Johan Jeuring
Faculteit Informatica en Departement Informatica
Open Universiteit Universiteit Utrecht
Postbus 2960 Postbus 80.089
6401 DL Heerlen 3508 TB Utrecht
J.T.Jeuring@uu.nl

Bastiaan Heeren
Faculteit Informatica
Open Universiteit
Postbus 2960
6401 DL Heerlen
Bastiaan.Heeren@ou.nl

Onderwijs

Feedback genereren in leeromgevingen voor algebra

Het gebruik van leeromgevingen bij het wiskundeonderwijs in algebra is de afgelopen jaren toegenomen. Het geven van feedback, bijvoorbeeld door een diagnose van een stap die een leerling heeft gezet te stellen, een hint te geven, of een volledige uitwerking van een opgave te geven, is binnen veel van deze omgevingen niet mogelijk, of zeer arbeidsintensief om te specificeren. In dit artikel presenteren Johan Jeuring en Bastiaan Heeren het Ideas-raamwerk, waarmee leeromgevingen automatisch feedback kunnen geven. Voor het geven van feedback wordt gebruik gemaakt van fundamentele concepten uit de wiskunde en informatica, zoals herschrijfstappen, views of normaalvormen en herschrijfstrategieën. Door het probleem van het geven van feedback te transformeren naar het ontleden van gebruikersstappen aan de hand van een herschrijfstrategie, wordt gebruik gemaakt van kennis over ontleden om automatisch feedback te geven.

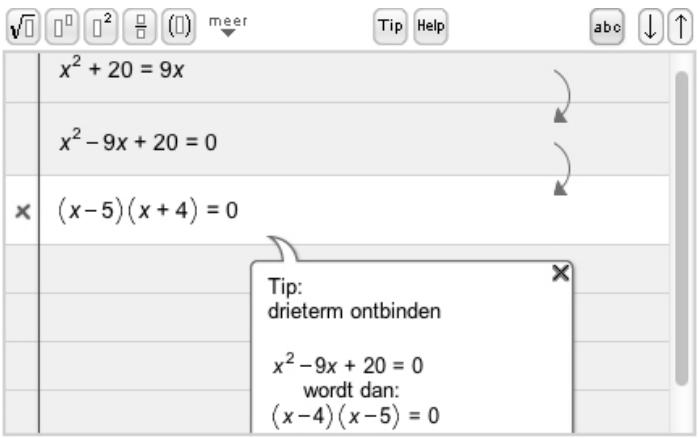
Het gebruik van leeromgevingen bij het wiskundeonderwijs in algebra is de afgelopen jaren toegenomen. De methodes *Getal & Ruimte* en *Moderne Wiskunde* maken gebruik van de Digitale Wiskunde Omgeving (DWO) van het Freudenthal Instituut [5], MathDox [3] wordt gebruikt in de eerstejaars wiskundevakken en bij de aansluitingsvakken voor de masterprogramma's op de technische universiteiten, en de Open Universiteit en verschillende andere Europese universiteiten maken gebruik van Math-Bridge [9]. Deze drie leeromgevingen worden zowel in het voortgezet onderwijs als in het hoger onderwijs gebruikt en bieden theorie, voorbeelden, interactieve opgaven en toetsen aan. In gedrukte vorm zou de inhoud van deze leeromgevingen duizenden pagina's beslaan.

Elektronische leeromgevingen bieden een leerling de mogelijkheid om onderwijsmateriaal te selecteren dat bij hem of haar past, en om zelfstandig te oefenen met wiskundeopgaven, op een willekeurig tijdstip. Leeromgevingen kunnen direct feedback geven op het werk van leerlingen, en hun vorderingen bijhouden. Hoewel het niet eenvoudig is om leeromgevingen en de feedback die zij geven zo op te zetten dat het leren wordt bevorderd, is er door de jaren heen het nodige bewijs geleverd dat leeromgevingen het leren van algebra effectief ondersteunen [4].

