CONSTRAINT LANGUAGES

Rob J.M. Hartog

Binnen de informatica worden steeds meer 'talen' of 'programmeeromgevingen' ontwikkeld ten behoeve van die gebruiker die zich niet bezig wenst te houden met het ontwikkelen van algorithmen. Deze trend wordt vaak beschreven als een verschuiving van procedureel naar declaratief programmeren. De gebruiker die declaratief programmeert geeft idealiter slechts aan wat hij/zij weet c.q. wenst, en niet hoe het informatiesysteem de betreffende kennis dient te gebruiken c.q. de gewenste informatie dient op te leveren. Een voorbeeld van deze verschuiving vormt de opkomst van zogenaamde constraint languages. Dit artikel is bedoeld als eerste kennismaking met constraint languages.

Inleiding

De term 'constraint language' is minstens 20 jaar oud en begrippen als 'constraints' en 'constraint satisfaction' zijn zeker nog ouder. Goede Nederlandse termen zouden zijn: 'beperkingstalen', 'beperkingen' en 'het voldoen aan beperkingen', maar vooralsnog zijn deze termen niet ingeburgerd. Daarom zal ik mij aansluiten bij de Nederlandse gewoonte om Amerikaanse termen onvertaald te laten (v.g. joy-stick en disk-drive i.p.v. pretstok en schijf draaier).

Nu er sinds kort ook constraint languages zijn die het research stadium bijna geheel verlaten hebben en voor produktiedoeleinden gebruikt kunnen worden zoals b.v. CHARME (Opledu '89) mogen we ook in Nederland een toenemende belangstelling voor deze talen verwachten. Vandaar deze inleiding tot constraint languages.

Dit artikel is slechts bedoeld om de lezer een eerste conceptueel model aan te bieden. Voor formele definities van constraints en verwante begrippen wordt de lezer verwezen naar Steele, Guesgen, Leler (1980).

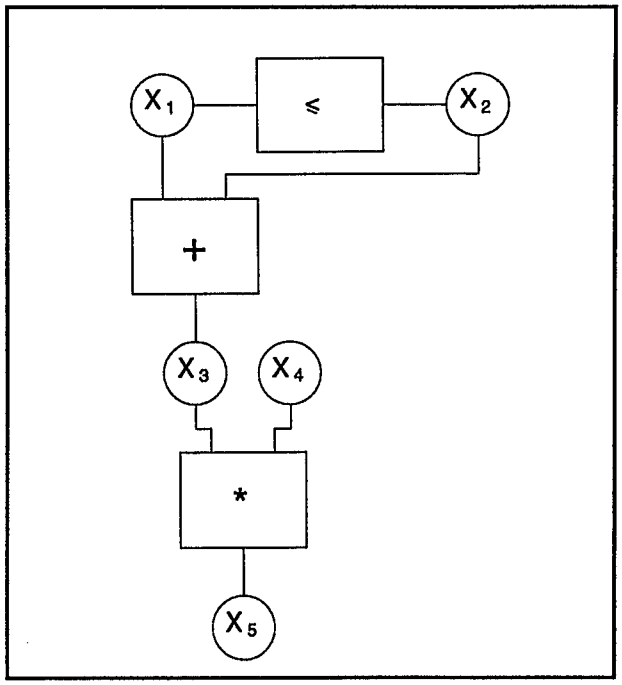
Constraint languages worden hier vooral geintroduceerd als een voorbeeld van een ontwikkeling in de richting van programmeertalen die de gebruiker ontlasten van de noodzaak om algorithmen te bedenken. Er zijn overigens nog andere invalshoeken van waaruit constraint languages besproken kunnen worden, men denke hierbij bijvoorbeeld aan het programmeren van parallele machines.

Tot slot zou dit artikel ook een eerste opstap moeten vormen voor lezers die zich verder in constraint languages willen verdiepen. Wie meer wil weten van constraint languages kan het beste hoofdstuk 3 uit Winston (1984) lezen en dan de eerste drie hoofdstukken uit Leler)1988) en vervolgens, afhankelijk van het interessegebied een selektie maken uit de overige literatuur.

Een voorbeeld van een constraint netwerk

Een programma in een constraint language kan gevisualiseerd worden als een netwerk dat opgebouwd is uit constraints en variabelen. De constraint language biedt de gebruiker een aantal 'voor gebakken' constrainttypen (primitieven) aan. Afhankelijk van het doel van de constraint language zal de constraint language een bepaalde verzameling primitieven bevatten.

In figuur 1 is een voorbeeld van zo'n constraint netwerk gegeven.

