetten te winnen is wanneer zich een Z-dreiging voordoet op het bord. Uit de witte stenen en de velden gemarkeerd met *a* in figuur 3.12 blijkt de naamgeving van dit type kennisregels. Om aan deze regel te voldoen moeten zich twee drie-groepen en een twee-groep op het bord bevinden. De twee-groep moet

daarbij op twee verschillende lege velden ieder een overlap maken met één van de drie-groepen. Het is hierbij niet toegestaan dat de tegenstander reeds een drie-groep heeft.



Figuur 3.12: Voorbeeld van een z-dreiging voor Wit.

De premisse van de kennisregel voor Wit is als volgt:

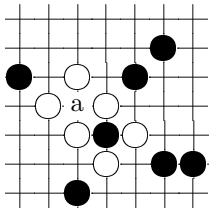
```
white.three(Group1), white.three(Group2),
¬equal(Group1, Group2), white.two(Group3),
in(Field1, Group1), empty(Field1), in(Field1, Group3),
in(Field2, Group2), ¬equal(Field1, Field2),
empty(Field2), in(Field2, Group3),
¬overlap(Group1, black, 3, Field1),
¬overlap(Group2, black, 3, Field2)
```

Het veld dat gespeeld moet worden, is de waarde die de variabele *Field1* heeft op het moment dat de kennisregel van toepassing is.

Een z-dreiging voor Zwart komt nooit voor, omdat bij het spelen van het tweede *a*-veld een dubbel-vier ontstaat. Aangezien een dergelijke situatie tot verlies leidt voor Zwart, leidt dit type kennisregels niet tot winst voor Zwart. Om deze reden is er geen kennisregel die een z-dreiging voor Zwart beschrijft.

Kruis van type 1 Dit is het tweede type kennisregels dat winst in vijf zetten beschrijft. Hij stelt dat een speler wint indien zich een bepaalde combinatie van twee open twee-groepen op het bord voordoet. Wanneer deze twee groepen verschillende richtingen hebben en ze een leeg overlappend veld binnen één van de drie middelste velden hebben, kan de speler aan zet winst behalen door het lege overlappende veld te spelen. Hierdoor ontstaan twee open-drie groepen waarvan er hoogstens één te verdedigen is. De niet verdedigde open drie-groep kan vervolgens tot een winstsituatie worden uitgebreid. Voor Zwart moet bekeken worden of de ontstane dubbel-drie toegestaan is. Dit gebeurt aan de hand van het *lead.to.black.loss*-predikaat. Als extra voorwaarde geldt dat de tegenstander geen drie-groepen mag hebben. Een voorbeeld van een winnende situatie die herkend wordt als kruis van type 1 voor Wit staat afgebeeld in figuur 3.13.

De premisse van de kennisregel die de situatie voor Wit beschrijft is:



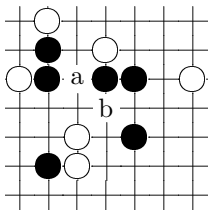
Figuur 3.13: Voorbeeld van een kruis van type 1 voor Wit.

```
¬black.three(Group_),
white.open.two(Group1), white.open.two(Group2),
¬same.direction(Group1, Group2),
centered.in(Field1, Group1), empty(Field1), centered.in(Field1, Group2)
```

Bij deze en alle nog volgende kennisregels moet het veld *Field1* gespeeld worden om tot een gewonnen situatie te komen.

Een vergelijkbare situatie voor Zwart zal niet tot een winstsituatie leiden. De reden hiervoor is dat het spelen van *Field1* leidt tot een dubbel-drie. Wanneer deze dubbel-drie verboden is, zal dit direct tot verlies voor Zwart leiden. Wanneer de dubbel-drie niet verboden is, zal hij niet tot winst leiden, juist omdat een dubbel-drie alleen toegestaan is wanneer deze niet tot winst leidt.

Kruis van type 2 Deze derde en laatste kennisregel van diepte vijf beschrijft winst door een combinatie van een drie-groep en een open twee-groep. De voorwaarde die moet gelden, is dat deze groepen verschillende richtingen en een overlappend leeg veld moeten hebben. Het lege veld moet zich op één van de middelste drie velden van de open twee-groep bevinden. Hier geldt niet de voorwaarde dat de tegenstander geen drie-groepen mag hebben omdat deze constant directe dreigingen moet verdedigen en daardoor zelf nooit een dreiging kan uitbreiden. Een situatie die aan deze voorwaarden voldoet, staat in figuur 3.14. Zwart kan hier de partij winnen door het met *a* gemarkeerde veld te spelen. Wit moet de ontstane vier-groep verdedigen. Zwart speelt vervolgens het veld *b* en creëert een open vier-groep. Wit kan deze groep slechts aan één kant verdedigen, waarna Zwart de andere zijde kan spelen voor de partij.



Figuur 3.14: Voorbeeld van een kruis van type 2 voor Zwart.

De premisse van de kennisregel voor Wit is:

```
white.open.two(Group1), white.three(Group2),
¬same.direction(Group1, Group2),
centered.in(Field1, Group1), empty(Field1), in(Field1, Group2),
¬overlap(Group2, black, 3, Field1)
```

De bijbehorende premisse voor Zwart luidt als volgt:

```
black.open.two(Group1), black.three(Group2),
¬same.direction(Group1, Group2),
centered.in(Field1, Group1), empty(Field1), in(Field1, Group2),
¬overlap(Group2, white, 3, Field1),
in(Field2, Group1), empty(Field2), ¬equal(Field1, Field2)
¬lead.to.black.loss(Field1, Field2)
```

Kennisregels op niveau 7

De kennisregels op niveau 7 zijn de laatste kennisregels die besproken worden. Op dit niveau zijn drie typen kennisregels onderscheiden. Deze typen vallen alle in de categorie van het dubbel kruis.

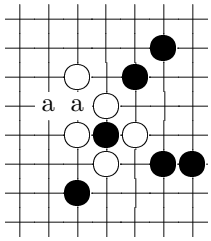
Dubbel kruis van type 1 Dit type kennisregels beschrijft een winstsituatie in zeven zetten. De situatie die hier beschreven wordt, is er één waarin de speler aan zet twee open twee-groepen zodanig uit kan breiden dat er een andere open drie-groep ontstaat. Zoals eerder is gezien, is een open drie-groep winst voor de speler aan zet.

Een gedetailleerdere blik toont dat er twee verschillende open twee-groepen nodig zijn. Tevens is een open één-groep vereist die een andere richting moet hebben dan de beide open twee-groepen. De middelste drie velden van de open één-groep dienen te bestaan uit een veld dat reeds gevuld is en twee lege velden die overlappen, ieder met één van de open twee-groepen. Het mag hierbij niet zo zijn dat de tegenstander met zijn eerste geforceerde zet beide open twee-groepen kan verdedigen of zelf een dreiging in de vorm van een drie-groep kan creëren.

Een voorbeeld van een dubbel kruis van type in 1 staat afgebeeld in figuur 3.15. De situatie in deze figuur kan gewonnen worden wanneer de speler aan zet de beide open twee-groepen uitbreidt door achtereenvolgens beide velden gemarkeerd met a te spelen. De tegenstander moet de ontstane open drie-groepen verdedigen om direct verlies te voorkomen. Wanneer beide open twee-groepen zijn uitgebreid en verdedigd, is een nieuwe open drie-groep ontstaan die eenvoudig in winst is om te zetten.

De premisse van de kennisregel voor Wit is als volgt:

```
¬black.three(Group_),
white.open.two(Group1), white.open.two(Group2), ¬equal(Group1, Group2),
white.open.one(Group3), ¬same.direction(Group1, Group3),
```



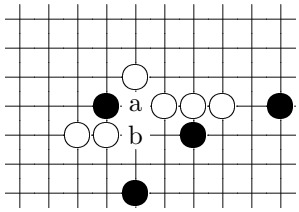
Figuur 3.15: Voorbeeld van een dubbel kruis van type 1 voor Wit.

```
                        ¬same.direction(Group2, Group3),
centered.in(Field1, Group1), empty(Field1), centered.in(Field1, Group3),
centered.in(Field2, Group2), ¬equal(Field1, Field2),
empty(Field2), centered.in(Field2, Group3),
¬overlap(Group1, Group2, Field1),
¬overlap(Group1, black, 2, Field1)
```

Een dubbel kruis van type 1 voor Zwart komt nooit voor omdat het doen van zetten op de velden *Field1* en *Field2* tot een dubbel-drie leidt. Ook hier geldt dat dit ofwel tot een verliessituatie leidt of tot een situatie die niet gegarandeerd tot winst leidt.

Dubbel kruis van type 2 Dit type kennisregels beschrijft een tweede winstsituatie in zeven zetten. Zij is van toepassing wanneer zich op het bord een bepaalde combinatie van een drie-groep, een open twee-groep en een open één-groep bevindt. De open één-groep moet hierbij een andere richting hebben dan de drie- en open twee-groep en met beide groepen een overlap hebben op verschillende velden binnen de middelste drie velden. Het overgebleven middenveld moet reeds gevuld zijn.

Indien deze situatie zich voordoet, kan de winst behaald worden door de drie-groep uit te breiden op het veld dat overlapt met de open één-groep. Nadat de tegenstander de ontstane vier-groep heeft moeten verdedigen, wordt het veld gespeeld dat de overlap vormt tussen de open twee-groep en de van oorsprong open één-groep. Met deze zet ontstaat een niet te stoppen combinatie van twee open drie-groepen. Om te voorkomen dat de tegenstander zelf dreigingen veroorzaakt is de extra voorwaarde opgenomen dat deze geen drie-groepen mag hebben. Een voorbeeld van een dergelijke situatie staat afgebeeld in figuur 3.16.



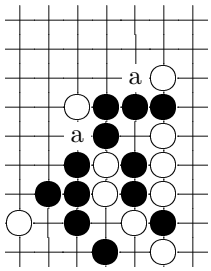
Figuur 3.16: Voorbeeld van een dubbel kruis van type 2 voor Wit.

De premissen van de kennisregels die zijn opgesteld om deze situatie te beschrijven staan hieronder afgebeeld. De premisse voor Wit is:

```
¬black.three(Group_),
white.three(Group1), white.open.two(Group2), white.open.one(Group3),
¬same.direction(Group1, Group3), ¬same.direction(Group2, Group3),
in(Field1, Group1), empty(Field1), centered.in(Field1, Group3),
centered.in(Field2, Group2), ¬equal(Field1, Field2),
empty(Field2), centered.in(Field2, Group3),
¬overlap(Group1, black, 2, Field1)
```

Ook deze situatie leidt voor Zwart niet tot gegarandeerde winst. Na het uitbreiden van de drie-groep zal de ontstane vier-groep verdedigd moeten worden. Met de tweede zwarte zet worden de bestaande open twee-groep en de zojuist gevormde open twee-groep beide uitgebreid waardoor een dubbel-drie ontstaat. Een dergelijke situatie zal niet tot gegarandeerde winst leiden.

Dubbel kruis van type 3 Dit type kennisregels beschrijft een situatie waarin twee verschillende drie-groepen zodanig uitgebreid kunnen worden dat een onstopbare open drie-groep ontstaat. Deze laatste groep kan ontstaan door een open één-groep die een andere richting heeft dan de beide drie-groepen. Voorwaarde hiervoor is dat van de middelste drie velden van deze groep één veld reeds gevuld is, en de andere twee een overlap hebben, ieder met één van de drie-groepen. Een voorbeeld hiervan staat vermeld in figuur 3.17. Door het uitbreiden van de twee verticale drie-groepen op de velden die gemarkeerd zijn met de letter *a*, ontstaat een verticale open drie-groep. Met deze groep kan winst behaald worden via de T-dreiging. De totale winstvariant heeft een lengte van zeven zetten.