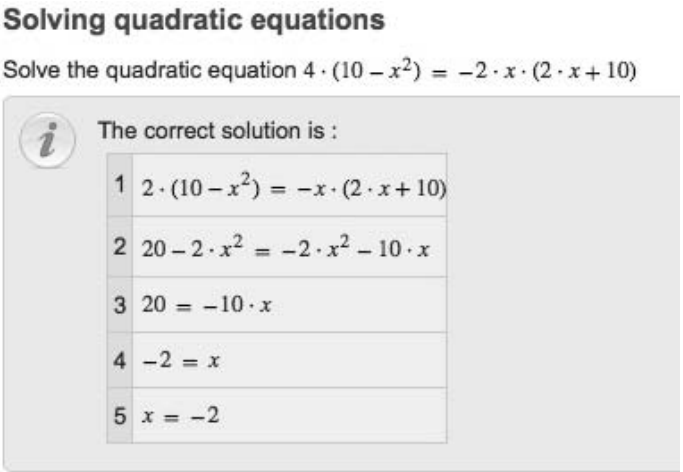
Alle bovengenoemde wiskundeleeromgevingen bieden opgaven aan waarin een leerling een expressie met variabelen, breuken, haakjes en/of polynomen moet vereenvoudigen, of een lineaire, kwadratische, wortel, logaritmische, exponentiële, gebroken of polynomiale (on)gelijkheid op moet lossen. Zeker op het niveau van het voortgezet onderwijs zijn deze opgaven relatief eenvoudig: er bestaan beslissingsprocedures waarmee de opgaven opgelost kunnen worden. Een oplossing vergt meestal slechts een aantal stappen en het aantal verschillende manieren om een opgave op te lossen is vaak beperkt. Hoewel het aantal manieren om dit soort opgaven op te lossen beperkt is, zijn er toch nog talloze mogelijkheden om verschillende oplossingen te construeren, mede door allerlei vereenvoudigen expliciet of impliciet of in een andere volgorde uit te voeren. In een experiment onder negen docenten kregen we zeven verschillende uitwerkingen voor het oplossen van de lineaire vergelijking $3(4x-1)+3=7x-14$.

Mede door de variatie in volgorde en vereenvoudigingen is het geven van goede feedback in een leeromgeving een lastig probleem. De drie hierboven genoemde leeromgevingen gaven tot voor kort alleen goed/fout feedback, of vereisten de volledige specificatie van welke feedback waar en wanneer moet worden gegeven in een opgave. Het geven van goed/fout feedback helpt leerlingen bij het oplossen van opgaven, maar vertelt bij een fout niet *wat* er fout is of wat een goede volgende stap is. Voor het expliciet beschrijven van feedback in een opgave die een leerling in meerdere stappen oplost, zijn in deze leeromgevingen tientallen tot honderden regels code nodig. Gegeven dat de leeromgevingen duizenden opgaven aanbieden, betekent dit dat er alleen al voor de feedback honderdduizenden regels code nodig zijn. Niet alleen vergt het ontwikkelen van opgaven dan zeer veel werk, het wijzigen en verbeteren van opgaven wordt nagenoeg onmogelijk, omdat iedere verbetering op tientallen tot honderden verschillende plaatsen uitgevoerd moet worden.

De afgelopen vijf jaar hebben we het Ideas-raamwerk voor het geven van feedback in leeromgevingen ontwikkeld [7] (http://ideas.cs.uu.nl). De bovenstaande leeromgevingen gebruiken het Ideas-raamwerk om feedback te geven aan een leerling die een opgave in meerdere stappen oplost. In Figuur 1 staat een voorbeeld van het gebruik van het Ideas-raamwerk in de DWO. Een leerling kan op ieder punt in de opgave vragen om een 'Tip', 'drieterm ontbinden' in dit voorbeeld, of om 'Help', '$x^2-9x+20=0$ wordt dan: $(x-4)(x-5)=0$' in dit voorbeeld. De tips en de help worden automatisch berekend uit de *strategie* voor het oplossen van kwadratische vergelijkingen. De docent hoeft alleen maar de opgave te specificeren en aan te geven dat de opgave opgelost moet worden met behulp van de strategie voor kwadratische vergelijkingen. Alle tips en help worden dan automa-



Figuur 1 Gebruik van IDEAS-raamwerk in de DWO



Figuur 2 Gebruik van IDEAS-raamwerk in Math-Bridge

tisch berekend wanneer de leerling daar om vraagt.