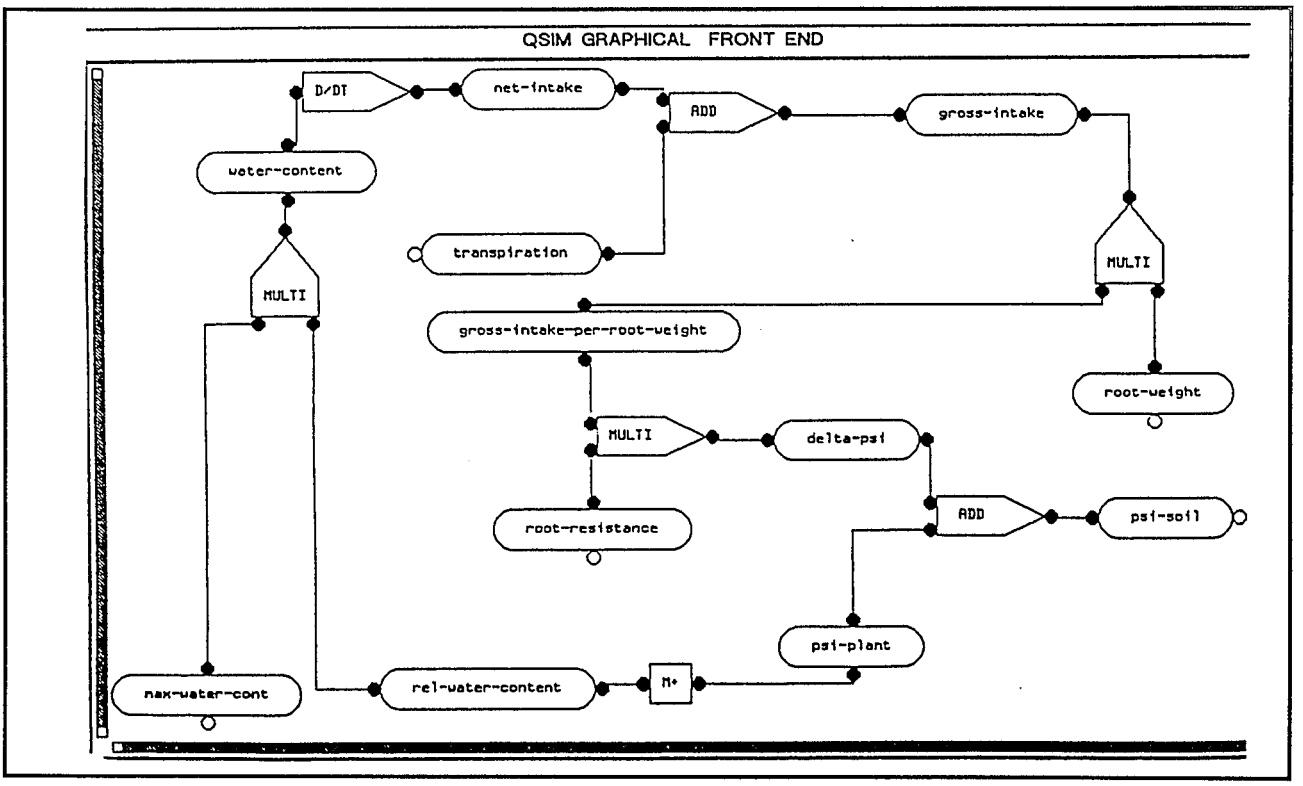


De constraints in dit netwerk zijn instantiaties van drie verschillende constrainttypen.

(In verband met het niet-formele karakter van dit artikel zal verder de term constraint zowel gebruikt worden voor de instantiatie van een constrainttype als voor een constrainttype zelf, zoals vaak de term entiteit gebruikt wordt voor entiteittype maar ook voor entiteit.)

Bekijken we nu de ≤ constraint in het netwerk in figuur 1 dan zien we dat deze een beperking oplegt aan de toegelaten waarden van x1 en x2 . x1 en x2 worden wel de terminals van de constraint genoemd. De initiele waardenverzamelingen waarop een variabele gedefiniëerd is noemen we het domein van die variabele.

Veronderstel dat het domein van x1 bestaat uit de gehele getallen 81 t/m 100 en het domein van x2 uit de gehele getallen 1 t/m 100 dan kan alleen voldaan worden aan de ≤ constraint als op de rechter terminal slechts een deel van het domein van x2 aangeboden wordt. Door uit het domein van x2 de gehele getallen 1 t/m 80 te 'schrappen' en door vervolgens uit de produktverzameling van de resulterende domeinen van x1 en x2 alleen die paren (x1,x2) te selecteren waarvoor x1 ≤ x2 gaat de toestand van de ≤ constraint over van 'not-satisfied' naar 'satisfied'. Het hele netwerk is 'satisfied' als elke constraint in de toestand 'satisfied' is.