Figuur 3.17: Voorbeeld van een dubbel kruis van type 3 voor Zwart.

Om te voorkomen dat de tegenstander met zijn eerste of tweede geforceerde zet zelf een dreiging creëert, zijn extra voorwaarden opgenomen. Deze stellen dat de tegenstander bij het verdedigen van de vier-groepen die ontstaan na het spelen van één of beide van de velden gemarkeerd met *a*, geen vier-groepen mag creëren. Deze voorwaarden worden ondervangen middels het *overlap*-predikaat.

De premisse van de kennisregel voor Wit luidt hiermee als volgt:

```
white.three(Group1), white.three(Group2), ¬equal(Group1, Group2),
white.open.one(Group3), ¬same.direction(Group1, Group3),
                        ¬same.direction(Group2, Group3),
in(Field1, Group1), empty(Field1), centered.in(Field1, Group3),
in(Field2, Group2), ¬equal(Field1, Field2),
empty(Field2), centered.in(Field2, Group3),
¬overlap(Group1, black, 3, Field1),
¬overlap(Group2, black, 3, Field2),
¬overlap(Group1, Group2, black, 3, Field1, Field2)
```

Voor Zwart moet worden nagegaan of het doen van één van de eerste drie zetten geen niet-toegestane situatie creëert. De premisse van de kennisregel voor Zwart is derhalve uitgebreider en luidt:

```
black.three(Group1), black.three(Group2), ¬equal(Group1, Group2),
black.open.one(Group3), ¬same.direction(Group1, Group3),
                        ¬same.direction(Group2, Group3),
in(Field1, Group1), empty(Field1), centered.in(Field1, Group3),
in(Field2, Group2), ¬equal(Field1, Field2),
empty(Field2), centered.in(Field2, Group3),
¬overlap(Group1, white, 3, Field1),
¬overlap(Group2, white, 3, Field2),
¬overlap(Group1, Group2, white, 3, Field1, Field2),
in(Field3, Group3), ¬equal(Field1, Field3), ¬equal(Field2, Field3),
empty(Field3), ¬lead.to.black.loss(Field1, Field2, Field3)
```

Hoofdstuk 4

Het programma

Dit hoofdstuk beschrijft hoe de besproken speltheorie en kenniselementen zijn toegepast binnen een computerprogramma. Dit computerprogramma heeft de naam NOIRE meegekregen en is ontwikkeld in de programmeertaal Delphi. Het is in staat het spel renju te spelen. Een dergelijk computerprogramma moet, gegeven een bepaalde spelsituatie, een goede zet kunnen bepalen. Hoe NOIRE dit probleem oplost wordt beschreven in §4.1. In §4.2 wordt beschreven hoe het programma de extern opgeslagen kennisregels interpreteert en na kan gaan of ze van toepassing zijn. Tenslotte wordt in §4.3 de interface tussen programma en gebruiker beschreven.

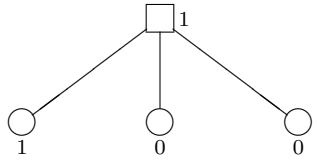
4.1 Het zoekproces

In deze sectie zal het zoekproces beschreven worden dat het programma doorloopt en dat leidt tot het doen van een zet. Allereerst wordt in een theoretische achtergrond beschreven hoe een zoekproces in het algemeen wordt aangepakt door een computerprogramma. Hierna zal meer gedetailleerd besproken worden hoe NOIRE het zoekproces tot uitvoering brengt. Dit zoekproces in opgesplitst in twee delen: eerst het tactische zoekproces, daarna het positionele. Kortweg dient het tactische zoekproces om te onderzoeken of vanuit de spelsituatie winst behaald kan worden. Het positionele zoekproces levert een zet wanneer het tactische zoekproces geen winst heeft kunnen vinden.

Theoretische achtergrond

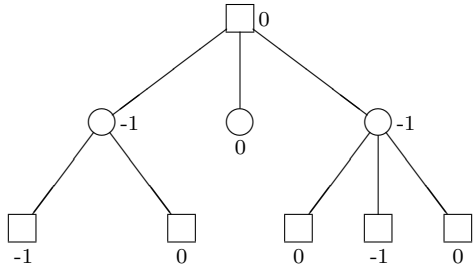
Om in een spel als renju een goede zet te kunnen doen is het noodzakelijk om vooruit te kijken. Dit vooruitkijken wordt hier besproken aan de hand van een variantenboom [4]. Een variantenboom is een grafische weergave van een spelsituatie met alle mogelijke vervolgsituaties die bereikt kunnen worden na het doen van één of meer zetten. In deze variantenbomen worden situaties waarin Zwart aan zet is weergegeven met een vierkant en de situaties waarin Wit aan zet is met een cirkel. Een situatie die gewonnen is door Zwart krijgt de waarde 1, een situatie die gewonnen is door Wit de waarde -1 en een remise krijgt

de waarde 0. Deze waarden worden onder de cirkels en vierkanten genoteerd. In figuur 4.1 wordt een eenvoudige variantenboom getoond waarbij voor Zwart vanuit de startsituatie drie mogelijke zetten bestaan. De eerste zet levert winst voor Zwart op, de tweede en derde leiden tot een remise. Wanneer de computer als zwarte speler in deze situatie een zet zou moeten bepalen, zou dit de eerste zet zijn omdat deze hem leidt tot winst, waar de andere twee zetten slechts remise opleveren. In het algemeen zal Zwart die zet kiezen die hem de grootst mogelijke waarde oplevert, in dit geval de eerste zet met een waarde 1 tegen de laatste twee zetten met waarde 0. Met andere woorden: Zwart kiest altijd de zet die hem de *maximale* waarde oplevert. De waarde van een vierkant wordt hiermee bepaald door het maximum van de cirkels die onder het vierkant liggen. Deze waarde is in dit voorbeeld 1 en wordt naast het vierkant weergegeven.



Figuur 4.1: Een eenvoudige variantenboom.

In een uitgebreidere variantenboom zoals in figuur 4.2 is het voor Zwart niet mogelijk meteen de beste zet te bepalen omdat niet alle varianten direct een waarde opleveren. In dergelijke gevallen moet eerst dieper in de variantenboom gekeken worden. Per variant voor Zwart wordt bekeken welke zetten Wit zou kunnen doen. Uit deze zetten kiest Wit de beste, dat wil zeggen die zet die hem de *minimale* waarde oplevert. Deze waarde gebruikt Zwart om op het hoogste niveau de beste zet te bepalen. In het voorbeeld zal Wit na de eerste variant van Zwart de eerste zet kiezen. Deze levert hem de waarde -1. Na de derde variant van Zwart zal Wit de tweede zet kiezen. Zwart heeft op het hoogste niveau nu de keus uit waarden van -1, 0 en -1. De keuze zal vallen op de tweede zet die een maximale waarde van 0 opbrengt. De hier beschreven techniek van minimaliseren en maximaliseren heet het *minimax algoritme* [4].

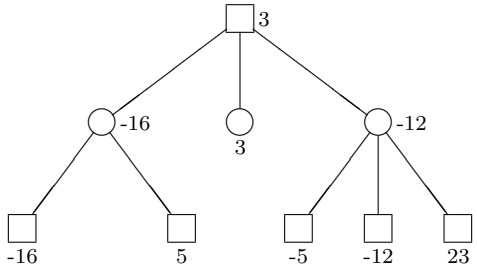


Figuur 4.2: Een uitgebreidere variantenboom.

Het nadeel van het toepassen van minimax bij spelen als renju is dat de situaties waarin winst voor een van beide spelers of een remise voorkomt vaak tientallen zetten diep liggen.

Hier komt bij dat per situatie ongeveer 200 varianten mogelijk zijn. Dit resulteert in een variantenboom die zo groot is, dat geen enkele computer hem door zal kunnen rekenen. Om dit probleem op te lossen is een techniek ontwikkeld die de variantenboom slechts tot een bepaalde diepte doorzoekt. Deze techniek heet *fixed-depth search* [4]. Een nadeel van deze aanpak is echter weer dat op voorhand niet bepaald kan worden wat een goede diepte is. Een volgende stap is daarom het toepassen van een *iterative-deepening* zoekproces [4]. Hierbij worden meerdere fixed-depth zoekprocessen na elkaar uitgevoerd, steeds met een grotere zoekdiepte. Ieder zoekproces resulteert in een zet. Hoe groter de zoekdiepte, des te beter zal de zet zijn. Het zoekproces zou na een aantal iteraties afgekapt kunnen worden, bijvoorbeeld omdat een bepaalde tijdsduur verstreken is. Het resultaat is een zo goed mogelijke zet binnen een controleerbare tijdsduur.

Een probleem dat nog niet behandeld is, is dat van de beoordeling van de eindsituaties in een fixed-depth zoekproces. Omdat slechts tot een bepaalde diepte gezocht wordt, is het niet zo dat de spelsituaties op de maximale diepte een winst– of remise-situatie voorstellen. Toch is hier een waarde vereist om op het hoogste niveau tot een goede zetkeuze te komen. Om dit op te lossen wordt een *evaluatiefunctie* [4] gebruikt die een schatting geeft van het evenwicht in het spel. Niet alleen worden zoals eerder de waarden -1, 0 en 1 gebruikt, maar ook alle tussenliggende waarden. Voor het gemak worden de grenzen -1 en 1 opgeschaald naar -100.000 en 100.000. Het doel is het onderling kunnen vergelijken van spelsituaties in het minimax algoritme. Wanneer spelsituatie A gunstiger is voor Zwart dan spelsituatie B, zal de evaluatiefunctie voor A een hogere waarde moeten teruggeven dan voor B. In het minimax algoritme heeft dit het gewenste resultaat dat Zwart vanwege de evaluatiewaarden A zal verkiezen boven B. Hoe de evaluatiefunctie tot een waarde komt staat vermeld op pagina 37 bij de beschrijving van het positionele zoekproces.



Figuur 4.3: Een variantenboom met evaluatiewaarden.

Vaak is het mogelijk te snoeien in de variantenboom. Hiermee wordt bedoeld dat niet alle varianten compleet doorzocht hoeven worden om toch tot de juiste zet te komen. Dit principe staat bekend onder de naam *alpha-beta* [4] en zal worden toegelicht aan de hand van figuur 4.3. Deze figuur toont een variantenboom met drie mogelijke zetten voor Zwart op het hoogste niveau. Na de eerste zet voor Zwart zijn er twee varianten voor Wit. Wit zal hiervan de variant met de minimale evaluatiewaarde kiezen; in dit geval de eerste met een waarde van -16. Zwart maximaliseert op het hoogste niveau en heeft na het doorzoeken van de eerste twee varianten de keuze uit zetten met de waarden -16 en 3. In de derde

variant voor Zwart heeft Wit de keuze uit drie zetten. Na evaluatie van de eerste hiervan blijkt een waarde van -5. Ongeacht de waarden van de laatste twee zetten zal Wit, omdat hij minimaliseert, een score behalen die niet groter zal zijn dan -5. Vergelijken we dit met de waarde 3 van de tweede variant voor Zwart dan blijkt op dit punt al dat Zwart nooit voor de derde variant zal kiezen. Het verder doorzoeken van deze variant heeft hierdoor geen enkele zin meer. De laatste twee varianten van Wit kunnen gesnoeid worden.

Om zo veel mogelijk te kunnen snoeien is het noodzakelijk dat de boom goed geordend is. Wanneer in het voorbeeld de tweede en derde zet voor Zwart omgedraaid zouden worden, zou er niet gesnoeid kunnen worden. Wanneer de eerste en tweede zet omgedraaid zouden worden zou ook in de, dan, tweede tak gesnoeid kunnen worden. Hoe beter de boom geordend is, des te meer kan er gesnoeid worden. Bij de bespreking van de tactische en positionele zoekprocessen zal besproken worden hoe de zetten geordend worden.