In Figuur 2 staat een voorbeeld van het gebruik van het IDEAS-raamwerk in Math-Bridge. Ook de uitwerking wordt automatisch berekend uit de strategie voor het oplossen van kwadratische vergelijkingen. Math-Bridge kan voor veel opgaven voorbeelduitwerkingen laten zien.

In dit artikel beschrijven we hoe leeromgevingen gebruik maken van het IDEAS-raamwerk, en hoe het IDEAS-raamwerk feedback geeft. We zullen met name de achterliggende concepten van het IDEAS-raamwerk introduceren. We beginnen met een aantal voorbeelden van interacties van leerlingen met een leeromgeving. Daarna beschrijven we hoe leeromgevingen gebruik maken van het IDEAS-raamwerk en welke *services* het raamwerk aanbiedt. We beschrijven de eisen die we stellen aan onze *domain reasoners*, de centrale componenten in ons raamwerk, en introduceren herschrijfregels voor het beschrijven van de stappen die een leerling kan uitvoeren. Vervolgens kunt u lezen hoe we omgaan met vereenvoudigingsstappen die een leerling impliciet of expliciet uitvoert en hoe we de verschillende componenten voor het oplossen van een opgave combineren in een strategie. Tot slot beschrijven we hoe andere leeromgevingen het probleem van het geven van feedback oplossen en geven we onze conclusies.

Interacties in leeromgevingen

In deze paragraaf geven we een aantal voorbeelden van interacties in een leeromgeving zoals de DWO. We relateren de voorbeelden aan de concepten die we later zullen bespreken.

Een leerling lost de lineaire vergelijking $6x - 2 = 2x + 14$ op. Als de leerling niet weet hoe zij een lineaire vergelijking op moet lossen kan ze om een uitwerking vragen. Ze krijgt dan:

$$\begin{aligned} & 6x - 2 = 2x + 14 \\ \Rightarrow\ & \{\text{trek } 2x \text{ van beide zijden af}\} \\ & 4x - 2 = 14 \\ \Rightarrow\ & \{\text{tel } 2 \text{ bij beide zijden op}\} \\ & 4x = 16 \\ \Rightarrow\ & \{\text{deel beide zijden door } 4\} \\ & x = 4 \end{aligned}$$

Deze uitwerking, die bestaat uit het toepassen van een drietal regels in een bepaalde volgorde, wordt berekend uit de *strategie* voor lineaire vergelijkingen, zie de paragraaf 'Strategieën' verderop.

Als een leerling niet een hele uitwerking wil zien, maar wel wil weten wat de volgende stap is, dan kan ze om een hint vragen. Ook de hint wordt uit de strategie berekend en bestaat uit een enkele regel. Stel bijvoorbeeld dat de leerling de opgave heeft herschreven naar $6x - 2x = 14 + 2$, dan is de hint: combineer gelijkvormige termen.

In een leeromgeving willen we ook stappen van leerlingen analyseren. Als een leerling een stap maakt die de strategie verwacht, dan wordt dit gerapporteerd. Veelvoorkomende fouten worden ook gerapporteerd, maar kunnen uiteraard niet uit een strategie worden berekend, omdat een strategie alleen maar goede oplossingen berekent. Iedere strategie is voorzien van een verzameling regels die veelvoorkomende fouten representeren. Deze regels worden geprobeerd als een leerling een fout maakt. Als de leerling in de eerste stap $6x - 2 = 2x + 14$ herschrijft naar $8x - 2 = 14$, dan reageert de leeromgeving met: je hebt rechts $2x$ afgetrokken en links $2x$ bijgeteld. Regels en buggy regels (veelvoorkomende fouten) komen verderop in de paragraaf 'Herschrijfregels' aan bod.