Veronderstel b.v. dat het domein van x3 bestaat uit de gehele getallen 1 t/m 365, het domein van x4 uit de gehele getallen 7 en 8 en het domein van x5 uit de gehele getallen 1 t/m 1770. Dan zijn er in eerste instantie:

20 * 80 * 365 * 2 * 1770 mogelijke combinaties van waarden van x1, x2, x3, x4, x5 (m.a.w. de produktverzameling van de domeinen bevat 20673600 elementen). Na constraint satisfaction zijn alleen die combinaties van waarden over waarvoor alle constraints in het netwerk 'satisfied' zijn. O.a. impliceert dit in het voorbeeld van figuur 1 dat er grote delen van de initieel opgegeven domeinen zijn geschrapt. Zo is o.a. direct in te zien dat voor x5 alleen die gehele getallen over blijven die deelbaar zijn door 7 of door 8.

Nogmaals zij benadrukt dat degene die het constraint netwerk opbouwt zich in principe niet bezig houdt met de manier waarop de constraints van de toestand 'not-satisfied' naar de toestand 'satisfied' gebracht worden. De constraint satisfaction algorithmen die ingebouwd zijn in de constraint language zijn idealiter niet zichtbaar voor de gebruiker. Soms zal het constraint satisfaction algorithme niet anders doen dan systematisch de produktverzameling van de betrokken domeinen doorzoeken op waardencombinaties die voldoen aan het constraint netwerk, soms zullen bepaalde constraints gecombineerd worden ('constraint rewriting') soms zullen andere technieken gebruikt worden.

Voor het conceptuele model dat de gebruiker nodig heeft maakt dit niets uit: de gebruiker kan tegen het systeem aankijken alsof dit systematisch op een 'domme' manier alle mogelijke waardencombinaties test en al of niet weggooit.

Het ontbreken van toekenningsopdrachten

Het declaratieve karakter van 'constraint programming' wordt vaak uitgelegd aan de hand van het volgende voorbeeld:

Zij C = (F - 32) * 5/9 de wiskundige vergelijking die de relatie aangeeft tussen de temperatuur uitgedrukt in graden Celcius (C) en de temperatuur uitgedrukt in graden Fahrenheit (F).

Wie nu een imperatieve taal gebruikt om een programma te schrijven dat, gegeven een der variabelen, de andere variabele moet uitrekenen zal zowel de toekenningsopdracht C := (F - 32) * 5/9 als de toekenningsopdracht F := 32 + 9/5 * C moeten opnemen in dat programma. De constraint programmeur neemt echter alleen de *relatie* C = (F - 32) * 5/9 in het programma op. Er zijn dus geen toekenningsopdrachten en bovendien hoeft de constraint programmeur vergelijkingen niet 'om te schrijven'. Toevoegen van de relatie K = C -273 (de relatie tussen de temperatuur uitgedrukt in kelvin en de temperatuur uitgedrukt in graden Fahrenheit) in het constraint programma correspondeert met het toevoegen van vier (!) toekenningsopdrachten in het imperatieve programma.

Een voorbeeld met kwalitatieve constraints

In figuur 2 is een voorbeeld gegeven van een heel ander constraint netwerk. De constraints in dit netwerk zijn instantiaties van kwalitatieve constrainttypen. Het netwerk vormt een kwalitatief groeimodel voor een plant. De domeinen van de variabelen bestaan uit paren van kwalitatieve waarden. Voorbeelden van zulke paren zijn: {'positief','stijgend'} en {'nul','stijgend'}.

De MULTI-constraint die gross-intake-per-root-weight, root- resistance en delta-psi (= het verschil in waterpotentiaal tussen bodem en plant) verbindt staat b.v. niet toe dat de bruto wateropname per wortelgewicht toeneemt als de wortelweerstand positief is en het verschil in

waterpotentiaal tussen bodem en plant afneemt. Het constraintnetwerk is geimplementeerd in QSIM een kwalitatief simulatieprogramma (Kuipers, 1986) Uit het voorbeeld zal duidelijk zijn dat met name de aan de gebruiker aangeboden primitieven als ook bepaalde aspecten van het constraint satisfaction algorithme anders van aard zijn dan bij het eerste voorbeeld.