Het tactisch zoekproces

Het zoekproces dat Noire uitvoert is op te splitsen in twee onderdelen. In het eerste onderdeel wordt gepoogd te bewijzen of de geldende spelsituatie winst kan opleveren voor de speler aan zet. Dit wordt gedaan door gebruik te maken van een techniek die in [10] is besproken als een *tactisch zoekproces*. De opzet is om het aantal mogelijke zetten beperkt te houden tot enkel die zetten die een dreiging vormen en een dreiging verdedigen. Deze zetten vormen de zogenaamde *threatspace* [2, 8]. Door het beperkt aantal zetten ontstaat een erg smalle variantenboom. Dit heeft tot gevolg dat binnen afzienbare tijd erg diep gezocht kan worden. Hierdoor kunnen winsten gevonden worden tot op 30 tot 40 zetten diep, iets dat zonder het gebruik van deze techniek vrijwel onmogelijk is.

Een vereiste voor een gegarandeerde winstvariant is dat in het zoekproces minstens die zetten worden bekeken die een dreiging voor de aanvaller kunnen weerleggen. Bij Noire worden deze zetten bekeken, de zogenaamde *geforceerde zetten* [2, 8]. Dit zijn precies die zetten die de tegenstander kan doen om te voorkomen dat hij het spel direct verliest. Van alle mogelijke zetten verliest de verdediger wanneer hij niet een van de geforceerde zetten speelt. De geforceerde zetten zelf zijn de enige zetten die worden meegenomen in de variantenboom. De speler aan zet kan zijn tegenstander dwingen tot het doen van geforceerde zetten door zelf *forcerende zetten* te doen. Forcerende zetten zijn precies die zetten die de tegenstander dwingen te antwoorden met een geforceerde zet.

Noire herkent twee soorten forcerende en geforceerde zetten. De eerste soort bevat forcerende zetten die een drie-groep uitbreiden tot een vier-groep. De bijbehorende geforceerde zet is de zet die de vier-groep verdedigt. Dit is tevens de enige mogelijkheid die de verdediger heeft om te voorkomen dat hij verliest. De tweede soort bevat de forcerende zetten die een open drie-groep creëren. De tegenstander moet deze verdedigen omdat anders een niet te stoppen open vier-groep kan ontstaat. De geforceerde zetten zijn de twee of drie zetten die deze dreiging ongedaan kunnen maken. Los van deze zetten kan de verdediger zelf in de aanval gaan wanneer hij één of meerdere drie-groepen heeft. Door een dergelijke groep uit te breiden, zal de oorspronkelijke aanvaller zich eerst moeten verdedigen. Hiermee kan een geforceerde zet van het tweede type een forcerende zet van het eerste type

worden. Voor Zwart geldt in verband met de overlines de regel dat zetten slechts forcerend zijn wanneer zich in de randvelden van de groep die de dreiging veroorzaakt geen zwarte stenen bevinden.

De zetten voor beide spelers worden geordend aan de hand van de evaluatiefunctie in een 1-diep minimax zoekproces. Er wordt tevens gebruik gemaakt van de killer-heuristiek [4]. Deze techniek benoemt op iedere zoekdiepte een killerzet. Dit is de zet die het beste resultaat levert. Wanneer het zoekproces op een zoekdiepte belandt waarvoor reeds een killerzet is bepaald, zal het deze zet, als hij voorkomt in de gegenereerde zetten, als eerste doorzoeken. De besparingen die hierdoor behaald worden kunnen enorm zijn. De filosofie achter de killerzet is dat de beste zet die gevonden wordt op een bepaalde diepte in een zoekproces wellicht ook de beste zet zal zijn in andere posities op dezelfde diepte in de zoekboom. Het is daarom verstandig om de killerzet, indien aanwezig, als eerste te beschouwen in een nieuw zoekproces.

Het programma kijkt wanneer de aanvaller aan zet is eerst of hij zelf geforceerd wordt. Dit zou kunnen voorkomen wanneer de verdediger met een geforceerde zet zelf een dreiging creëert waarop gereageerd moet worden. Wanneer dit niet zo is zal hij aanvallen door in zijn zetselectie de forcerende zetten te kiezen. Zijn tegenstander zal hierop niets anders kunnen doen dan te reageren met geforceerde zetten. Wanneer de aanvaller eerder zelf geforceerd werd, kan het zijn dat er geen dreiging meer bestaat voor de verdediger. In dit geval wordt de zoektak niet verder doorzocht en wordt een waarde van 0 teruggegeven.

Wanneer in een zoektak een situatie voorkomt waarbij een speler slechts één geforceerde zet ter beschikking heeft, wordt het zoekproces verlengd. De reden hiervoor is dat de zet die gedaan moet worden enkel deze ene zet kan zijn. Het eigenlijke zoekproces wordt hierdoor met één niveau verkort. Om dit te voorkomen wordt de zoekdiepte met één opgehoogd wanneer de verdediger aan zet is en zelfs met twee wanneer de aanvaller aan zet is.

Voordat de zetten gegenereerd worden, wordt eerst aan de hand van de kennisregels bekeken of de situatie gewonnen is voor één van beide spelers. De kennisregels worden per niveau doorzocht. Dit wil zeggen dat eerst de regels worden doorzocht die directe winst opleveren. Daarna worden de regels die winst in één zet beschrijven. Daarna de regels die winst in twee zetten beschrijven, enzovoort. Per niveau worden de regels eerst doorzocht voor de speler aan zet en pas daarna voor zijn tegenstander. Wanneer een kennisregel van toepassing is, wordt niet doorgezocht naar eventuele andere kennisregels die van toepassing zijn. Wanneer een kennisregel van toepassing is voor de tegenstander van de speler aan zet, zal de speler aan zet een zet moeten doen die voorkomt dat de kennisregel na zijn zet nog altijd van toepassing is. Als hij dit verzuimt, zal zijn tegenstander vervolgens een winstsituatie aangereikt krijgen. Wanneer een kennisregel van toepassing is voor de speler aan zet wordt niet verder gezocht in de huidige zoektak [5]. De waarde die wordt teruggegeven is de maximale score verminderd met het niveau van de kennisregel en de diepte waarop de regel geldt. Zo zal een situatie die op 12 diep in de zoekboom onderzocht wordt en waarin een kennisregel van niveau 7 geldt een waarde teruggeven van $100.000 - 12 - 7 = 99.981$. Op het hoogste niveau kan hieruit afgeleid worden dat een winstvariant van 19 diep gevonden is. Wanneer een winstvariant voor de aanvaller gevonden is, worden de eventuele andere varianten niet verder onderzocht.

Via minimax wordt uiteindelijk op het hoogste niveau een waarde bepaald. Wanneer deze waarde groter is dan 99.775[1], staat vast dat een winstvariant gevonden is. Het zoekproces wordt beëindigd en de zet die wordt voorgesteld door het zoekproces wordt gespeeld. Wordt na alle iteraties van het iterative-deepening zoekproces geen dergelijke waarde gevonden dan moet geconcludeerd worden dat het tactische zoekproces geen winstvariant heeft kunnen vinden. In dit geval vervolgt het programma met een positioneel zoekproces.

Het positioneel zoekproces

Wanneer het tactische zoekproces geen winstvariant heeft kunnen blootleggen, wordt een nieuw zoekproces gestart. In dit zoekproces worden alle mogelijke zetten opgenomen in de variantenboom. Het gevolg hiervan is dat veel minder diep gezocht kan worden dan bij het tactische zoekproces binnen dezelfde tijd. Dit onderdeel dient dan ook enkel voor het vinden van een redelijke zet wanneer niet bewezen kan worden dat de partij gewonnen is.

In tegenstelling tot het tactische zoekproces is bij dit onderdeel een evaluatiefunctie nodig voor het beoordelen van de knopen op het laagste niveau van de boom. De evaluatiefunctie die hier gebruikt is, is erg eenvoudig van aard. Het is enkel een gewogen sommatie van de aantallen groepen van iedere cardinaliteit die iedere speler heeft. Groepen met een hogere cardinaliteit leveren zoals te verwachten is een hogere score op. Het verschil in de totale score van beide spelers levert uiteindelijk de evaluatiewaarde. In formulevorm ziet dit er als volgt uit:

$$E = \sum_{i=1}^{5} (\omega_i^{\alpha} \cdot \theta_i^{\alpha} - \omega_i^{\delta} \cdot \theta_i^{\delta})$$

Hierbij staat E voor de evaluatiewaarde, i voor de cardinaliteit, α voor de aanvaller, δ voor de verdediger, ω voor een gewichtsfactor en θ voor het aantal groepen van een bepaalde cardinaliteit dat een speler heeft. Er worden verschillende gewichtsfactoren gehanteerd voor de aanvaller en de verdediger. Dit is gedaan omdat de aanvaller het initiatief heeft. Daardoor worden zijn groepen meer waard dan die van de verdediger. De gewichtsfactoren die gehanteerd worden zijn $\omega^{\alpha} = [\ 1 \quad 16 \quad 81 \quad 256 \quad 100.000\ ]$ en $\omega^{\delta} = [\ 1 \quad 9 \quad 49 \quad 169 \quad 100.000\ ]$. Deze waarden zijn overgenomen uit het onderzoek van Uiterwijk [10]. Groepen worden enkel als zwarte groepen meegeteld wanneer er zich in de randvelden van deze groepen geen zwarte stenen bevinden. Alleen deze groepen zijn voor Zwart interessant voor de evaluatiefunctie omdat zij tot vijf-groepen kunnen worden uitgebreid. De groepen met zwarte stenen in de randvelden worden niet tot de zwarte groepen gerekend en worden derhalve ook niet in de evaluatiefunctie meegenomen. Er wordt een kleine bonus toegekend voor zetten die dicht in de buurt van het centrum van het speelbord liggen.

Ook hier is een goede zetordening belangrijk. Op het diepste niveau gebeurt dit door middel van de positie op het bord. Hoe dichter een zet bij het centrum van het bord ligt, des te eerder wordt hij doorzocht. Op alle andere niveaus gebeurt dit door het doen van een 1-diep zoekproces, gecombineerd met de killer-heuristiek.

[1] Deze waarde is gelijk aan de maximale evaluatiewaarde verminderd met het aantal velden op het bord.

Wanneer het positionele zoekproces afgerond is, wordt een zet teruggegeven. Het is echter mogelijk dat deze zet leidt tot een situatie waarin de tegenstander een gegarandeerde winst kan behalen. Om dit te voorkomen wordt na het zoekproces een *tactische check* uitgevoerd [10]. Deze check gaat na of de situatie die ontstaat na het doen van de voorgestelde zet winst oplevert voor de tegenstander. Dit wordt gedaan door een tactisch zoekproces uit te voeren vanuit het oogpunt van de tegenstander nadat de eigen zet gedaan is. Wanneer dit zoekproces geen winst kan vinden, kan de zet veilig worden uitgevoerd. Wanneer dit wel zo is, wordt een nieuw positioneel zoekproces gestart waarbij op het hoogste niveau de eerder gevonden zet niet als optie wordt beschouwd. Dit proces wordt herhaald totdat een zet is gevonden die geen winst voor de tegenstander oplevert, of totdat bewezen is dat alle zetten winst opleveren voor de tegenstander.

