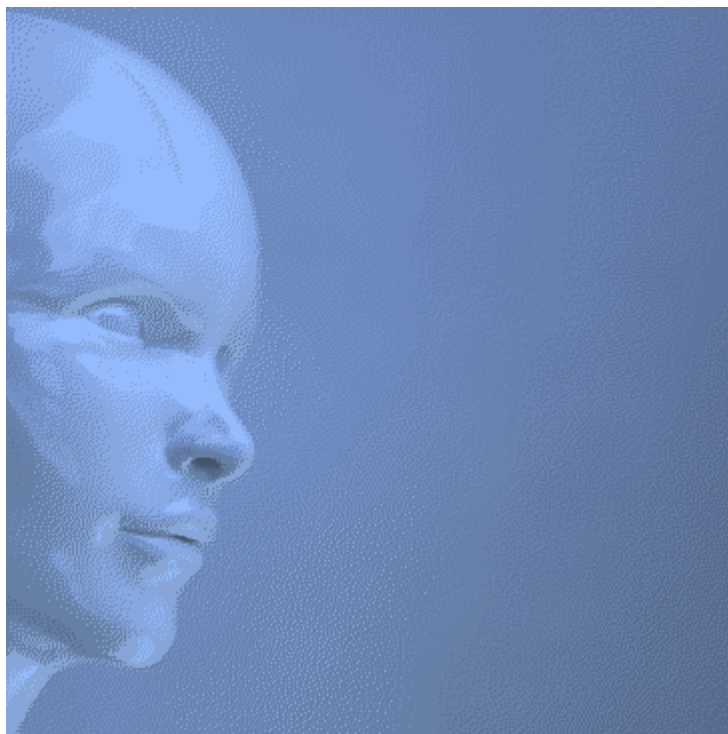


Kunnen computers denken?

Ontdek het zelf



Inhoudsopgave

1. Inleiding.....	4
1.1. Het Thinknowlogy-concept.....	4
2. De algebra en logica in natuurlijke taal.....	5
2.1. Een generalisatie/specificatie-definitie.....	5
2.1.1. Een relatie-specificatie.....	5
2.1.2. Verzamelingen.....	6
2.1.2.1. Een specificatie-verzameling.....	6
2.1.2.2. Een relatie-verzameling.....	6
2.1.2.3. Een generalisatie-verzameling.....	6
2.1.3. Toewijzingen.....	7
2.1.3.1. Een specificatie-toewijzing.....	7
2.1.3.2. Een generalisatie-toewijzing.....	7
2.1.3.3. Toewijzingen en grammaticale verleden tijd.....	8
2.2. Selecties.....	9
2.3. Het zelfstandig trekken van conclusies.....	10
2.3.1. Specificatie-substitutie conclusies.....	10
2.3.1.1. Samengestelde specificatie-substitutie conclusies.....	11
2.3.2. Bezittelijke omkeer-conclusies.....	12
2.4. Het doen van aannames.....	13
2.4.1. Generaliseerde aannames.....	13
2.4.2. Bezittelijke voorwaardelijke-specificatie-aannames.....	14
2.5. Het zelfstandig stellen van vragen.....	15
2.5.1. Vragen om het aanscherpen van informatie te stimuleren.....	15
2.6. Het vinden van oorzakelijke verbanden.....	15
3. Het opbouwen van de kennisstructuur.....	16
3.1. Het creëren van een generalisatie/specificatie.....	16
3.1.1. Het creëren van de nieuwe woorden.....	17
3.1.2. Het creëren van een specificatie-verbinding.....	18
3.1.2.1. Het creëren van een relatie-verbinding.....	18
3.1.3. Het creëren van een verzameling.....	19
3.1.4. Het creëren van een toewijzing.....	19
3.1.4.1. Het creëren van een specificatie-toewijzing.....	20
3.1.4.2. Het creëren van een generalisatie-toewijzing.....	20
3.2. Het creëren van een selectie.....	21
3.2.1. Het uitvoeren van selecties.....	21
4. Dialogen.....	22
4.1. De grammatica.....	22
4.1.1. Het lezen van een zin.....	22
4.1.1.1. Alleen acceptatie van bekende zinsstructuren.....	23
4.1.2. Het antwoorden met een zin.....	23
4.1.2.1. Het zelfstandig stellen van vragen.....	23
5. Toepassingen.....	24
5.1. Programmeren in natuurlijke taal.....	24

5.2. Het beoordelen en corrigeren van teksten en specificaties.....	24
5.3. Het geautomatiseerd maken van samenvattingen.....	25
5.4. Vertaalprogramma dat de context begrijpt.....	25
5.5. Een zoekmachine die vragen kan beantwoorden.....	25
5.6. Digitale docent of zelfstudiehulp.....	26

1. Inleiding

- Kun je meer met computersoftware bereiken dan veredelde rekentaken, opslag van bestanden en het automatiseren van bedrijfsprocessen?
- Zullen we ooit vragen kunnen stellen aan zoekmachines i.p.v. het zoeken van woorden en woordcombinaties op webpagina's?
- Zullen computers ooit met natuurlijke taal kunnen omgaan?
- Computers kunnen gegevens verwerken. Maar zullen computers ooit informatie kunnen verwerken? Anders gezegd: Zullen we ooit informatie kunnen automatiseren?