Na constraint satisfaction van een kwalitatief constraintnetwerk zoals in figuur 2 geven de overblijvende kwalitatieve waarden aan wat het kwalitatieve gedrag van de plant zal zijn. QSIM doet in feite méér want het genereert ook mogelijke nieuwe waarden voor de variabelen; d.w.z. QSIM breidt de opgegeven domeinen uit met zinvolle waarden die in eerste instantie niet door de modelbouwer zijn opgegeven. Hoewel QSIM niet zonder meer een constraint language genoemd mag worden is programmeren in QSIM wel een vorm van constraint programming.

Constraint programming vergeleken met andere vormen van declaratief programmeren

Zoals hierboven al is aangegeven levert informatica onderzoek steeds meer 'talen' c.q. ontwikkelomgevingen op die declaratief programmeren ondersteunen. In deze paragraaf zal de klasse der constraint languages nader getypeerd worden door de verschillen met een aantal bekende declaratieve 'talen' c.q. omgevingen aan te stippen.

Spreadsheet programma's

Leler (1988, pp. 86) classificeert spreadsheet programma's (Multiplan, Excel, Lotus 1-2-3 etc). niet onder de constraint languages omdat veranderingen in de bekende waarde van variabelen in een formule slechts in één richting van cel naar cel gepropageerd zouden worden door het onderliggende constraint satisfaction mechanisme (Mij ontbrak overigens de tijd om na te gaan of dit nog steeds waar is voor alle laatste versies van de bekende spreadsheet programma's). Afgezien hiervan geeft het spreadsheet programma een goed beeld van 'constraint like programming': in een spreadsheet toetst de gebruiker een formule in waar een imperatieve taal als PASCAL een aantal assignment statements vereist. De spreadsheet gebruiker hoeft zich doorgaans niet bezig te houden met een of andere vorm van algorithmisch denken.

Computer algebra systemen

'Computer algebra systemen' is de in Nederland gangbare term voor 'symbolic algebra systems' zoals MACSYMA. Dit zijn in feite expertsystemen die stelsels vergelijkingen en integralen oplossen op de manier waarop een wiskundige (expert) dat zou doen, dat wil zeggen door symbolische manipulatie (dit i.t.t. de numerieke methoden die doorgaans gebruikt worden in simulatietalen en in constraint languages). Aangezien vergelijkingen constraints genoemd moeten worden en aangezien de gebruiker alleen de constraints, de variabelen, de domeinen en eventueel randvoorwaarden opgeeft aan het systeem is het vanuit de optiek van de gebruiker verdedigbaar om deze 'symbolic algebra systems' op één lijn met constraint languages te beschouwen. In de wetenschappelijke literatuur worden computer algebra systemen echter doorgaans niet als constraint languages behandeld.

PROLOG

De op resolutie/unificatie gebaseerde algorithmen die aan PROLOG interpreters ten grondslag liggen kunnen beschouwd worden als constraint satisfaction algorithmen. Een belangrijke verschil tussen talen uit de PROLOG familie en constraint languages is dat in een constraint language constraint satisfac tion kennis is opgenomen, en wel kennis die minder algemeen geldig is dan het resolutie principe. Men denke aan constraint satisfaction van eenvoudige vergelijkingen.

Er is een trend waar te nemen van uitbreiding van PROLOG (Colmerauer '90) waardoor het onderscheid met constraint languages steeds meer lijkt te gaan vervagen.

Vraagtalen

Een voorbeeld van een vraagtaal is SQL. Vraagtalen zijn bedoeld om de gebruiker te laten specificeren WAT hij/zij uit een database wil hebben zonder dat de gebruiker zich met het HOE (i.c. navigatie) hoeft bezig te houden. Als zodanig zijn vraagtalen een voorbeeld van de eerder gesignaleerde verschuiving van procedureel denken naar declaratief denken. Omdat vraagtalen ontwikkeld worden in een heel andere wetenschappelijke context dan constraint languages, een context waar vaak een ander taalgebruik gehanteerd wordt, gaat op het eerste gezicht het verband tussen vraagtalen en constraint languages niet veel verder dan de overeenkomst in declaratieve benadering. In feite is het verband tussen vraagtalen zoals SQL en constraint languages veel fundamenteler. Hierop kan echter in het kader van dit artikel niet nader worden in gegaan.

Kanttekeningen

Overigens zijn binnen de bestaande talen de idealen die eraan ten grondslag hebben gelegen nog niet verwezenlijkt. Vooralsnog is het niet mogelijk om zinvol gebruik te maken van PROLOG, SQL ... zonder dat men zich met de procedurele aspecten bezig hoeft houden, en het is zelfs de vraag of dat ooit mogelijk zal zijn.