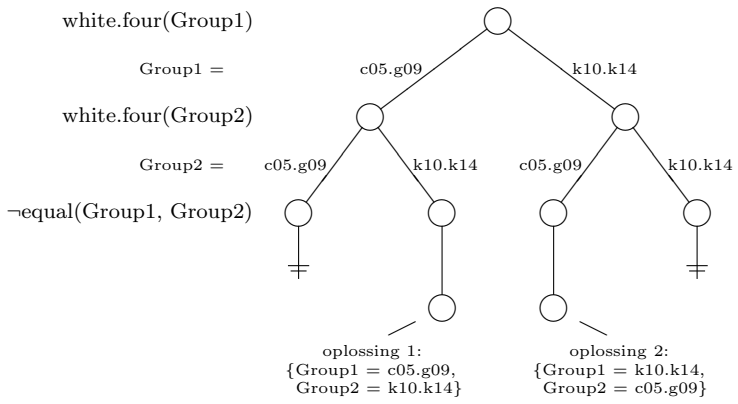
4.2 Het inferentiemechanisme

Het inferentiemechanisme heeft als functie te bepalen of een kennisregel van toepassing is in een gegeven spelsituatie. Het nagaan of een kennisregel van toepassing is, wordt het *matchen* van een kennisregel genoemd. Om een kennisregel te matchen, worden de literals die deel uitmaken van de premisse van de kennisregel één voor één gematcht. Het matchen van een literal houdt in dat bekeken wordt of de literal van toepassing is in de gegeven situatie en eventueel onder welke omstandigheden. Het kan tot op zekere hoogte vergeleken worden met het oplossen van een wiskundige formule.

> De wiskundige formule $x^2 = 9$ kan gematcht worden, met oplossingen $x = 3$ en $x = -3$. De formule $x^2 = -9$ kan niet gematcht worden omdat een kwadraat nooit negatief kan zijn. De formule $3 = 3$ kan gematcht worden zonder dat een oplossing nodig is. De formule $x + y = 4$ kan gematcht worden, maar heeft oneindig veel oplossingen.

Wanneer de literal *white.four(Group1)* gematcht moet worden, wordt verwacht dat er geen oplossing is wanneer zich op het speelbord geen witte vier-groepen bevinden. Wanneer dit wel het geval is, zou het antwoord moeten luiden dat de literal gematcht kan worden en wel precies dan wanneer de variabele *Group1* een waarde aanneemt die overeenkomt met één van de witte vier-groepen die aanwezig zijn op het bord.

Wanneer een literal matcht, wordt voor ieder van de oplossingen gekeken of de volgende literal gematcht kan worden. Wanneer een literal niet gematcht kan worden, wordt de volgende oplossing van de vorige literal doorzocht. Wanneer een punt wordt bereikt waarin alle literals gematcht zijn, kan geconcludeerd worden dat de kennisregel van toepassing is. De waardetoekenningen die op dit punt gelden worden een oplossing voor de kennisregel genoemd. Het proces van het matchen van een kennisregel wordt schematisch weergegeven in figuur 4.4. In deze figuur wordt aangegeven hoe het matchen van de premisse *white.four(Group1), white.four(Group2), ¬equal(Group1, Group2)* plaatsvindt. Er wordt uitgegaan van een speelbord waarop zich twee witte vier-groepen bevinden. Deze groepen krijgen hier de namen *c05.g09* en *k10.k14* mee. De drie literals die gematcht moeten



Figuur 4.4: Het matchen van een kennisregel.

worden, staan aan de linker zijkant aangegeven. Daar tussenin staan in kleinere letters de variabelen waaraan een waarde wordt toegekend. Deze waarden worden naast de takken in de boom aangegeven. Iedere waardetoekenning staat voor een oplossing van een literal. Wanneer een literal niet gematcht kan worden wordt dit aangegeven aan de hand van het teken $\dashv$. Wanneer naast een tak geen oplossing vermeld staat, wil dit zeggen dat de literal gematcht kan worden maar dat hiervoor geen extra elementen aan de deeloplossing hoeven worden toegevoegd. In het onderste gedeelte van de figuur wordt aangegeven welke oplossingen gevonden zijn. Aan de hand van deze oplossingen kan eventueel een zet bepaald worden die gespeeld moet worden om de winstsituatie uit te buiten.

Het matchen van een literal gebeurt aan de hand van de predikaten en afleidingen die in hoofdstuk 3 zijn beschreven. Er wordt gekeken of het predikaat en het aantal argumenten van de literal overeenkomen met een van de beschreven predikaten uit §3.3. Wanneer dit zo is, wordt een procedure uitgevoerd die nagaat of de literal gematcht kan worden in de geldende situatie. Als voorbeeld wordt hier de procedure besproken die uitgevoerd wordt voor het predikaat *in*. De overige procedures zijn van vergelijkbare aard.

> Het predikaat *in* telt vier argumenten. Het eerste argument is een veld, het tweede een groep. Het derde en vierde argument zijn getallen tussen 0 en 6. Deze twee argumenten zullen hierna de namen i_1 en i_2 meekrijgen. Ze moeten reeds een waarde hebben omdat anders de literal niet gematcht kan worden. Het predikaat bekijkt of het veld van het eerste argument element is van de groep van het tweede argument en voorkomt op één van de plaatsen tussen het derde en het vierde argument. Zo gaat de aanroep *in(Field1, Group1, 1, 5)* na of *Field1* een basisveld is van *Group1*. De literal *in(Field2, Group2, 0, 6)* gaat na of *Field2* element is van *Group2* waarbij ook de randvelden bekeken worden.
>
> Wanneer zowel het eerste als het tweede argument een waarde hebben wordt bekeken of het veld zich op één van de posities tussen i_1 en i_2 van de groep bevindt. Als dit zo is kan de literal gematcht worden, anders niet. Er worden geen extra oplossingen teruggegeven.

Wanneer enkel het eerste argument een waarde heeft, kan de literal altijd gematcht worden. De oplossingen die teruggegeven worden zijn precies die groepen waarbij het bekende veld op één van de posities tussen i_1 en i_2 ligt. Deze groepen kunnen snel bepaald worden omdat voor ieder veld op het bord wordt bijgehouden van welke groepen zij deel uitmaken.

Ook wanneer alleen het tweede argument een waarde heeft, kan de literal gematcht worden. De antwoorden die hier gelden zijn de velden die op de posities i_1 tot en met i_2 van de groep liggen. Ook deze informatie is snel te achterhalen omdat deze reeds in het programma is opgeslagen.

De situatie waarin geen van de eerste twee argumenten een waarde hebben is niet meegenomen in het programma. Het is vergelijkbaar met het oplossen van de eerder genoemde formule $x + y = 4$. Het aantal oplossingen is simpelweg te groot, doch hier niet oneindig. Het gevolg hiervan is dat het erg lang duurt om de literal op te lossen. Door het goed ordenen van de literals in de kennisregels en afleidingen kan vaak worden voorkomen dat dit soort problemen opgelost moeten worden. In het programma zijn deelprocedures die dergelijke problemen oplossen niet opgenomen.

Wanneer een literal gematcht moet worden waarbij de combinatie van predikaat en aantal argumenten niet overeenkomt met één van de predikaten uit §3.3, worden de afleidingen aangesproken. Voor iedere afleiding wordt bekeken of hier het predikaat en het aantal argumenten in de premisse wel overeenkomen met die van de te onderzoeken literal. Wanneer dit zo is wordt geprobeerd de body van de afleiding te matchen. Wanneer dit lukt is meteen de literal gematcht. Voordat de body gematcht wordt, worden eerst alle waardetoekenningen die gelden overgedragen op de premisse.

In dit voorbeeld gaan we uit van het matchen van de literal *overlap(Group2, black, 3, Field1)*, zoals die voorkomt in de kennisregel die een kruis van type 2 voor Wit beschrijft. We gaan ervan uit dat er reeds een deeloplossing bestaat met de waardetoekenningen *Group2=c10.g6* en *Field1=f7*. De literal matcht qua predikaat en aantal argumenten met de afleiding met premisse *overlap(Group, Color, N, Field)*. Voordat de body van de afleiding gematcht wordt, worden binnen de afleidingen de volgende waardetoekenningen gedaan. Group krijgt als waarde c10.g6. Color krijgt de waarde black, N de waarde 3 en Field tenslotte de waarde f7.

Wanneer een afleiding zou bestaan met als premisse *overlap(Group, black, 2, Field)* zou deze niet aanmerking komen om gematcht te worden omdat de derde argumenten van literal en premisse van de afleiding niet overeenkomen. In een dergelijk geval kan de literal niet aan de hand van deze afleiding gematcht worden.

Hierna wordt geprobeerd de body van de afleiding te matchen. Dit gaat op dezelfde wijze als het matchen van de premisse van een kennisregel. Eventueel wordt hierbij ook gebruik gemaakt van afleidingen. Wanneer de body niet gematcht kan worden wordt het zoekproces voor de huidige afleiding gestaakt. Wanneer de body wel gematcht kan worden, worden de oplossingen teruggegeven aan de literal. De literal bepaalt voor iedere oplossing of deze toegevoegd wordt aan zijn eigen oplossingen. Dit gebeurt wanneer zich in de oplossing

van de afleiding waardetoekenningen bevinden die binnen de literal onbekend zijn voor variabelen die deel uitmaken van de argumenten van de literal.

> Als eerste voorbeeld wordt het matchen van de literal *open(e3.e7)* bekeken aan de hand van de afleiding *open(Group) :- in(Field1, Group, 0, 0), empty(Field1), in(Field2, Group, 5, 5), empty(Field2).* Wanneer de groep *e3.e7* inderdaad een open groep is, zullen de variabelen *Field1* en *Field2* tijdens het matchen van de body een waarde hebben gekregen. Omdat deze variabelen niet in de gevraagde literal voorkomen, zullen hun waarden niet worden opgenomen in de deeloplossing.
>
> In het volgende voorbeeld gebeurt dit echter wel. Hier wordt het matchen van de literal *white.three(Group2)* aan de hand van de afleiding *white.three(Group) :- group(Group, white, 3)* besproken. Stel dat na het matchen van de body, de variabele *Group* de waarde *k8.k12* heeft. Deze waarde wordt als waarde voor de variabele *Group2* in de literal toegevoegd aan de deeloplossing.

Wanneer een kennisregel matcht wordt aan de hand van de eerste oplossing een zet bepaald. De zet die binnen de kennisregel is opgenomen zal hiervoor één van de variabelen moeten zijn die ook in de premisse voorkomt. Dit wordt op voorhand nagegaan voor alle kennisregels. De waarde die de variabele aanneemt in de oplossing is de zet die gedaan zal worden en die de speler aan zet de eerste stap levert in de richting van de overwinning.

4.3 De gebruikersinterface

In deze sectie wordt de gebruikersinterface beschreven. Dit wordt gedaan aan de hand van de menu-items die in het programma ingebouwd zijn. Aan de hand van deze menu-items kunnen de basisfunctionaliteiten van het programma benut worden.

Een nieuw spel beginnen Om een nieuw renjuspel te beginnen is een wizard ontwikkeld die enkele vragen stelt. Deze wizard bestaat uit drie formulieren waarin onder andere de aard en kleur gevraagd worden van de beide spelers. In figuur 4.5 worden deze formulieren getoond. Nadat alle vragen beantwoord zijn, wordt een nieuw spel begonnen. Het programma onthoudt de antwoorden die gegeven worden, zodat deze automatisch kunnen worden ingevuld op het moment dat de wizard op een later tijdstip opnieuw wordt opgestart. Uiteraard kunnen deze waarden, indien gewenst, overschreven worden.

Wanneer de wizard is doorlopen, wordt een nieuw spel gestart. In figuur 4.6 wordt een afbeelding getoond van een spel tussen mens en computer dat op deze manier is opgestart. Menselijke spelers kunnen hierop door een klik met de muis een zet doen. Ook is de mogelijkheid gegeven voor mensen om te passen. Het speelvenster kan vergroot en verkleind worden. De inhoud van het venster waaronder de gespeelde stenen zullen zich hierbij aan de grootte van het venster aanpassen.

Een bestaand spel laden Het is mogelijk ieder willekeurig legitiem renjuspel in het programma in te laden. Zo kan bijvoorbeeld een eerder afgebroken spel tussen twee

Figuur 4.5: Drie screenshots uit de wizard ‘nieuw spel’.