Er zijn vele pogingen ondernomen om computers te laten 'denken', maar als er geen concept achter die pogingen ligt, zijn zulke pogingen vruchteloos. Alleen met een goed doordacht concept heb je kans van slagen.

1.1. Het Thinknowlogy-concept

Het Thinknowlogy-concept kent de volgende principes:

- 1) Programmeertalen zijn gebaseerd op algebra en logica;
- 2) Ieder mens heeft een aangeboren gevoel voor algebra en logica, al is het bij de één sterker ontwikkeld dan bij de ander. Dit komt o.a. tot uiting in natuurlijke taal;
- 3) Door de algebra en logica in natuurlijke taal te combineren met de algebra en logica van programmeertalen, is het mogelijk om te programmeren in natuurlijke taal;
- 4) Het uiteindelijke doel is om informatie te automatiseren.

Eerst wordt onderzocht hoe de algebra en logica in natuurlijke taal gecombineerd kan worden met de algebra en logica van programmeertalen, zodat je kunt programmeren in natuurlijke taal.

Later wordt onderzocht hoe informatie te automatiseren is.

2. De algebra en logica in natuurlijke taal

Hieronder worden een aantal aspecten van de algebra en logica in natuurlijke taal beschreven.

2.1. Een generalisatie/specificatie-definitie

De mens organiseert zijn gedachten door te groeperen, in hokjes te plaatsen, te generaliseren. Generaliseren betekent eigenlijk: De hoofdzaken onderscheiden van bijzaken. De hoofdzaken noemen we **generalisaties** en de bijzaken noemen we **specificaties**.

Het proces van het onderscheiden van generalisaties en specificaties zit zelfs tot op het laagste niveau in de taal verstopt. Neem de zin: “*Jan is een mens.*”.

“*Jan*” is het onderwerp en “*mens*” is een eigenschap van “*Jan*”. Anders gezegd: “*Jan*” is de generalisatie van de specificatie “*mens*”, omdat “*mens*” iets specifiek zegt over het onderwerp “*Jan*”.

Meer specificaties van Jan:

- “*Jan is een vader.*”
- “*Jan is een bakker.*”

In de voorafgaande zinnen werd telkens een onbepaald lidwoord gebruikt: “*Jan is een mens.*”, “*Jan is een vader.*” en “*Jan is een bakker.*”. Deze structuur noemen we in het Thinknowlogy-concept een **generalisatie/specificatie-definitie**.

2.1.1. Een relatie-specificatie

Neem de zin: “*Jan is een vriend van Mark.*”.

Generalisatie “*Jan*” heeft een “*vriend*”-relatie met “*Mark*”. Hierbij wordt “*vriend*” de **specificatie** genoemd en “*Mark*” de **relatie**. Daarom wordt dit een **relatie-specificatie** genoemd.

2.1.2. Verzamelingen

Hieronder wordt besproken hoe diverse woorden 'verzameld' worden.

2.1.2.1. Een specificatie-verzameling

Bij een generalisatie/specificatie-definitie kan de generalisatie meerdere specificaties hebben die bij elkaar horen, zoals in het voorbeeld: “*Een glas is vol of leeg.*”. Of specifieker: “*Een glas is vol, halfvol of leeg.*”.

Zo’n groep van specificaties noemen we een **specificatie-verzameling**.

2.1.2.2. Een relatie-verzameling

Neem de zin: “*Jan is een vriend van Mark en Paula.*”.

In dit geval worden de relaties “*Mark*” en “*Paula*” verzameld, wat we dan ook een **relatie-verzameling** noemen.

2.1.2.3. Een generalisatie-verzameling

Neem nu de volgende zinnen:

- “*Bush is de vorige president van de Verenigde Staten.*”
- “*Obama is de huidige president van de Verenigde Staten.*”

Hierbij zijn de specificaties en relatie gelijk, maar zijn de generalisaties verschillend. De generalisaties “*Bush*” en “*Obama*” horen blijkbaar bij elkaar en worden dan ook verzameld.

Dit noemen we een **generalisatie-verzameling**.

2.1.3. Toewijzingen

Een **toewijzing** is een generalisatie/specificatie met een toegewezen waarde. Een generalisatie/specificatie is in principe statisch, maar een toewijzing is dynamisch, omdat de toestand kan wijzigen.

Een toewijzing geeft dus de actuele toestand van een generalisatie/specificatie aan en is te herkennen aan door het **bepaalde lidwoord** in die zin.

2.1.3.1. Een specificatie-toewijzing

Voorbeeld: “*Het glas is halfvol.*”.

Dit is een generalisatie/specificatie-structuur, waarbij een bepaald lidwoord wordt gebruikt en de specificatie tot een specificatie-verzameling behoort, bijvoorbeeld: “*Een glas is vol, halfvol of leeg.*”. Dit noemen we een **specificatie-toewijzing**.

Een toewijzing geeft de actuele toestand aan en kan dus ook wijzigen, want als het glas leeggedronken is, geldt een nieuwe toestand: “*Het glas is leeg.*”.

Een voorbeeld van een toewijzing van een relatie-specificatie: “*Jan is de vader van Piet.*”.