We laten niet alle regels die worden toegepast om een vergelijking op te lossen zien. In de bovenstaande uitwerking hebben we bijvoorbeeld de stappen

$$\begin{aligned} & 4x - 2 = 14 \\ \Rightarrow\ & \{\text{tel } 2 \text{ bij beide zijden op}\} \\ & 4x - 2 + 2 = 14 + 2 \\ \Rightarrow\ & \{\text{optellen van constanten}\} \\ & 4x + 0 = 14 + 2 \\ \Rightarrow\ & \{0 \text{ is neutraal element van } +\} \\ & 4x = 14 + 2 \\ \Rightarrow\ & \{\text{optellen van constanten}\} \\ & 4x = 16 \end{aligned}$$

als één stap laten zien. In een strategie configureren we welke stappen we wel en niet laten zien met behulp van zogenaamde *views*, zie verderop. Hoewel we bovenstaande tussenstappen niet willen laten zien, mag een leerling die stappen uiteraard wel maken. Het effect van een *view* is dus afhankelijk van of we een stap laten zien of willen herkennen in het werk van een leerling.

Feedbackservices

Het IDEAS-raamwerk levert feedbackservices aan externe leeromgevingen. Het raamwerk bestaat uit een collectie van zogenaamde *webservices*. Een leeromgeving stuurt een *request* naar het raamwerk, waarin het bijvoorbeeld vraagt om een *hint*, een *diagnose* of een *uitwerking*. Bijvoorbeeld:

- Geef een *hint* voor de vergelijking $x^2 - 9x + 20 = 0$ die opgelost moet worden met de strategie voor kwadratische vergelijkingen. De *hint*-service geeft dan als antwoord: gebruik de regel `nice-factors` van de `algebra.equations.quadratic` opgaven. De DWO vertaalt deze regel naar tekst in 'drieterm ontbinden'.
- Geef een *diagnose* van de vergelijking $4x = 16$ gegeven de vorige vergelijking $6x = 16 - 2x$, en dat de strategie voor het oplossen van lineaire vergelijkingen met de balans-methode wordt gebruikt. De *diagnose*-service geeft de buggy regel `buggy.addbal10` van de `algebra.equations.linear.balance` opgaven terug, die de DWO vertaalt naar 'Je telt er rechts $2x$ bij op, maar links trek je $2x$ er vanaf'.
- Geef een *uitwerking* van de opgave $4(10 - x^2) = -2x(2x+10)$ die opgelost moet worden met de strategie voor kwadratische vergelijkingen. De *derivation*-service geeft als resultaat de lijst van vergelijkingen die gepresenteerd wordt in de screenshot van de Math-Bridge omgeving.

Naast *hint*, *diagnose* en *uitwerking* biedt het IDEAS-raamwerk nog een tiental services aan, variërend van *klaar* om te bepalen of een opgave opgelost is, tot *voorbeelden*, waarmee tientallen voorbeelden van de verschillende soorten opgaven worden verkregen.

Domain reasoners

Een *domain reasoner* genereert hints en uitwerkingen en analyseert stappen van leerlingen voor een specifiek domein. We hebben *domain reasoners* voor het oplossen van kwadratische vergelijkingen, lineaire vergelijkingen, et cetera. *Domain reasoners* zijn de centrale componenten in ons raamwerk. Een *domain reasoner* moet aan een aantal eisen voldoen.

Een *domain reasoner* kan automatisch hints en uitwerkingen genereren en automatisch stappen van een leerling analyseren. Ze lost een opgave op op de manier zoals de docent of het leerboek beschrijft (het *cognitive fidelity* principe [1]). De componenten waaruit een *domain reasoner* bestaat zijn observeerbaar, aanpasbaar en kunnen gebruikt worden in samenstellingen. Zo kunnen we bijvoorbeeld de strategie voor het oplossen van lineaire vergelijkingen hergebruiken in de strategie voor het oplossen van kwadratische vergelijkingen. Een *domain reasoner* maakt zoveel als mogelijk gebruik van generieke componenten, die voor verschillende soorten opgaven hergebruikt kunnen worden.

De belangrijkste onderdelen van een *domain reasoner* zijn: herschrijfstrategieën voor het beschrijven van hoe termen herschreven kunnen worden naar een oplossing, herschrijfregels die eenvoudige transformatiestappen op termen beschrijven, en views en canonieke vormen die specifieke vormen van termen herkennen en het automatisch vereenvoudigen van termen ondersteunen. In de volgende paragrafen zullen we ieder van deze drie componenten nader beschouwen.