Figuur 4.6: Screenshot van een beginnend spel.

menselijke spelers worden hervat. Een andere mogelijkheid is om verschillende stadia uit een spel dat ooit door twee grootmeesters is gespeeld aan de computer aan te bieden om te kijken hoe deze het spel uitspeelt. De teststellingen die in het volgende hoofdstuk voor de experimenten worden gebruikt, kunnen op deze manier in het programma worden ingeladen.

Een spel opslaan Gedurende een spel kunnen de gegevens van het spel tezamen met de gedane zetten worden opgeslagen. Hierdoor kan een spel worden stopgezet en op een later moment worden hervat.

Taalinstellingen De complete gebruikersinterface is opgesteld in de Engelse taal. Om de gebruiksvriendelijkheid te verhogen is een taalmodule opgezet die de complete interface om kan zetten naar een andere taal. Er is hiermee een Nederlandse taalmodule ontwikkeld. Alle termen die in de interface voorkomen zijn naar het Nederlands vertaald en in een los tekstbestand geplaatst. Op deze manier kan ook voor andere talen een taalmodule ontwikkeld worden. In het programma wordt de mogelijkheid geboden een taal te selecteren voor de interface.

Kenniseditor De afleidingen en kennisregels kunnen binnen het programma bekeken worden door middel van de kenniseditor. Een screenshot van de kenniseditor staat vermeld in figuur 4.7. Aan de hand van deze kenniseditor kunnen de kennisregels gewijzigd worden. De functionaliteit van de kenniseditor houdt hier echter op. Gewenste functies als het wijzigen van afleidingen of toevoegen of verwijderen van afleidingen of kennisregels zijn niet opgenomen. Buiten het programma om is het uiteraard toch mogelijk deze zaken uit te voeren. Dit zal plaatsvinden door het direct manipuleren van de tekstbestanden die de afleidingen en kennisregels bevatten.

Figuur 4.7: Screenshot van de kenniseditor.

Opties Aan de hand van dit formulier kunnen enkele instellingen die het programma gebruikt ingezien of gewijzigd worden. Een screenshot van dit formulier is afgebeeld in figuur 4.8. Voorbeelden van opties die ingesteld kunnen worden zijn onder andere de kleur van het speelbord.

Figuur 4.8: Screenshot van de programma-opties.

Hoofdstuk 5

Experimenten en resultaten

Dit hoofdstuk beschrijft de experimenten die gedaan zijn en de resultaten die hiermee bereikt zijn. In de eerste paragraaf wordt de aard van de experimenten geschetst. In de paragrafen 5.2 en 5.3 worden de twee verzamelingen teststellingen beschreven die gebruikt zijn om de experimenten uit te voeren. In de laatste paragraaf worden tenslotte de resultaten behandeld die behaald zijn met de experimenten.

5.1 Aard van de experimenten

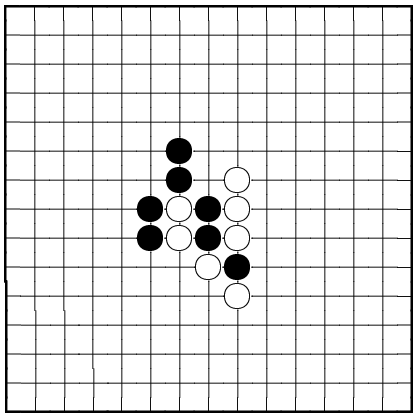
De experimenten die binnen dit onderzoek gedaan zijn, hebben betrekking op het tactische zoekproces van Noire. Aan de hand van een aantal stellingen wordt de kracht van dit zoekproces onderzocht. Van de onderzochte stellingen is bekend dat ze ofwel voor gomoku, ofwel voor renju gewonnen zijn voor de speler aan zet [10, 14]. De vraag is of Noire deze winst kan vinden. De opties van Noire zijn hierbij zodanig ingesteld dat in theorie oneindig diep gezocht wordt naar een oplossing.

Voor de gevallen waarin een winst gevonden wordt, wordt bekeken wat de invloed van een toename kennis hierop is. De experimenten worden hiervoor uitgevoerd met verschillende verzamelingen kennisregels, ieder met een verschillend kennisniveau[1]. Voor ieder kennisniveau wordt bekeken hoeveel posities in de variantenboom doorzocht moeten worden voordat de winst gevonden wordt. De veronderstelling is dat een hoger kennisniveau tot gevolg heeft dat het aantal doorzochte posities af zal nemen.

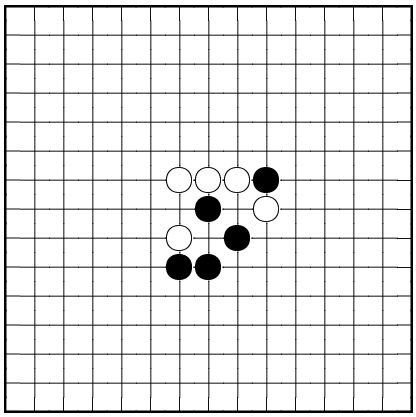
5.2 Stellingen van Uiterwijk

De eerste verzameling stellingen is de verzameling die door Uiterwijk in [10] is beschreven. De stellingen zijn afgebeeld in de figuren 5.1 en 5.2. In deze figuren is tevens aangege-

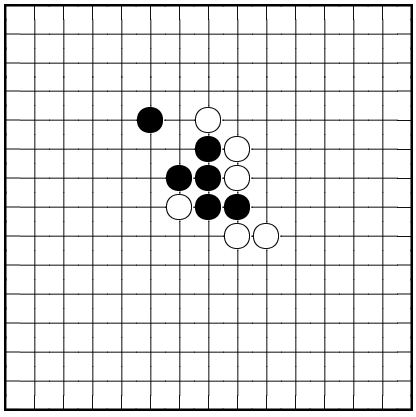
[1]Met een kennisniveau wordt bedoeld dat binnen het zoekproces slechts kennisregels behandeld worden die winst beschrijven op een diepte die niet boven een bepaalde waarde uitkomt. Zo wordt met kennisniveau 3 bedoeld, dat enkel kennisregels van niveau 0 tot en met 3 behandeld worden. De experimenten die hier gedaan zijn, zijn ingedeeld in kennisniveau 1, 3, 5, en 7.



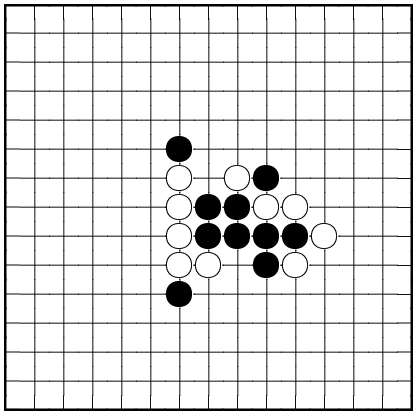
Stelling 1 (Zwart aan zet).



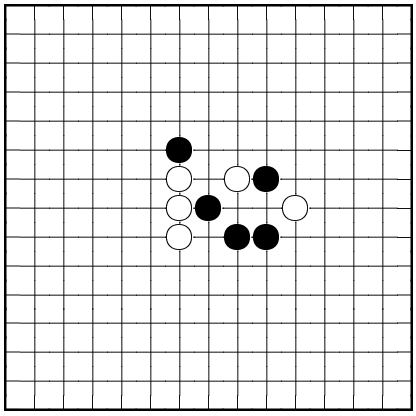
Stelling 2 (Zwart aan zet).



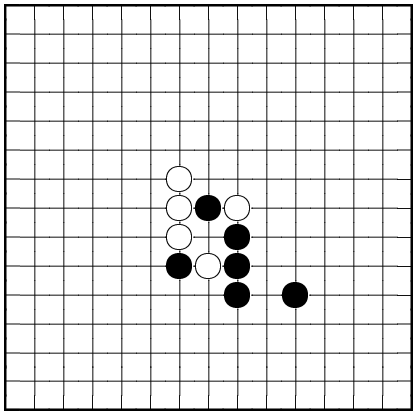
Stelling 3 (Zwart aan zet).



Stelling 4 (Zwart aan zet).

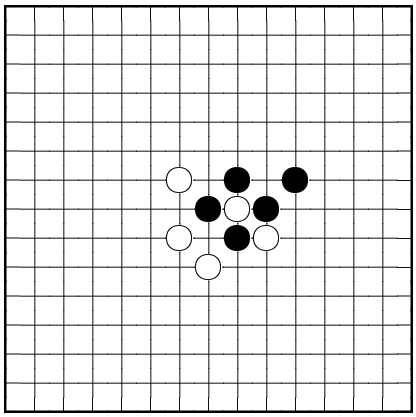


Stelling 5 (Zwart aan zet).

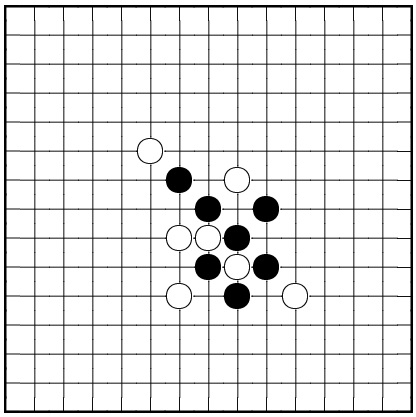


Stelling 6 (Wit aan zet).

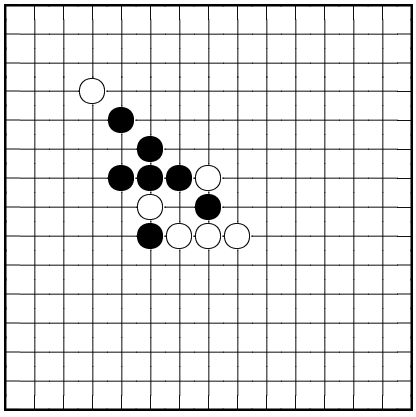
Figuur 5.1: Stellingen 1 tot en met 6 van Uiterwijk.



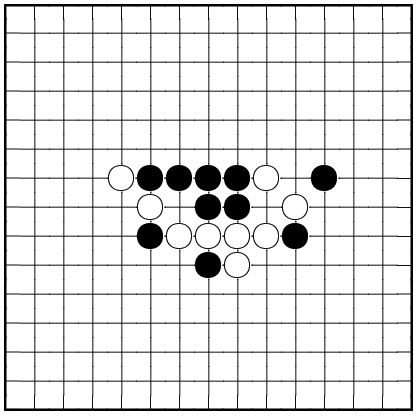
Stelling 7 (Zwart aan zet).



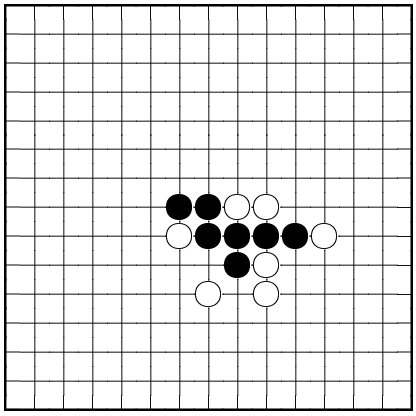
Stelling 8 (Zwart aan zet).



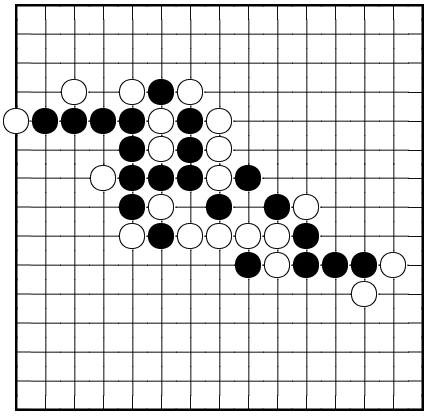
Stelling 9 (Wit aan zet).



Stelling 10 (Wit aan zet).



Stelling 11 (Zwart aan zet).