2.1.3.2. Een generalisatie-toewijzing

Neem de zinnen:

- “*Bush is de vorige president van de Verenigde Staten.*”
- “*Obama is de huidige president van de Verenigde Staten.*”

Dit zijn ook toewijzingen, omdat een bepaald lidwoord gebruikt, maar bij deze zinnen zijn de generalisaties verschillend en beide behoren tot dezelfde generalisatie-verzameling. Daarom worden deze toewijzingen **generalisatie-toewijzingen** genoemd.

2.1.3.3. Toewijzingen en grammaticale verleden tijd

Als een toewijzing wijzigt, noemen we de oude toestand in de taalkunde: **verleden tijd**.

Voor het glas gold ooit: “*Het glas is halfvol.*”. Maar als het glas is leeggedronken, geldt er:

- “*Het glas was halfvol.*”
- “*Het glas is (nu) leeg.*”

Ooit gold ook: “*Bush is de president van de Verenigde Staten.*”, maar na de verkiezingen is de toestand gewijzigd in:

- “*Bush is de vorige president van de Verenigde Staten.*” of “*Bush was de president van de Verenigde Staten.*”
- “*Obama is de (huidige) president van de Verenigde Staten.*”

2.2. Selecties

Natuurlijke taal voorziet ook in selecties, zoals bij de zin: *“Als het verkeerslicht geel of rood is, moet je stoppen.”*. Dus onder de conditie *“Het verkeerslicht is geel of rood”* geldt de actie *“moet je stoppen”*.

Vaak gelden er ook alternatieven. Als het verkeerslicht groen is, geldt hier: *“Je moet doorrijden.”*. Dus samengevoegd: *“Als het verkeerslicht rood of geel is, moet je stoppen, anders moet je doorrijden.”*.

Een selectie bestaat dus uit twee of drie delen:

- een conditie (*“Als het verkeerslicht geel of rood is”*),
- een actie (*“Je moet stoppen.”*)
- en eventueel een alternatieve actie (*“Je moet doorrijden.”*).

Hier gelden de generalisatie/specificatie-structuren:

- de conditie: *“Een verkeerslicht is geel of rood.”*;
- de acties: *“Je moet stoppen of doorrijden.”*.

Zowel condities als de acties zijn toewijzingen:

- De toewijzingen van een conditie worden gebruikt om te bepalen of de gegeven conditie waar is. De conditie uit het voorbeeld is dus waar als één van beide toewijzingen geldt: *“Het verkeerslicht is geel.”* of *“Het verkeerslicht is rood.”*;
- Afhankelijk van dit resultaat wordt dan de toewijzing van de actie of de alternatieve actie uitgevoerd, dus: *“Je moet stoppen.”* of *“Je moet doorrijden.”*.

2.3. Het zelfstandig trekken van conclusies

Wij zijn niet gewend dat software zelfstandig conclusies kan trekken. Hieronder wordt uitgelegd hoe dat te realiseren is.

2.3.1. Specificatie-substitutie conclusies

Beschouw de zinnen:

- “*Jan is een vader.*”
- “*Een vader is een man.*”

Allereerst een opmerking: De laatste zin noemen we een **definitie-zin**, omdat de generalisatie wordt gevormd door een onbepaald lidwoord en een zelfstandig naamwoord en ook de specificatie wordt gevormd door één of meer onbepaald lidwoorden en zelfstandig naamwoorden.

Uit deze zinnen mogen wij de conclusie te trekken: “*Jan is een man.*”.

Deze conclusie kan geautomatiseerd worden, door de volgende regels toe te passen:

- 1) Neem een zin, waarbij de (enkelvoudige) specificatie wordt gevormd door een onbepaald lidwoord en een zelfstandig naamwoord;
- 2) Zoek met die specificatie naar definitie-zinnen waarbij de bovengenoemde specificatie gelijk is aan de generalisatie van de definitie-zin;
- 3) Voor de eerstgenoemde zin geldt dan ook de specificatie van elke bijbehorende definitie.

Dus in het geval van de bovenstaande zin:

- 1) Neem de zin over “*Jan*”. De (enkelvoudige) specificatie van die zin is: “*een vader*”;
- 2) Voor die specificatie geldt de definitie: “*Een vader is een man.*”;
- 3) De software mag nu zelfstandig de conclusie trekken: “*Jan is een man.*”, door de specificatie van de zin over “*Jan*” te vervangen door de specificatie van de definitie-zin.

We noemen dit een **specificatie-substitutie conclusie**, omdat de specificatie van een eigenaam-zin vervangen mag worden door de specificatie van een definitie-zin.

2.3.1.1. **Samengestelde specificatie-substitutie conclusies**

Neem nu de zinnen:

- *“Een ouder is een vader of (een) moeder.”*
- *“Een vader is een man.”*
- *“Een moeder is een vrouw.”*

Volgens de (enkelvoudige) specificatie-substitutie conclusie mag de enkelvoudige specificatie van een zin worden vervangen door de specificatie van een definitie-zin. Maar ook bij een zin met een samengestelde specificatie, zoals *“Een ouder is een vader of (een) moeder.”*, mogen de specificaties worden vervangen als er voor beide een definitie bestaat.