Samenvatting en Afsluiting

Een belangrijk aspect van constraint languages is dat algorithmische kennis die al eerder door wiskundigen en informatici ontwikkeld is niet opnieuw door de gebruiker ontwikkeld of opgezocht hoeft te worden om vervolgens te worden vertaald naar een imperatieve taal zoals PASCAL.

De voordelen van constraint languages moeten dan ook gezocht worden in de tijdsbesparing die ze voor de gebruiker opleveren. De ontwikkeling van constraint languages is ook een voorbeeld van arbeidsverdeling: generieke algorithmische kennis wordt eerst door wiskundigen en informatici in de constraint language onder gebracht en vervolgens voegt de gebruiker er zijn/haar meer contextgebonden kennis aan toe.

De ontwikkeling van constraint languages moet gezien worden als een van de sporen waarlangs de verschuiving van procedureel denken naar declaratief denken zich vol-

trekt. Deze verschuiving komt ook tot uiting in de ontwikkeling van talen uit de PROLOG familie, de ontwikkeling van vraagtalen, de ontwikkeling van spreadsheetprogramma's en computer algebra systemen.

Referenties

Colmerauer A. *PROLOG III* Communications of the ACM vol 33 (1990)nr 7 pp 69-90

Guesgen H., Hertzberg J. *Local Propagation in Networks of filtering Constraints* TEX-B MEMO 16-87 GMD St Augustin 1987

Guesgen H.W. *CONSAT, Foundations of a system for constraint satisfaction.* TEX-B Bericht No 14 GMD St Augustin 1987

deKleer J., Sussman G.J. *Propagation of Constraints applied to circuit synthesis.* Circuit Theory and Applications, vol 8, (1980) pp 127-144.

Kuipers B. *Qualitative Simulation,* Artificial Intelligence 29 (1986) pp 289-338

Leler Wm *Constraint Programming Languages, their specification and generation,* Addison-Wesley 1988

Opledu A. , Markovich J., Tourbier Y., *CHARME: un language industriele de programmation par constraintes* Procs 9th international workshop on expert systems and their applications, Avignon 1989.

Sussman G.J., Steele G.L. jr, *CONSTRAINTS a language for expressing almost-hierarchical descriptions.* Artificial Intelligence 14 (1980), pp 1-39

Steele G.L. jr. *The definition and implementation of a computer programming language based on constraints.* dissertation, MIT, 1980 Mass.

Winston P.H. *Artificial Intelligence,* Addison-Wesley 1984 ❑

---

drs Rob Hartog is universitair docent bestuurlijke informatievoorziening bij de vakgroep Informatica van de Landbouwunversiteit te Wageningen, tel. 08370-83408.

BEDRIJFSBEZOEK TFDL

Op donderdag 14 maart a.s. is door VIAS een bedrijfsbezoek georganiseerd bij de stichting Technische en Fysische Dienst voor de Landbouw. Het doel van dit bezoek is de leden op de hoogte te brengen van de activiteiten die plaats vinden bij de TFDL op het gebied van de automatisering.
Dit bedrijfsbezoek was reeds gepland voor 14 december van het vorige jaar. Het tijdstip van dit bezoek bleek achteraf ongunstig. Er waren te weinig aanmeldingen om het bezoek te laten doorgaan. U wordt om deze reden dan ook met klem verzocht uzelf vooraf op te geven voor een bedrijfsbezoek. De gastorganisatie kan zo ook rekening houden met het aantal bezoekers.

De presentaties zullen plaats vinden in de Veluwezaal van het Centrum Techniek, aan de Mansholtlaan te Wageningen.

Voorlopig programma:

- 13.30 Opening door dhr. A.M.K van Beek, directeur TFDL
- 13.35 Instrumentatie-ontwikkeling voor het landbouwkundig onderzoek, dhr. F.W.H. Kampers, hoofd afdeling Instrumentatie
- 14:00 Vochtmeetsysteem voor kunstmatige substraten, J. Balendonck
- 14:30 Pauze
- 15:00 De afdeling Automatisering, R.A.M. van Lopik, hoofd afdeling Automatisering
- 15:30 AgroNet Fase III, J. van Keulen, groep datacommunicatie, afdeling Automatisering
- 16:00 De haalbaarheid van een kennissysteem voor heidebeheer, R.J.B. Zwanikken, groep ECIT, afdeling Automatisering
- 16:30 Sluiting

Voor dit bezoek kunt u zich tot uiterlijk 8 maart opgeven bij mevrouw B. van den Born-van Rees, telefoon 08370-76675.