Stelling 12 (Wit aan zet).

Figuur 5.2: Stellingen 7 tot en met 12 van Uiterwijk.

ven welke speler aan zet is. Deze verzameling bestaat uit stellingen die gewonnen zijn voor go-moku. Alle stellingen komen voort uit toernooi-partijen van Uiterwijk's go-moku-programma *Polygon*. Het onderzoek van Uiterwijk heeft aangetoond dat aan de hand van een tactisch zoekproces alle hier beschreven stellingen als gewonnen kunnen worden herkend met het programma dat voor go-moku is ontwikkeld. Wanneer dezelfde stellingen door NOIRE bekeken worden, blijkt dat niet in alle gevallen een winstvariant gevonden kan worden. De reden hiervoor is dat in het zoekproces bepaalde situaties voorkomen die bij go-moku toegestaan zijn, maar bij renju niet. Hierdoor zullen bepaalde winstvarianten die voor go-moku gelden, niet voor renju gelden.

5.3 Stellingen van Nosovsky

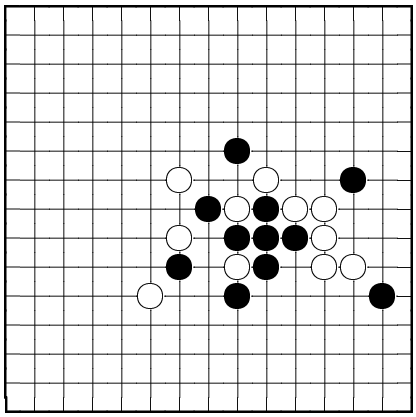
De tweede verzameling stellingen die gebruikt is, is afkomstig van de webpagina van Nosovsky [14]. Deze stellingen zijn afgebeeld in de figuren 5.3 en 5.4. Zij zijn gebruikt als criterium voor het renju wereldkampioenschap voor computerprogramma's in 2000. Deze stellingen zijn ontworpen met een erg hoge moeilijkheidsgraad. De reden hiervoor is dat aan de hand van deze stellingen precies díe programma's geselecteerd kunnen worden die in hoge mate in staat zijn probleemstellingen op te lossen. Deze selectie zegt echter niets over het algemene kwaliteit van de programma's.

Ook deze stellingen zijn niet alle door NOIRE herkend als gewonnen. De reden hiervoor is dat NOIRE in het tactische zoekproces enkel dreigingen en bedreigde situaties beschouwt. Bij sommige stellingen moeten zogenaamde "stille" zetten beschouwd worden om een oplossing te vinden. Wanneer kennisregels ontwikkeld zouden worden die niet alleen forcerende, maar ook stille zetten bevatten, kunnen waarschijnlijk meer stellingen opgelost kunnen worden.

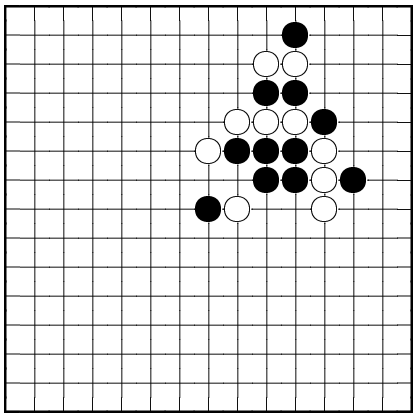
5.4 Behaalde resultaten

In deze sectie worden de resultaten besproken die zijn behaald met het doen van experimenten met de hiervoor genoemde stellingen. De stellingen hebben hierbij een naam gekregen. De naamgeving is opgebouwd uit een voorvoegsel en een volgnummer. De stellingen van Uiterwijk hebben als voorvoegsel `JU` gekregen, de stellingen van Nosovsky het voorvoegsel `AN`. Het volgnummer komt overeen met de nummers die in de figuren 5.1 tot en met 5.4 vermeld zijn. Deze volgnummers worden onder de 10 voorafgegaan door een 0. Stelling 6 van Uiterwijk krijgt hiermee als naam `JU06`.

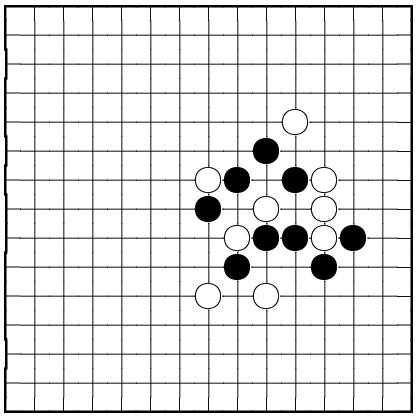
De resultaten zijn onderverdeeld in drie tabellen. In tabel 5.1 worden de resultaten van het onderzoek naar winstvarianten getoond. In deze tabel wordt voor de onderzochte stellingen aangegeven of een winstvariant gevonden is. Wanneer dit het geval is, wordt aangegeven voor welke speler de winst gevonden is en in hoeveel zetten. Tevens wordt de gevonden winstvariant getoond. De zetten worden hierbij gescheiden door een punt. Door het symbool · wordt de winstvariant in twee delen gesplitst. Het eerste deel is de



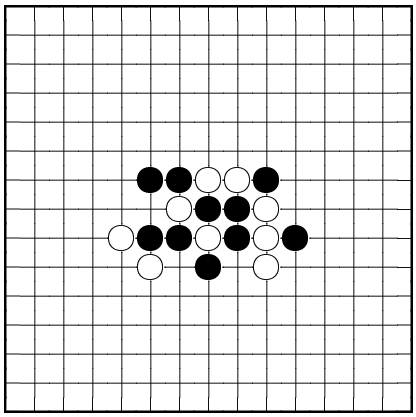
Stelling 1 (Zwart aan zet).



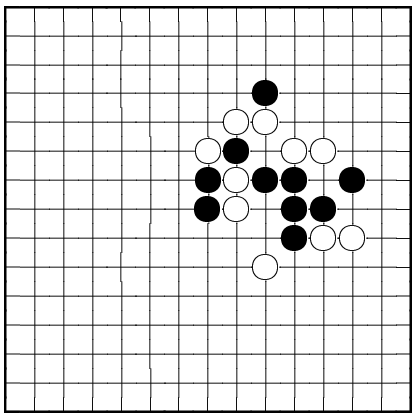
Stelling 2 (Wit aan zet).



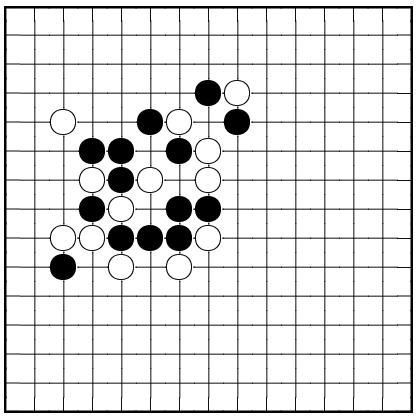
Stelling 3 (Zwart aan zet).



Stelling 4 (Wit aan zet).

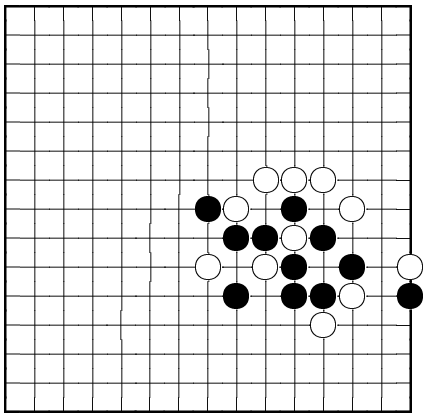


Stelling 5 (Zwart aan zet).

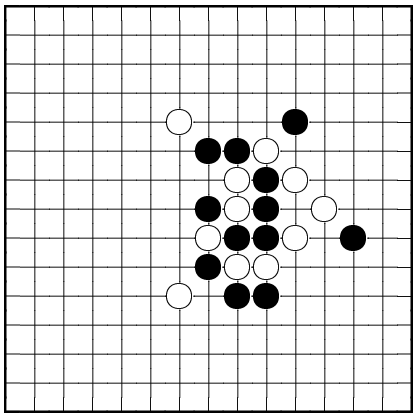


Stelling 6 (Wit aan zet).

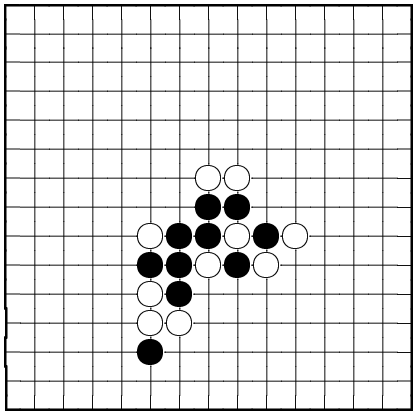
Figuur 5.3: Stellingen 1 tot en met 6 van Nosovsky.



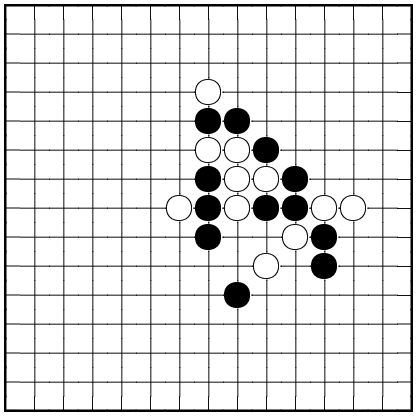
Stelling 7 (Zwart aan zet).



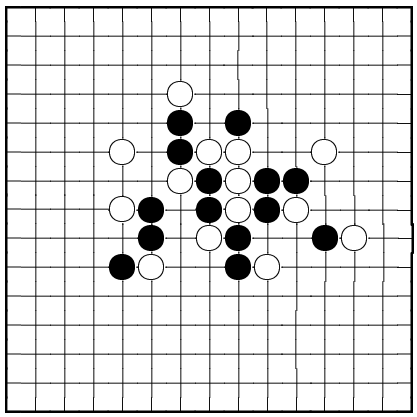
Stelling 8 (Wit aan zet).



Stelling 9 (Zwart aan zet).



Stelling 10 (Wit aan zet).



Stelling 11 (Zwart aan zet).