Door zowel specificatie *“vader”* als *“moeder”* te vervangen, door respectievelijk *“man”* en *“vrouw”*, luidt de conclusie uit de bovenstaande zinnen: *“Een ouder is een man of (een) vrouw.”*

Zo'n conclusie noemen we een **samengestelde specificatie-substitutie conclusie**, omdat een samengestelde specificatie wordt vervangen door een andere definitie.

Een ander vorm van een samengestelde specificatie-substitutie conclusie ontstaat als we enkelvoudige specificatie vervangen door een samengestelde specificatie van een definitie-zin, zoals bij deze zinnen:

- *“Piet is een kind (van Jan).”*
- *“Een kind is een zoon of (een) dochter.”*

Hier luidt de conclusie: *“Piet is een zoon of (een) dochter (van Jan).”*

2.3.2. Bezittelijke omkeer-conclusies

Beschouw nu de zin: “*Jan is de vader van Piet.*”.

Uit de zin mogen we de conclusie trekken: “*Piet heeft een vader (genaamd Jan).*”.

Allereerst: Een vervoeging van het werkwoord “*hebben*” noemen we hieronder een **bezittelijk werkwoord**, omdat het aangeeft dat de generalisatie iets bezit.

De regels voor een **bezittelijke omkeer-conclusie** zijn:

- 1) In de zin moet zowel de generalisatie als de relatie een eigennaam zijn. Bij het trekken van de conclusie moeten deze verwisseld worden;
- 2) De zin moet een vervoeging van het werkwoord “*zijn*” te hebben, welke in de conclusie vervangen moet worden door een vervoeging van het bezittelijk werkwoord “*hebben*”;
- 3) De zin moet een (enkelvoudige) specificatie met zelfstandig naamwoord bevatten. Als in de specificatie een bepaald lidwoord wordt gebruikt, dient het in de conclusie vervangen te worden door een onbepaald lidwoord.

Nu toegepast op de bovenstaande zin:

- 1) De generalisatie “*Jan*” en de relatie “*Piet*” zijn eigennamen en worden verwisseld;
- 2) Het werkwoord “*is*” wordt vervangen door het bezittelijk werkwoord “*heeft*”;
- 3) De (enkelvoudige) specificatie “*de vader*” bevat een zelfstandig naamwoord. Het bepaalde lidwoord “*de*” wordt vervangen door het onbepaalde lidwoord “*een*”.

Het resultaat is dan: “*Piet heeft een vader (genaamd Jan).*”.

2.4. Het doen van aannames

Naast conclusies, kunnen we ook aannames doen op basis van de beschikbare informatie, al staat de geldigheid van de aannames nog niet vast. Hieronder wordt uitgelegd hoe aannames te automatiseren zijn.

2.4.1. Generaliseerde aannames

Beschouw eerst deze zin: “Een *ouder* is een vader *of* (een) moeder.”.

Door het koppelwoord “*of*” kunnen de eerste zin ontleden in twee enkelvoudige definities:

- “Een vader is een *ouder*.”
- “Een moeder is een *ouder*.”

Beschouw nu de zinnen:

- “Een *vader* is een *ouder*.”
- “Jan is een *vader*.” of “Jan is de *vader* van Piet.”

We zien nu dat het specificatie-woord van de zin over “Jan” gelijk is aan het generalisatie-woord van de definitie-zin. We mogen hier dus een specificatie-substitutie conclusie trekken.

Dus uit de zinnen “Een *ouder* is een *vader of* (een) moeder.” en “Jan is de *vader* van Piet.” mogen we dus de conclusie trekken: “Jan is een *ouder* (van Piet).”.

Deze 'conclusie' noemen we een **generaliserende aanname**, omdat de zelfstandig-naamwoord-specificatie van een zin vervangen mag worden door de generalisatie van een exclusieve specificatie-verzameling van een (oorspronkelijke) definitie-zin.

Een generalisatie kent vaak uitzonderingen, vandaar dat het een aanname is en geen conclusie.

2.4.2. Bezittelijke voorwaardelijke-specificatie-aannames

Beschouw de zinnen:

- “Een gezin *heeft ouders en kinderen.*”
- “Jan is een *ouder van Piet.*”

Twee opmerkingen over de eerste zin:

- 1) Deze zin heeft een vervoeging van het bezittelijk werkwoord “*hebben*”;
- 2) Het voegwoord “*en*” geeft aan dat **beide** specificaties nodig zijn om de definitie geldig te maken (voorwaarde).

En twee opmerkingen over de laatste zin:

- 1) Deze relatie-specificatie kunnen we lezen als: “Jan” heeft een “*ouder*”-relatie met “*Piet*”;
- 2) Maar de relatie in de omgekeerde richting, van “*Piet*” naar “*Jan*”, is onbekend, waardoor we hier geen conclusies mogen trekken over “*Piet*”, ook al zouden we graag willen beweren:
 - “*Piet is een kind van Jan.*”
 - “*Jan heeft een kind (genaamd Piet).*”

Toch willen we iets met deze informatie doen, omdat het een schat aan informatie kan opleveren als de bovengenoemde aannames over “*Piet*” waar zijn. En als deze aannames onjuist blijken te zijn, is er in ieder geval duidelijkheid geschapen in de situatie en kan de informatie worden aangescherpt, iets waar we ook naar streven.