Herschrijfregels

Een leerling lost een opgave op door het toepassen van herschrijfregels. Een mogelijke oplossing voor de lineaire vergelijking $3(4x-1)+3 = 7x - 14$ is:

$$\begin{aligned} & 3(4x-1)+3 = 7x-14 \\ \Rightarrow\ & \{\texttt{distr-times}\} \\ & 12x = 7x - 14 \\ \Rightarrow\ & \{\texttt{var-left met parameter } 7x\} \\ & 5x = -14 \\ \Rightarrow\ & \{\texttt{times met parameter } \tfrac{1}{5}\} \\ & x = -\frac{14}{5} \end{aligned}$$

In deze oplossing worden drie herschrijfregels gebruikt: `distr-times`, vermenigvuldigen distribueert over optellen (of 'haakjes uitwerken', zoals de DWO deze regel beschrijft),

$$a(b+c) \Rightarrow ab + ac;$$

het aftrekken van een expressie aan beide zijden van een gelijkheid met als doel de expressies met variabelen naar links te brengen, `var-left`; en het delen door een expressie aan beide zijden van een gelijkheid. De laatste twee regels nemen de expressie die afgetrokken of gedeeld moet worden als argument.

Een *domain reasoner* kan een regel *toepassen* op een (sub)expressie, als in $3(4x-1) \Rightarrow 12x - 3$, om een *hint* of een *uitwerking* te geven. Ook kan een *domain reasoner* een expressie van een leerling analyseren om te bepalen of en welke regel door een leerling is toegepast. Bijvoorbeeld, is de expressie $12x - 3$ te verkrijgen uit de vorige expressie $3(4x-1)$ door het toepassen van een regel? De *domain reasoner* past dan alle toegestane regels toe op de vorige expressie, en bepaalt of een van de resultaten overeenkomt met de expressie die een leerling heeft ingevoerd.

Sommige regels verwachten een parameter, zoals de regel `times` in het bovenstaande voorbeeld:

$$\forall c: \quad a = b \Rightarrow ac = bc \qquad (\text{mits } c \neq 0).$$

Strikt genomen zijn dit soort regels geen herschrijfregels. Een *domain reasoner* kan zo'n regel alleen maar toepassen als er ergens een specifieke parameter c beschikbaar is, waarmee de regel geïnstantieerd kan worden. Vaak kunnen we voor zo'n regel een functie schrijven die een expressie van een leerling analyseert om te bepalen of een regel die gebruik maakt van een parameter is toegepast.

Een speciale categorie regels vormen de *buggy regels*. Deze regels beschrijven veel voorkomende fouten in oplossingen van leerlingen. Voorbeelden van buggy regels zijn:

$$\begin{aligned} (a+b)^2 &\not\Rightarrow a^2 + b^2, \\ \frac{a}{b} + \frac{c}{d} &\not\Rightarrow \frac{a+c}{b+d}, \\ \forall c: \quad a = b &\not\Rightarrow a + c = b - c. \end{aligned}$$

Iedere klasse van opgaven heeft een verzameling buggy regels die bij het oplossen van de opgave mogelijk door een leerling gebruikt worden. Wanneer de *domain reasoner* een stap van een leerling niet kan verklaren met behulp van een toegestane regel, worden de buggy regels geprobeerd. Als blijkt dat inderdaad een buggy regel is toegepast, dan wordt deze regel gerapporteerd. Het bepalen van wat een buggy regel is, is niet altijd eenvoudig. Bijvoorbeeld, het is mogelijk om honderden buggy regels te verzinnen alleen al voor opgaven over breuken [8]. In onze *domain reasoners* hebben we er voor gekozen om buggy regels door docenten in te laten brengen.