Figuur 5.4: Stellingen 7 tot en met 11 van Nosovsky.

stelling	winstvariant
JU01	winst gevonden voor Zwart in 21 zetten (H10.I11.I10.F10.H11.H9.K8.J9.L9.J7.I12.K6.L5.J4.K3.F9·K10.J10.J11.M8.H13)
JU02	winst gevonden voor Zwart in 29 zetten (I6.F6.H7.F9.E9.F5.H4.H5.K4.F7.F8.F3.F4.G8.J5.L3.E3.G5.J6.K6.K5.L4.I8.K10.I5.I4·K7.L8.G3)
JU03	winst gevonden voor Zwart in 17 zetten (F8.I11.F9.I12.I13.F7.F10.F12.E12.G10·E11.D12.E9.D9.E10.E8.E13)
JU04	winst gevonden voor Zwart in 21 zetten (I5.H4.L8.M9.I6.I4.K10.L11.K5.L4.J5.H5.J3.J4.K4.L3·L5.M5.I2.M6.H1)
JU05	geen winst gevonden
JU06	winst gevonden voor Wit in 23 zetten (G10.G11.H9.F11.F7.E6.F9.I4.I3.E9.H11.E8.I12.J13.E7.J6.L4.D7·I10.J11.I9.J9.I11)
JU07	geen winst gevonden
JU08	winst gevonden voor Zwart in 13 zetten (K9.L10.L8.K7.I11.J10.I8.K8·F8.G8.H10.E7.J12)
JU09	winst gevonden voor Wit in 21 zetten (J7.K7.I8.G10.L5.K6.I5.I4.J5.K5.K4.K8.K9.C9.D9.I6·J8.H6.J6.J9.J4)
JU10	geen winst gevonden
JU11	winst gevonden voor Zwart in 17 zetten (F9.E10.H9.H6.G9.E9.H10.H11.F8.E7.E8.D8·F10.E11.F7.F6.F11)
JU12	winst gevonden voor Wit in 19 zetten (I10.J9.H13.H12.F13.E14.I14.J15.J11.K12.I12.C6.D7.K10·I13.I11.G13.E13.J13)
AN01	winst gevonden voor Zwart in 29 zetten (L4.K5.J4.J5.H4.G3.I4.K4.F4.G4.H6.G5.H7.H5.H10.H9.I9.G11.L2.K3.L10.K9.J10.K10.K11.L12·J12.N8.I13)
AN02	winst gevonden voor Wit in 21 zetten (H11.G11.L13.N13.M12.H13.M13.I13.M11.M14.L6.L7.L12.K8.M6.K6.K7.I15·N10.O9.J14)
AN03	geen winst gevonden
AN04	geen winst gevonden
AN05	geen winst gevonden
AN06	winst gevonden voor Wit in 19 zetten (C8.C10.F10.H6.F8.F5.A10.B9.I9.F12.J9.G9·L9.K9.K10.I8.J11.H13.M8)
AN07	winst gevonden voor Zwart in 33 zetten (J5.H5.I4.H3.I3.I6.N5.O4.N3.M4.I2.I1.K4.J3.N7.O8.N4.N6.K3.K2.N1.N2.L3.M2.M3.O3.L2.K1.J4.M1·L6.H2.M7)
AN08	geen winst gevonden
AN09	winst gevonden voor Zwart in 11 zetten (D4.E5.L9.K8.D8.E7·G8.G9.J5.F9.K4)
AN10	geen winst gevonden
AN11	winst gevonden voor Zwart in 17 zetten (F11.H11.F10.F9.E9.J10.K10.G8.D11.E14.F13.D8·E12.D13.C10.G14.B9)

Tabel 5.1: Resultaten van onderzoek naar winstvarianten.

eigenlijke winstvariant die door het programma gevonden wordt. Aan het einde van een dergelijke variant is niet een reeds gewonnen situatie bereikt, maar is enkel een kennisregel van toepassing. De winstvariant die door de kennisregel wordt beschreven is het tweede deel van de weergegeven varianten. De resultaten zijn verkregen aan de hand van een tactisch zoekproces met kennisniveau 7. Dit komt neer op het gebruik van alle in hoofdstuk 3 beschreven kennisregels.

In tabel 5.2 worden de winstvarianten die NOIRE vindt bij de stellingen van Uiterwijk vergeleken met de winstvarianten zoals ze door Uiterwijk zelf gevonden zijn. De winstvarianten van NOIRE worden als eerste vermeld, gevolgd door die van Uiterwijk op een nieuwe regel. Alle varianten in deze tabel zijn gevonden door gebruik te maken van kennisniveau van 7. Uit de vergelijking blijkt dat in Uiterwijk's onderzoek vaak even lange of kortere winstvarianten gevonden worden. Slechts in enkele gevallen vindt NOIRE een kortere winstvariant. In veel gevallen zijn de winstvarianten voor een deel aan elkaar gelijk of bevatten zij voor een deel dezelfde zetten.

stelling	winstvariant	
JU01	H10.I11.I10.F10.H11.H9.K8.J9.L9.J7.I12.K6.L5.J4.K3.F9·K10.J10.J11.M8.H13	(21 zetten)
	F10.F9.H10.K3.J4.E10·E7.D6.I11	(9 zetten)
JU02	I6.F6.H7.F9.E9.F5.H4.H5.K4.F7.F8.F3.F4.G8.J5.L3.E3.G5.J6.K6.K5.L4.I8.K10.I5.I4·K7.L8.G3	(29 zetten)
	I6.F6.H7.F5.H5.H4.J7.F9.E9.F7.F8.F3.F4.K8.I8.K10.K6.J6.L7.K7.I5.I4·G8.E8.J5.K4.G5.K5.E3	(29 zetten)
JU03	F8.I11.F9.I12.I13.F7.F10.F12.E12.G10·E11.D12.E9.D9.E10.E8.E13	(17 zetten)
	F8.I11.F9.I12.I13.F7.F10.F12.E9.D9·E11.D12.E12.G10.E10.E8.E13	(17 zetten)
JU04	I5.H4.L8.M9.I6.I4.K10.L11.K5.L4.J5.H5.J3.J4.K4.L3·L5.M5.I2.M6.H1	(21 zetten)
	I5.H4.K5.L4.J5.H5.I6.I4.K4.L3·L5.M5.J3.J4.I2.H1.M6	(17 zetten)
JU05	-	
	J8.J10.J6.J5.H6.K9.H7.H9·K7.L7.H4.H5.I5.G3.L8	(15 zetten)
JU06	G10.G11.H9.F11.F7.E6.F9.I4.I3.E9.H11.E8.I12.J13.E7.J6.L4.D7·I10.J11.I9.J9.I11	(23 zetten)
	G10.G11.H9.J7.F9.E9.I12.J6.L4.I4.I3.H11.I9.J9·I10.I11.F7.E6.J11	(19 zetten)
JU07	-	
	L10.M11.L6.K7.L9.H9.L8.L7.M9.J9.L12.L11·J10.K11.M7.N6.I11	(17 zetten)
JU08	K9.L10.L8.K7.I11.J10.I8.K8·F8.G8.H10.E7.J12	(13 zetten)
	L8.H4.I8.K8.J7.J9.K7.M9·M7.L7.K9.L10.N6.O5.J10	(15 zetten)
JU09	J7.K7.I8.G10.L5.K6.I5.I4.J5.K5.K4.K8.K9.C9.D9.I6·J8.H6.J6.J9.J4	(21 zetten)
	J7.K7.I8.G10.I6.I5.L9.D9.C9.K8.I9.I10·J9.K9.G6.F5.K10	(17 zetten)
JU10	-	
	J5.G8.J6.J8.G4.H5.J4.J3.K4.L3.I4.H4.L4.M4·I3.I5.K5.M3.H2.G1.L6	(21 zetten)
JU11	F9.E10.H9.H6.G9.E9.H10.H11.F8.E7.E8.D8·F10.E11.F7.F6.F11	(17 zetten)
	H9.H6.F9.E10.H11.H10.E8.G10.F8.D8.F10.F7·G9.E11.E9.D9.I9	(17 zetten)
JU12	I10.J9.H13.H12.F13.E14.I14.J15.J11.K12.I12.C6.D7.K10·I13.I11.G13.E13.J13	(19 zetten)
	I10.J9.F13.E14.H13.H12.J11.G8.I12.D7.C6.G14.K10.L9.K12.L13.I14.J15·I13.I11.G13.E13.J13	(23 zetten)

Tabel 5.2: Vergelijking van winstvarianten met Uiterwijk.

Met de 15 stellingen van Uiterwijk en Nosovsky waarbij een winstvariant gevonden is, is nieuw onderzoek gedaan. Op vier verschillende kennisniveaus is bekeken hoeveel posities doorzocht moeten worden voordat de winstvariant gevonden is. De resultaten van dit onderzoek staan vermeld in tabel 5.3. Voor bepaalde stellingen is op kennisniveau 1

geen winstvariant gevonden. Aan de hand van gegevens die door het programma worden bijgehouden tijdens het zoekproces kan worden afgeleid dat in ieder van deze situaties de grens van 150 miljoen onderzochte situaties ruim is overschreden. In de tabel zijn voor deze situaties waarden van 10 miljoen gebruikt om de vertekening met de overige situaties niet te groot te maken.

stelling	kennisniveau 1		kennisniveau 3		kennisniveau 5		kennisniveau 7	
JU01	>10.000.000	(100.0%)	1.351.443	(13.5%)	270.449	(2.7%)	293.647	(2.9%)
JU02	>10.000.000	(100.0%)	76.301	(0.8%)	13.418	(0.1%)	76.377	(0.8%)
JU03	>10.000.000	(100.0%)	59.587	(0.6%)	57.480	(0.6%)	57.409	(0.6%)
JU04	>10.000.000	(100.0%)	101.599	(1.0%)	99.762	(1.0%)	99.762	(1.0%)
JU06	>10.000.000	(100.0%)	633.669	(6.3%)	36.107	(0.4%)	32.413	(0.3%)
JU08	>10.000.000	(100.0%)	58.955	(0.6%)	61.383	(0.6%)	69.357	(0.7%)
JU09	>10.000.000	(100.0%)	61.279	(0.6%)	11.662	(0.1%)	11.312	(0.1%)
JU11	>10.000.000	(100.0%)	1.308	(0.0%)	447	(0.0%)	447	(0.0%)
JU12	146.973	(100.0%)	241.393	(164.2%)	72.920	(49.6%)	73.227	(49.8%)
AN01	75	(100.0%)	68	(90.7%)	68	(90.7%)	68	(90.7%)
AN02	>10.000.000	(100.0%)	9.221	(0.1%)	8.780	(0.1%)	8.772	(0.1%)
AN06	>10.000.000	(100.0%)	9.686	(0.1%)	11.527	(0.1%)	8.429	(0.1%)
AN07	1.466	(100.0%)	1.327	(90.5%)	1.543	(105.3%)	1.543	(105.3%)
AN09	>10.000.000	(100.0%)	398.873	(4.0%)	35.169	(0.4%)	37.241	(0.4%)
AN11	>10.000.000	(100.0%)	71.952	(0.7%)	6.789	(0.1%)	6.791	(0.1%)
totaal	>120.148.514	(100.0%)	3.076.661	(2.6%)	687.504	(0.6%)	776.795	(0.6%)

Tabel 5.3: Resultaten van onderzoek naar aantallen onderzochte posities.

Uit de tabel valt af te lezen dat verreweg de meeste stellingen laten zien dat het verhogen van het kennisniveau tot een waarde van 5 een verlaging van het aantal doorzochte posities veroorzaakt. Deze constatering is een bevestiging van de vijfde doelstelling uit hoofdstuk 1. De verhoging tot kennisniveau 7 blijkt weinig extra invloed te hebben. In enkele gevallen komt het voor dat de aantallen doorzochte posities stijgen naarmate het kennisniveau toeneemt. Een verklaring hiervoor is te vinden in het feit dat gebruik gemaakt wordt van de killer heuristiek (zie §4.1). Wanneer deze techniek niet gebruikt wordt, is de variantenboom van een bepaald kennisniveau een subverzameling van de variantenboom van een lager kennisniveau. Onderin de variantenboom worden afhankelijk van het kennisniveau meer of minder situaties doorzocht. Dit heeft echter geen effect op de andere delen van de boom. Wanneer de killer heuristiek wel gebruikt wordt, heeft dit wel effect. Onder invloed van het kennisniveau kan een bepaald niveau in de boom wel of niet onderzocht worden. Wanneer de killer heuristiek gebruikt wordt, wordt de beste zet bewaard wanneer dit niveau wel doorzocht wordt. Wanneer op een andere plek in de boom op hetzelfde niveau gezocht wordt, wordt eerst de voorgestelde zet behandeld. Hierdoor kan, afhankelijk van

het kennisniveau, de boom anders opgebouwd worden. De opzet van de killer heuristiek is uiteraard om hierbij voor een verbetering te zorgen, maar dit hoeft niet altijd het geval te zijn. In de gevallen waarbij het aantal doorzochte posities toeneemt met het kennisniveau werkt de killer heuristiek in het nadeel. Wanneer de heuristiek niet toegepast zou worden, zou weliswaar een afname verschijnen, maar zouden de aantallen doorzochte posities vele malen hoger uitvallen. De killer heuristiek heeft daarom wel degelijk nut.