We mogen **bezittelijke voorwaardelijke-specificatie-aannames** doen:

- 1) Als er sprake is van een vervoeging van het bezittelijk werkwoord “*hebben*”;
- 2) Als er sprake is van een **voorwaardelijke** specificatie-verzameling;
- 3) Omdat vanuit “*kinderen*” gezien, een relatie met “*ouders*” momenteel de enig mogelijke relatie in de omgekeerde richting is;
- 4) Als er later andere relaties mogelijk blijken te zijn, zullen de betreffende aannames getoetst moeten worden;
- 5) Als we alle conclusies, die we uit een aanname kunnen trekken, ook weer als aanname aanmerken, totdat die eerste aanname bevestigd of verworpen is. Dan dienen alle aannames, die op basis van die eerste aanname zijn gedaan, herzien te worden;
- 6) Als een aanname wordt bevestigd, krijgt het de status van bewering of conclusie;
- 7) Als op basis van een samengestelde specificatie-substitutie een conclusie getrokken kan worden, noemen we dit geen aanname maar een vraag.

2.5. Het zelfstandig stellen van vragen

Als het systeem informatie mist, is het mogelijk om het systeem automatisch vragen te laten stellen aan de gebruiker. Ook strijdige informatie kan worden aangetoond en d.m.v. het stellen van vragen worden gecorrigeerd.

Hieronder wordt een methode uitgewerkt om het aanscherpen van informatie te stimuleren.

2.5.1. Vragen om het aanscherpen van informatie te stimuleren

Al eerder zagen we de conclusie: *“Piet is een zoon of (een) dochter (van Jan).”*.

Maar zo'n conclusie is een beetje vaag, vanwege de onbeantwoorde keuze, aangegeven door het voegwoord *“of”*. Om te stimuleren dat deze informatie aangescherpt wordt, kunnen we zo'n conclusie ook als een vraag presenteren. Dat doen we door de vervoeging van het werkwoord *“zijn”* vooraan de zin te plaatsen en de zin te eindigen met een vraagteken.

Het resultaat is dan: *“Is Piet een zoon of (een) dochter (van Jan)?”*.

2.6. Het vinden van oorzakelijke verbanden

Het is mogelijk om in een tekst automatisch naar oorzakelijke verbanden te zoeken, waardoor het systeem conclusies kan trekken op basis van de reeds bekende informatie. Dit wordt later uitgewerkt.

3. Het opbouwen van de kennisstructuur

Iemand met programmeer-ervaring heeft in het voorgaande wellicht al enkele basisprincipes van programmeertalen ontdekt: Een **assignment** in de vorm van een toewijzing, een **if-then-else** structuur in de vorm van een selectie en misschien ook een **declaratie van een variabele** in de vorm van een generalisatie/specificatie-definitie: Je geeft de structuur aan van de variabele, maar je wijst die variabele nog geen waarde toe.

Met deze overeenkomsten tussen natuurlijke talen en programmeertalen kan een kennisstructuur worden opgebouwd. Hierdoor ontstaat een koppeling tussen natuurlijke taal en programmeertalen en is het in principe mogelijk om te programmeren in een natuurlijke taal, wat de basis kan worden van een ‘denkende’ computer.

Hieronder wordt per onderdeel uitgelegd hoe een kennisstructuur opgebouwd kan worden a.d.h.v. zinnen in natuurlijke taal.

3.1. Het creëren van een generalisatie/specificatie

Een belangrijk basiselement van de kennisstructuur is de generalisatie/specificatie. Hieronder worden de diverse aspecten van deze structuur beschreven en hoe daarmee de kennisstructuur wordt opgebouwd.

3.1.1. Het creëren van de nieuwe woorden

Neem de zin: “*Jan is een vader.*”.

Het systeem moet eerst de woorden “*Jan*” en “*vader*” creëren, voordat de kennisstructuur opgebouwd kan worden. Dus als deze woorden nog niet bestaan in het systeem, of als ze niet van het juiste grammaticale type zijn, moeten deze woorden eerst gecreëerd worden.

Jan

Het woord “*Jan*” begint met een hoofdletter en is het eerste woord van de zin. Het kan dus van het grammaticale type *eigenaam* zijn of van een onbekend grammaticale type, als het oorspronkelijke woord niet met een hoofdletter zou beginnen.

In dit geval worden er twee woorden gecreëerd:

- “*Jan*” van het grammaticale type *eigenaam*;
- “*jan*” (zonder hoofdletter) van een onbekend letterkundig type.

Tijdens het opbouwen van de kennisstructuur wordt maar één van beide woorden gebruikt. Daarna wordt het ongebruikte woord verwijderd, zodat het systeem niet bevuild raakt. Hierover later meer.

vader

Het woord “*vader*” wordt vooraf gegaan door een lidwoord, dus hier is het grammaticale type duidelijk: een zelfstandig naamwoord.

In de zin “*Een glas is vol, halfvol of leeg.*” is het niet duidelijk van welke grammaticale type de woorden “*vol*”, “*halfvol*” en “*leeg*” zijn. Wel wordt ervan uit gegaan dat ze van hetzelfde grammaticale type zijn.