Sommige stappen die een leerling moet zetten hebben geen visueel resultaat. Bijvoorbeeld, bij het oplossen van een kwadratische gelijkheid die mooie factoren heeft, zoals $x^2 + x - 6 = 0$, moet een leerling eerst de factoren bepalen en daarna de expressie factoriseren. Hier is sprake van twee regels: het bepalen van de factoren van een kwadratische expressie en het factoriseren met behulp van de gevonden factoren. Een ander voorbeeld van stappen zonder visueel effect zijn de stappen die een *domain reasoner* intern gebruikt om door een expressieboom te lopen, om herschrijfregels op een bepaalde plaats toe te passen. Zo wordt in het voorbeeld van de kwadratische vergelijking de expressie links van het gelijkteken gefactoriseerd; de expressie rechts wordt ongemoeid gelaten. Het doorlopen van de ex-

pressieboom wordt ook met behulp van regels beschreven. Herschrijfregels zonder visueel effect noemen we *minor* regels. Minor regels spelen een belangrijke rol bij het implementeren van 'gestalt view' [2] gerelateerde regels waarin een leerling een expressie moet analyseren om te bepalen of de expressie aan een voorwaarde voldoet en gebaseerd op die analyse de volgende stap zet.

Views

Binnen de wiskunde zijn er een aantal conventies bij het tonen van afleidingsstappen en expressies aan leerlingen. In deze paragraaf beschrijven we hoe we daar mee omgaan en welke concepten we daarvoor gebruiken.

In de vorige paragraaf wordt een afleiding voor een oplossing van een lineaire vergelijking gegeven. Daarin komt onder andere de stap $3(4x-1)+3 = 7x-14 \Rightarrow 12x = 7x-14$ voor, die `distr-times` wordt genoemd. Maar wat gebeurt er eigenlijk in detail aan de linkerkant van de gelijkheid in deze regel?

$$
\begin{aligned}
& 3(4x-1)+3 \\
\Rightarrow\ & \{\text{definitie van } -\} \\
& 3(4x+(-1))+3 \\
\Rightarrow\ & \{\texttt{distr-times}\} \\
& (3\cdot 4x+3\cdot(-1))+3 \\
\Rightarrow\ & \{\text{associativiteit van } \cdot\} \\
& ((3\cdot 4)x+3\cdot(-1))+3 \\
\Rightarrow\ & \{\text{vermenigvuldigen van constanten}\} \\
& (12x+3\cdot(-1))+3 \\
\Rightarrow\ & \{\text{vermenigvuldigen van constanten}\} \\
& (12x+(-3))+3 \\
\Rightarrow\ & \{\text{associativiteit van } +\} \\
& 12x+((-3)+3) \\
\Rightarrow\ & \{\text{optellen van constanten}\} \\
& 12x+0 \\
\Rightarrow\ & \{0 \text{ is neutraal element van } +\} \\
& 12x
\end{aligned}
$$

Deze stap is dus niet een enkele herschrijfregel, maar bestaat uit een serie van bijna tien herschrijfregels! Een leerling mag in plaats van een enkele herschrijfstap één of meerdere van de bovenstaande tussenstappen gebruiken, maar we willen dat onze *domain reasoner* de tussenstappen niet laat zien als de leerling om een *hint* of een *uitwerking* vraagt. Deze granulariteit in herschrijfstappen kan overigens verschillen voor verschillende leerlingen. Het ligt voor de hand dat gevorderde leerlingen minder tussenstappen zetten dan beginnende leerlingen die voor het eerst met een specifieke categorie opgaven oefenen.

Bij het tonen van expressies aan leerlingen willen we graag veelgebruikte, *canonieke*, vormen gebruiken. Zo willen we liever $a-b$ dan $a+(-b)$ laten zien, en schrijven we $x+0$ liever direct als x. Ook dit is weer afhankelijk van het niveau van de leerlingen.

Tenslotte komen we een vergelijkbaar probleem tegen bij het beschrijven van de herschrijfregels die leerlingen mogen toepassen. We definiëren de regel

$$a(b+c) \Rightarrow ab+ac$$

maar we willen liever niet ook de regels:

$$
\begin{aligned}
a(b-c) &\Rightarrow ab-ac, \\
-a(b+c) &\Rightarrow -ab-ac
\end{aligned}
$$

en andere varianten definiëren.