Om te kijken wat het algemene nut van een hoger kennisniveau is, zijn alle aantallen posities in de onderste rij van tabel 5.3 gesommeerd. Hier blijkt nog eens duidelijk hoe sterk de afname is. Tussen de onderlinge kennisniveaus 1 en 3 is de besparing ruim 97%. Tussen de kennisniveaus 3 en 5 komt daar nog eens een besparing van ruim 77% bij. Tussen de kennisniveaus 5 en 7 is geen sprake van besparing. Door de killer heuristiek is het totaal aantal doorzochte posities bij kennisniveau 7 groter dan bij kennisniveau 5. Met de huidige verzameling kennisregels is het daarom gunstiger een kennisniveau van maximaal 5 te hanteren. Het toevoegen van extra regels van kennisniveau 7 kan echter in de toekomst een flinke besparing opleveren. Toch is de totale besparing tussen de kennisniveaus 1 en 7 ruim 99%. In de praktijk heeft deze besparing tot gevolg dat binnen een vooraf vastgelegde tijdsspanne die voor het doen van één zet geldt, veel vaker een winst gevonden kan worden wanneer een hoger kennisniveau gehanteerd wordt.

Hoofdstuk 6

Conclusies en aanbevelingen

In dit hoofdstuk wordt primair bekeken in hoeverre de gestelde doelstellingen gerealiseerd zijn. Verder worden enkele conclusies genoemd die aan de hand van het gedane onderzoek getrokken zijn. Als afsluiting worden enkele aanbevelingen voor vervolgonderzoek genoemd.

6.1 Realisatie van de doelstellingen

De doelstellingen die in hoofdstuk 1 genoemd zijn, worden hier stuk voor stuk behandeld.

De eerste doelstelling is het ontwikkelen van een formele taal. Deze taal is daadwerkelijk ontwikkeld en wel op basis van de predikaatlogica. De opbouw van de ontwikkelde taal, die de naam Blanche heeft meegekregen, staat beschreven in hoofdstuk 2. Hiermee is de eerste doelstelling gehaald.

De tweede doelstelling draait om het opstellen van kennisregels die gebruikt kunnen worden om winsten in renjusituaties te kunnen herkennen. Deze kennisregels en de entiteiten, predikaten en afleidingen die gebruikt zijn om de regels op te stellen zijn in hoofdstuk 3 behandeld. Validatie van de kennisregels heeft aangetoond dat ze inderdaad gewonnen renjusituaties kunnen herkennen. Hiermee is ook de tweede doelstelling gehaald.

In hoofdstuk 4 worden de derde en vierde doelstellingen behandeld. Deze doelstellingen hebben betrekking op een inferentiemechanisme voor de kennisregels en een computerprogramma dat gebaseerd is op dit inferentiemechanisme en de kennisregels en dat in staat is renju te spelen. In hoofdstuk 4 wordt behandeld hoe het computerprogramma en het bijbehorende inferentiemechanisme zijn opgebouwd en hoe zij functioneren. Het programma heeft de naam Noire gekregen. In dit hoofdstuk wordt ook behandeld welk vermogen Noire heeft om renju te spelen. Beide doelstellingen over programma en inferentiemechanisme zijn aan de hand van dit hoofdstuk gehaald.

De laatste doelstelling laat onderzoeken of het zoekproces van het ontwikkelde programma ingekort kan worden naarmate meer kennis aan het programma wordt aangeboden. De experimenten die voor dit onderzoek gedaan zijn en de resultaten die hiermee behaald zijn, staan in hoofdstuk 5 vermeld. De resultaten tonen aan dat het gebruiken van

meer kennis in het programma het zoekproces inderdaad verkort. Hiermee is ook de vijfde doelstelling gerealiseerd.

De in hoofdstuk 1 genoemde probleemstelling voor dit project luidde:

> Is het mogelijk een renjuprogramma te maken dat gebaseerd is op extern opgeslagen kennisregels?

Er is een renjuprogramma gemaakt, zoals blijkt uit het helen van de vierde doelstelling. Het halen van de derde doelstelling toont aan dat in het programma een inferentiemechanisme is opgenomen dat gebruik van extern opgeslagen kennis mogelijk maakt. De kennisregels die als basis dienen zijn opgesteld binnen het domein van het spel renju, zoals blijkt uit de tweede doelstelling. De communicatie tussen programma en extern opgeslagen kennisregels wordt geregeld door de formele taal uit de eerste doelstelling. Aan de hand van deze uiteenzetting mag geconcludeerd dat aan de probleemstelling voldaan is.

6.2 Conclusies naar aanleiding van het onderzoek

Verdere conclusies die getrokken kunnen worden, zijn gebaseerd op de verschillende onderdelen van het onderzoek.

De opzet van de ontwikkelde taal Blanche is geslaagd voor de doeleinden waarvoor zij ontwikkeld is. Als algemeen inzetbare kennisrepresentatietaal is zij echter te beperkt. Het ontbreken van de mogelijkheid tot het creëren van functies of het gebruik van een of-teken zullen een dergelijke inzetbaarheid snel in de weg staan.

De indeling van de kennis in predikaten, afleidingen en kennisregels is een goede geweest. De predikaten zijn weliswaar in het programma opgenomen en daardoor weinig flexibel, maar zijn daarmee de noodzakelijke en domein-specifieke elementen die aanwezig moeten zijn voor een verdere kennisopbouw. Zij zijn dusdanig gekozen dat zij een zo breed mogelijke kennisverzameling kunnen beschrijven. De afleidingen en kennisregels worden buiten het programma bijgehouden en zijn daardoor naar eigen keuze te wijzigen of uit te breiden. De afleidingen die hier gedefinieerd zijn, zijn deels algemeen en herbruikbaar en deels specifiek voor bepaalde kennisregels ontwikkeld. Deze verzameling kan uitgebreid worden om nieuwe domeinen te beschrijven. De kennisregels blijken goed in staat winstsituaties te beschrijven. Een uitbreiding van de verzameling kennisregels kan zeker nuttig zijn maar voorzichtigheid is hier geboden. Een veelheid aan kennisregels hoeft niet altijd in het voordeel van de prestaties van het programma te zijn. Wanneer kennisregels aan het systeem worden toegevoegd die erg specifieke situaties beschrijven die in de praktijk zelden voorkomen, zullen de prestaties waarschijnlijk afnemen. Dergelijke kennisregels zullen iedere keer doorzocht moeten worden waarbij slechts in een zeer enkel geval een regel van toepassing zal zijn.

Het ontwikkelde programma Noire werkt goed, maar kan uitbreidingen gebruiken. Qua interne werking kan deze uitbreiding resulteren in een betere evaluatiefunctie. Een uitbreiding aan de buitenkant van het programma kan leiden tot een betere kenniseditor.

6.3 Aanbevelingen voor vervolgonderzoek

Aan de hand van de ervaringen die gedurende dit onderzoek opgedaan zijn, zijn de volgende vier aanbevelingen voor vervolgonderzoek geformuleerd. De eerste hiervan is het uitvoeren van de uitbreidingen aan de verschillende onderdelen van dit onderzoek, zoals ze in §6.2.

Tevens kan onderzoek gedaan worden naar het optimaliseren van de volgorde van de literals in de premissen van de kennisregels. Door de meest restrictieve literals vooraan in de premisse te plaatsen, zal eerder duidelijk worden wanneer een kennisregel niet van toepassing is. Hierdoor zal het matchen van kennisregels sneller gaan verlopen, wat ten goede komt aan de kracht van het programma. In het gedane onderzoek is zeker gelet op de volgorde van de literals. Er kan echter niet beweerd worden dat deze volgorde optimaal is. Door verder onderzoek zou eventueel een schema ontwikkeld kunnen worden waarmee de volgorde van literals in reeds ontwikkelde en de nog te ontwikkelen kennisregels geoptimaliseerd kan worden.

Door het flexibele karakter van de verzameling kennisregels kan de pure essentie van het spel veranderd worden. Wanneer een bepaalde subverzameling van de hier beschreven kennisregels wordt geconstrueerd, is het programma in staat het spel go-moku te spelen. Met deze tweede verzameling kennisregels kan een vergelijking gemaakt worden met het onderzoek van Uiterwijk [10]. Met een dergelijk kan worden bekeken of alle stellingen uit hoofdstuk 5 nu wel allemaal opgelost kunnen worden. Tevens zou een toernooi opgesteld kunnen worden tussen de programma's Noire en Polygon om eventueel aspecten van beide programma's te verbeteren.

Een laatste aanbeveling voor vervolgonderzoek is te onderzoeken of kennisregels door de computer zelf opgesteld kunnen worden. Iedere keer dat door het programma een evaluatiewaarde wordt teruggegeven die boven de waarde 99775 ligt, is er een winstsituatie gevonden. Door een uitbreiding zou het programma in staat moeten kunnen zijn deze situaties te herkennen en er patronen in te herkennen. De eerder genoemde waarschuwing is ook hier op zijn plaats. Er moet gewaakt worden voor te specifieke situaties die wel winst opleveren maar zo zelden voorkomen dat het toevoegen van een kennisregel zelden tot voordeel zal leiden. Een hulpmiddel voor dit probleem zou kunnen zijn om een kennisregel slechts aan de bestaande verzameling toe te voegen wanneer de bijbehorende situatie meer dan een bepaald aantal malen is voorgekomen. Dit aantal zou een absolute drempelwaarde kunnen zijn, maar bijvoorbeeld ook een bepaald percentage.

Referenties

[1] L.V. Allis, *Searching for solutions in games and artificial intelligence*, Ph.D. thesis, Rijksuniversteit Limburg (1994). ISBN 90-9007488-0.

[2] L.V. Allis, H.J. van den Herik, M.P.H. Huntjens, *Go-moku solved by new search techniques*, in: *Computational intelligence: an international journal*, Blackwell publishers (1996), pp. 7-24. ISSN 0824-7935.

[3] J.F.A.K. van Benthem, H.P. Ditmarsch, J. Ketting, W.P.M. Meyer-Viol, *Logica voor informatici, tweede editie*, Addison-Wesley (1994). ISBN 90-6789-4842.

[4] P. van Diepen, H.J. van den Herik, *Schaken voor computers*, Academic service (1987). ISBN 90-6233-271-4.

[5] H.J. van den Herik, *Computerschaak, schaakwereld en kunstmatige intelligentie*, Academic service (1983). ISBN 90-6233-093-2.

[6] V.J. Rayward-Smith, *Inleiding in de theorie van formele talen*, Academic service (1986). ISBN 90-6233-242-0.

[7] G. Sakata, W. Ikawa, *Five-in-a-row, renju*, The ishi press (1981).

[8] T. Thomsen, *Lambda-search in game trees - with application to go*, in: *ICGA journal*, vol. 23, no. 4 (2000), pp. 203-217. ISSN 1389-6911.

[9] J.W.H.M. Uiterwijk, *Kennisbehandeling in positionele spelen: computeranalyse van four-in-a-row*, in: J. Treur (Ed.), *Proceedings NAIC'91*, Stichting informatica congressen, Amsterdam (1991), pp. 193-207. ISBN 90-5005-038-7.

[10] J.W.H.M. Uiterwijk, *Knowledge and strategies in go-moku*, in: H.J. van den Herik, L.V. Allis (Eds.), *Heuristic programming in artificial intelligence 3 (the third computer olympiad)*, Ellis Horwood (1992), pp. 165-179. ISBN 0-13-388265-9.

[11] J.W.H.M. Uiterwijk, H.J. van den Herik, *The advantage of the initiative*, in: *Information sciences*, (2000), pp. 43-58. ISSN 0020-0255.

[12] J. Wágner, I. Virág, *Solving renju*, in: *ICGA journal*, vol. 24, no. 1 (2001), pp. 30-35. ISSN 1389-6911.

[13] http://www.lemes.se/renju, *De officiële webpagina van de Renju International Federation.*

[14] http://www.japan-games.da.ru, *Alexander Nosovsky's WWW page.*