3.1.2. Het creëren van een specificatie-verbinding

Nu het systeem de woorden kent, kunnen die woorden met elkaar verbonden worden. Beter gezegd: Het generalisatie-woord wordt verbonden met het specificatie-woord.

Voorbeeld: “*Jan is een vader.*” en “*Jan is een bakker.*”.

Bij de eerste zin wordt vanaf het generalisatie-woord “*Jan*” een **specificatie-verbinding** gelegd naar het specificatie-woord “*vader*” en in de tweede zin van “*Jan*” naar “*bakker*”.

In de kennisstructuur is nu bekend dat Jan zowel een vader is als een bakker.

3.1.2.1. Het creëren van een relatie-verbinding

Om de relatie-specificatie “*Jan is een vriend van Mark.*” toe te voegen aan de kennisstructuur, wordt generalisatie-woord “*Jan*” d.m.v. een specificatie-verbinding verbonden met specificatie “*vriend*” en met daarin opgenomen een verwijzing naar relatie “*Mark*”.

Er wordt dus een specificatie-verbinding gebruikt met een tweede verwijzing erin. Deze speciale specificatie-verbinding wordt een **relatie-specificatie-verbinding** of een **relatie-verbinding** genoemd.

Die relatie-verbinding moet gelezen worden als: “*Jan*” heeft een “*vriend*”-relatie met “*Mark*”. Hetzelfde geldt voor “*Bush*” en “*Obama*” die een “*president*”-relatie met de “*Verenigde Staten*” hebben.

Bij de zin “*Jan is een vriend van Mark en Paula.*” worden twee relatie-verbindingen gecreëerd vanaf “*Jan*”: Beide met specificatie “*vriend*”, maar één met een relatie-verwijzing naar “*Mark*” en één met “*Paula.*”.

3.1.3. Het creëren van een verzameling

Neem de zin: “*Een glas is vol, halfvol of leeg.*”.

De **specificatie-woorden** van generalisatie “*glas*” worden ‘verzameld’ door ze wederzijds met verzamel-verbindingen aan elkaar te verbinden:

- “*vol*” wordt met een oplopende verzamelingsverbinding met “*halfvol*” verbonden;
- “*halfvol*” wordt met een oplopende verzamelingsverbinding met “*leeg*” verbonden;
- “*leeg*” wordt met een aflopende verzamelingsverbinding met “*halfvol*” verbonden;
- “*halfvol*” wordt met een aflopende verzamelingsverbinding met “*vol*” verbonden.

Het ‘oplopen’ of ‘aflopen’ van een verzamelingsverbinding geeft de volgorde van de opsomming aan.

Het verzamelen van generalisaties en relaties en gebeurt op dezelfde manier:

Het verzamelen van generalisatie-woorden

- “*Bush is de vorige president van de Verenigde Staten.*”
- “*Obama is de huidige president van de Verenigde Staten.*”

Het verzamelen van relatie-woorden

- “*Jan is de vader van Piet en Marie.*”

3.1.4. Het creëren van een toewijzing

Een toewijzing is een generalisatie/specificatie met een toegewezen waarde. Daarom wordt eerst een generalisatie/specificatie-structuur gecreëerd, als deze nog niet bestaat, en daarna de toewijzingsstructuur.

3.1.4.1. Het creëren van een specificatie-toewijzing

Een specificatie-toewijzing met een specificatie-verzameling kan maar één van de toestanden tegelijk weergeven, als het voegwoord “of” in de zin wordt gebruikt.

Neem nu de zin “Een glas is vol, halfvol of leeg.” en de huidige toestand (toewijzing): “Het glas is halfvol.”.

Als nu het glas wordt leeggedronken, geldt de nieuwe toestand: “Het glas is leeg.”. Dan wordt de toewijzingsverbinding van “glas” naar “halfvol” deaktiveert, waardoor deze toewijzing als grammaticale verleden tijd geïnterpreteerd wordt. Daarna wordt een toewijzingsverbinding van “glas” naar “leeg” gecreëerd, om vast te leggen dat de toestand van het glas nu “leeg” is.

Meerdere toewijzingen per generalisatie

Een specificatie-toewijzing kan meerdere toewijzingen hebben. Zo kunnen beide toewijzingen tegelijk gelden: “Het glas is leeg.” en “Het glas is blauw.”.

Een toewijzing met meerdere relaties

Bij het creëren van een specificatie-toewijzing met meerdere relaties, wordt voor elke relatie een aparte specificatie-toewijzing gecreëerd.

Twee opmerkingen:

- In de zin “Jan is de vader van Piet en Marie.” geeft het voegwoord “en” aan, dat beide relaties aan elkaar zijn verbonden en dus tegelijk gelden;
- Ook als je daarna zegt: “Jan is de bakker van Paul.”, blijven de bovenstaande toewijzingen actief, omdat “vader” en “bakker” deze geen verzameling vormen.