Voor het berekenen van canonieke vormen, voor het bepalen van de granulariteit van regels, en voor het beperken van het aantal herschrijfregels dat we moeten definiëren gebruiken we zogenaamde *views* [6]. Een *view* bestaat uit twee onderdelen: een *match*-functie die een waarde van het domein als argument neemt, en een waarde in het codomein oplevert, en een *build*-functie die weer terug gaat naar het domein vanuit het codomein. De compositie van de twee functies, *build* $\circ$ *match*, geeft een canonieke vorm. Bijvoorbeeld, de *plusView* neemt een expressie als argument, en probeert de expressie als de som van twee expressies te schrijven. De *plusView* herkent dus voorkomens van + op topniveau in een expressie, of probeert die te creëren:

$$
\begin{aligned}
\textit{match plusView}\ (a-b) &\Rightarrow a+(-b), \\
\textit{match plusView}\ -(a+b) &\Rightarrow -a+(-b).
\end{aligned}
$$

De *build*-functie van de *plusView* voert de inverse acties uit op deze termen. Hieronder staat een voorbeeld van hoe we gebruik maken van *views* in onze afleidingen:

$$
\begin{aligned}
& 3(4x-1) \\
\Rightarrow\ & \{\textit{match plusView} \text{ op } 4x-1\} \\
& 3(4x+(-1)) \\
\Rightarrow\ & \{\texttt{distr-times}\} \\
& 3\cdot 4x+3\cdot(-1) \\
\Rightarrow\ & \{\text{vermenigvuldigen van constanten}\} \\
& 12x+(-3) \\
\Rightarrow\ & \{\textit{build plusView} \text{ op } 12x+(-3)\} \\
& 12x-3
\end{aligned}
$$

Veel herschrijfregels maken gebruik van verschillende *views* om de linkerkant van de regel te matchen, en om de expressie verkregen na toepassing van de regel vervolgens op te schonen. *Views* en geparametriseerde regels lossen het probleem van het expliciet maken van alle herschrijfregels in een oplossing van een opgave op.

Strategieën

Een herschrijfstrategie beschrijft hoe een term herschreven kan worden naar een oplossing. De beginterm is de opgave die de leerling voorgeschoteld krijgt, en de oplossing is de uitdrukking die als antwoord wordt verwacht. We illustreren het concept van herschrijfstrategieën met een aantal strategieën voor het oplossen van een kwadratische vergelijking. De beginterm van een opgave waarin een leerling een kwadratische vergelijking op moet lossen is een vergelijking waarin mogelijk aan beide zijden kwadratische expressies voorkomen. Een oplossing bestaat uit twee of minder lineaire vergelijkingen van de vorm $x = c$, waar c staat voor een willekeurige constante.

Een naïeve strategie voor het oplossen van een kwadratische vergelijking staat het toepassen van willekeurige herschrijfstappen op expressies en vergelijkingen toe totdat de vergelijking van de goede vorm is. Zo'n strategie keurt elke correcte herschrijfstap goed, zelfs al brengt de stap een leerling verder van een goede oplossing. Als een leerling om een *hint* vraagt, dan zullen er in het algemeen veel stappen mogelijk zijn in deze strategie. Als er meerdere stappen mogelijk zijn raadpleegt de *hint*-service een *ordening* op regels die voor een opgave wordt gespecificeerd. Het is echter zo goed als onmogelijk een ordening te specificeren die ervoor zorgt dat de gegeven hints niet al te vreemd zijn. Een naïeve strategie biedt weinig ondersteuning aan een leerling om het doel op de gewenste manier te bereiken.

Een meer algoritmische strategie voor het oplossen van een kwadratische vergelijking bestaat uit het eerst herschrijven van de vergelijking naar de vorm $ax^2+bx+c=0$, en dan het toepassen van de abc-formule. Deze strategie biedt meer ondersteuning voor het bereiken van een oplossing. De meeste wiskundedocenten verwachten echter meer

```
quadraticStrategyG =
  label "Quadratic Equation Strategy" $ repeatS $
    -- Relaxed strategy: even if there are "nice" factors,
    -- allow use of quadratic formula
    somewhere (generalForm <|> generalABCForm)
     |> somewhere zeroForm
     |> somewhere constantForm
     |> simplifyForm
     |> topForm
  where
      -- ax^2 + bx + c == 0, without quadratic formula
    generalForm = label "general form" ...

    generalABCForm = ...

    ...
```

Figuur 3

van hun leerlingen: een leerling moet bijvoorbeeld ook een kwadratische vergelijking op kunnen lossen door een term met mooie factoren te factoriseren en de zo verkregen lineaire vergelijkingen op te lossen.