3.1.4.2. Het creëren van een generalisatie-toewijzing

Ook generalisatie-toewijzingen kunnen maar één toestand tegelijk weergeven, omdat ze door een generalisatie-verzameling aan elkaar zijn verbonden:

Als eerst de toewijzing geldt “Bush is de president van de Verenigde Staten.” en na de verkiezingen geldt “Obama is de president van de Verenigde Staten.”, dan wordt de toewijzing van “Bush” naar “de president van de Verenigde Staten” gedeactiveerd, waardoor deze toewijzing als grammaticale verleden tijd geïnterpreteerd wordt, en daarna wordt er een toewijzing van “Obama” naar “de president van de Verenigde Staten” aangemaakt.

3.2. Het creëren van een selectie

Net als de generalisatie/specificatie is ook de selectie een belangrijk basiselement van de kennisstructuur.

Om een kennisstructuur van een selectie op te bouwen, worden eerst alle generalisatie/specificatie-structuren aangemaakt, waaruit die selectie is opgebouwd. Vervolgens wordt de conditie in een [conditie-lijst](#) opgeslagen, de actie in een [actie-lijst](#) en de alternatieve-actie in een [alternatieve-actie-lijst](#).

Neem de zin: *“Als het verkeerslicht geel of rood is, moet je stoppen, anders moet je doorrijden.”*.

Eerst worden dus de generalisatie/specificaties *“Het verkeerslicht is geel of rood.”*, *“Je moet stoppen.”* en *“Je moet doorrijden.”* aangemaakt. Vervolgens wordt *“Het verkeerslicht is geel of rood.”* in de lijst van condities geplaatst met een verwijzing naar de bijbehorende actie en de alternatieve-actie. *“Je moet stoppen.”* wordt in de actie-lijst geplaatst en *“Je moet doorrijden.”* wordt in de alternatieve-actie-lijst geplaatst.

3.2.1. Het uitvoeren van selecties

Nadat het systeem een zin gelezen en verwerkt heeft, gaat het systeem alle condities in de kennisstructuur na en voert eventueel de bijbehorende actie of alternatieve-actie uit. Dit proces wordt herhaald totdat er geen verandering meer in het systeem plaatsvindt. Het systeem is dan ‘in rust’; het is gereed om de volgende zin te lezen en te verwerken.

Hierdoor lijkt het een beetje alsof het systeem ‘denkt’, maar eigenlijk wordt er gewoon een programma uitgevoerd dat in natuurlijke taal geschreven is.

4. Dialogen

In het voorgaande hebben we zinnen in natuurlijke taal omgezet naar een kennisstructuur. Maar om een dialoog te kunnen voeren met de gebruiker, moet het systeem ook kunnen antwoorden in leesbare zinnen. Daarvoor moet de opgebouwde kennisstructuur weer naar natuurlijke taal worden omgezet.

4.1. De grammatica

Voor zowel het lezen als het antwoorden in een bepaalde taal, is het nodig dat het systeem de grammatica van die taal kent. Voor elke gewenste taal moet dus vooraf de grammatica gedefinieerd worden.

Om het systeem flexibel te houden, wordt deze grammatica als een tekstbestand opgeslagen. Tijdens het opstarten van het systeem worden één of meerdere grammatica-bestanden ingelezen.

Het is ook mogelijk om dialogen te voeren in meerdere talen en uiteindelijk moet het ook mogelijk worden om het systeem automatisch te laten vertalen, maar daarover later meer.

4.1.1. Het lezen van een zin

De grammatica wordt bij het lezen van een zin gebruikt:

om te controleren of de zin voldoet aan de gedefinieerde grammatica;

- om onderscheid te kunnen maken tussen de verschillende structuren, zoals generalisatie/specificaties, toewijzingen en selecties;
- om bepaalde woorden vooraf te definiëren, maar daar later meer over.

4.1.1.1. Alleen acceptatie van bekende zinsstructuren

In het voorgaande hebben we gezien dat er voor het omzetten van zinnen naar een kennisstructuur, per zinsstructuur (generalisatie/specificaties, toewijzingen en selecties) een concept nodig is, om de informatie uit die zinnen te vertalen naar de kennisstructuur. Ook hebben we gezien dat deze zinsstructuren worden vastgelegd in een grammatica-bestand.

Andersom geredeneerd: Het systeem kan dus alleen zinsstructuren accepteren die zijn vastgelegd in het grammatica-bestand, omdat het alleen voor die zinsstructuren een concept heeft om de informatie uit die zinnen om te zetten naar de kennisstructuur. Dit is een beperking die het systeem heeft, want eigenlijk we willen alle aangeboden zinsstructuren kunnen inlezen en de informatie daaruit kunnen omzetten naar onze kennisstructuur.

Het uiteindelijke doel van het project is dan ook om voor zoveel mogelijk zinsstructuren een concept te bedenken, die de informatie uit die bepaalde zinsstructuur vertaalt naar de kennisstructuur.

Het zal dus nog erg veel tijd en moeite kosten om dit doel te bereiken, maar dan is het ook mogelijk om informatie automatisch te laten verwerken en komt de ‘denkende’ computer in zicht.

4.1.2. Het antwoorden met een zin

De grammatica wordt door het systeem gebruikt om met de woorden een grammaticaal-correcte zin te creëren, waarna deze aan de gebruiker gepresenteerd kan worden.

4.1.2.1. Het zelfstandig stellen van vragen

Het systeem kan hiaten en tegenstellingen in de kennisstructuur ontdekken, maar hierover later meer.

Daarnaast kan het systeem ook zelfstandig vragen stellen aan de gebruiker over de gevonden hiaten en tegenstellingen, om zo de opgeslagen informatie aan te vullen, te corrigeren of te verduidelijken. Dit lijkt op intelligent gedrag en draagt bij aan het gevoel dat computers kunnen denken.

5. Toepassingen

Als de voorgaande theorie voldoende doorontwikkeld wordt, zouden computers tekstuele informatie, zoals teksten op internet, documenten van bedrijven, schoolboeken en Technisch Ontwerpen van software-ontwikkelaars, kunnen inlezen en ‘begrijpen’.

Hieronder volgt een aantal toekomstige mogelijkheden waarvoor dit systeem uiteindelijk kan worden ingezet als de grammatica van het systeem voldoende zinsstructuren aan kan.

5.1. Programmeren in natuurlijke taal

Door het selectie-concept is het mogelijk om met het systeem automatisch logische opdrachten in natuurlijke taal uit te laten voeren. Hierdoor kun je programmeren in natuurlijke taal. Zo is het bijvoorbeeld mogelijk om de spelregels van een spel in te lezen, waarna je het spel tegen de computer kunt spelen, zonder het spel vooraf te hoeven programmeren.

Software-ontwikkeling

Een serieuzer voorbeeld is om een Technisch Ontwerp uit de software-ontwikkeling in te lezen en tot op een bepaald niveau uit te voeren, zonder dat er geprogrammeerd hoeft te worden. Het zou minimaal als een soort prototyping-systeem kunnen dienen, maar in principe kan hierdoor ook de software-ontwikkeling op zich geautomatiseerd worden.

Het is dus eigenlijk het automatiseren van de automatisering.

5.2. Het beoordelen en corrigeren van teksten en specificaties

Het systeem kan hiaten en tegenstellingen in teksten vinden, ook die hiaten en tegenstellingen waar een mens soms niet snel aan zou denken. Het systeem kan ze benoemen en eventueel ook oplossen door specifieke vragen te stellen. Op den duur zou het daarmee mogelijk moeten zijn om het systeem teksten en specificaties te laten beoordelen en corrigeren.

5.3. Het geautomatiseerd maken van samenvattingen

Het maken van een samenvatting is in principe niets anders dan het scheiden van hoofdzaken en bijzaken, waarbij steeds meer bijzaken worden weggelaten, net zolang tot de tekst de gewenste grootte heeft bereikt.

Als een samenvatting van een document automatisch gemaakt kan worden, is dat vooral handig voor managers die met lijvige documenten een vergadering in gaan. Ze kunnen dan op eenvoudige wijze een samenvatting van een gewenste grootte maken, om zo toch voldoende geïnformeerd te zijn over de inhoud van de documenten.

5.4. Vertaalprogramma dat de context begrijpt

De meeste vertaalprogramma's vertalen woord voor woord en herkennen geen uitdrukkingen. Omdat woorden vaak meerdere betekenissen hebben en de vertaalprogramma's de context daarvan niet begrijpen, is een automatische vertaling vaak van slechte kwaliteit.

Dit systeem kan in principe de context 'begrijpen', waardoor het mogelijk moet zijn om het geautomatiseerd vertalen naar een hoger niveau te tillen.

5.5. Een zoekmachine die vragen kan beantwoorden

Momenteel doorzoeken we het internet nog steeds door zelf intelligent steekwoorden te kiezen, waarvan we hopen dat die woorden relevante webpagina's opleveren en op die manier hopen we onze vragen beantwoord te krijgen.

Het is op die manier relatief gemakkelijk de betekenis van termen en moeilijke woorden te vinden, maar het wordt moeilijker als je een vraag beantwoord wilt hebben waar expertise voor nodig is, waar je de termen niet van weet of waarbij het moeilijk is om de juiste context aan te geven.

En in welke taal probeer je de informatie te vinden? Waarschijnlijk zul je Engelstalige woorden uitproberen. Maar wat doe je als je de juiste vertaling van de woorden niet weet of als de informatie die je zoekt bijvoorbeeld alleen in het Chinees beschikbaar is? Hoe krijg je dan je antwoord op je vraag, als je die taal niet beheerst?

Alleen als een zoekmachine de beschikbare informatie 'begrijpt' en in de juiste context ziet, is het mogelijk om vragen te stellen aan die zoekmachine.

5.6. Digitale docent of zelfstudiehulp

Een docent bepaalt aan het begin van het schooljaar, d.m.v. de studieboeken en het aantal beschikbare lessen, welke hoofdstukken en onderwerpen dat jaar zullen worden behandeld. Elke les geeft de docent een samenvatting aan de leerlingen, over de te bestuderen stof. Bovendien kan een docent vragen over die stof beantwoorden, doordat hij/zij de stof tot in detail beheerst.

Als het lukt om zowel automatisch samenvattingen te maken (van de te behandelen stof) én om zoekmachines vragen te laten beantwoorden, zou het ook mogelijk moeten zijn om het doceren van een theoretisch vak te automatiseren of in ieder geval om een digitale docent als zelfstudiehulp te gebruiken.