De 'expert'-strategie voor het oplossen van een kwadratische vergelijking bestaat uit de volgende stappen:

1. Analyseer de vorm van de vergelijking.
2. Herschrijf zo mogelijk de vergelijking in een mooie vorm.
3. Gebruik de volgende substrategie, afhankelijk van de vorm van de vergelijking:
   a. Als er geen lineaire term is: neem de vierkantswortel.
   b. Als er geen constante term is: factoriseer en los de lineaire vergelijkingen op.
   c. Als er mooie factoren zijn: factoriseer en los de lineaire vergelijkingen op.
   d. Gebruik in alle andere gevallen de abc-formule.

Een *domain reasoner* die gebruik maakt van deze strategie kan precies aangeven wat de goede volgende stap is, op een manier die gebruikelijk is in de Nederlandse schoolboeken. Ook de *uitwerkingen* die door de *domain reasoner* worden gegenereerd zullen in het algemeen goedgekeurd worden door Nederlandse wiskundedocenten.

Voor het beschrijven van een strategie hebben we een *strategietaal* ontwikkeld. Onze strategietaal bevat verschillende componenten die nodig zijn om het soort strategieën die hierboven staan te specificeren. Dit zijn:

- *Pas een herschrijfregel toe*, zoals vermenigvuldigen distribueert over optellen.
- *Sequentie*. 'Eerst s dan t' schrijven we als s <*> t.
- *Keuze*. 'Gebruik s of t' schrijven we als s <|> t. Een variant van keuze heeft voorkeur voor het toepassen van het eerste argument: s |> t past s toe indien mogelijk, en anders t.
- *Slaag of faal*. De strategieën *succeed* en *fail* zijn de neutrale elementen van respectievelijk sequentie en keuze.
- *Labels*. We gebruiken labels om posities aan te geven in een strategie. Aan zo'n positie kunnen we een specifieke feedbacktekst koppelen.
- *Recursie*. We gebruiken *fix* voor herhaling, bijvoorbeeld bij het toepassen van stappen totdat geen stap meer mogelijk is.

De algoritmische strategie voor het oplossen van een kwadratische vergelijking wordt bijvoorbeeld gespecificeerd door: *rightzzero* <*> *abc*, waarin *rightzzero* de strategie is voor het herschrijven van een vergelijking zodat de rechterterm gelijk is aan 0 en *abc* de regel is die de *abc*-formule implementeert. Figuur 3 laat een fragment van de strategie voor het oplossen van kwadratische vergelijkingen zien die in gebruik is in de bijbehorende *domain reasoner*. De magercursieve onderdelen zijn componenten van de strategietaal.

Een *domain reasoner* gebruikt een herschrijfstrategie voor het volgen van een oplossing van een leerling en voor het geven van *hint*s en *uitwerking*en. De strategie beschrijft welke sequenties van herschrijfregels zijn toegestaan. We beschouwen een strategie als een contextvrije grammatica, en ontleden de sequentie van herschrijfstappen van een leerling om te controleren of de stappen de strategie volgen. Een *hint* is het eerste symbool (een herschrijfregel) dat is toegestaan volgens de strategie. Een *uitwerking* bestaat uit een correcte zin van de grammatica. Een opgave is *klaar* als de strategie de lege zin accepteert. Op deze manier geven we feedback met behulp van functionaliteit voor ontleden.

Conclusies

Het automatisch geven van feedback en hints in een leeromgeving voor algebra is een uitdagend probleem, dat slechts door weinig leeromgevingen wordt opgelost. Sinds een paar jaar hebben we het Ideas-raamwerk ontwikkeld, waarmee *hint*s, *uitwerking*en en *diagnose*s gegeven kunnen worden. Verschillende leeromgevingen maken nu gebruik van het raamwerk.

Voor het bieden van feedback-services beschrijft het raamwerk wiskundige kennis met behulp van herschrijfregels voor het uitvoeren van eenvoudige herschrijfstappen, views voor het beschrijven van wiskundige normaalvormen en het automatisch uitvoeren van vereenvoudigingen, en herschrijfstrategieën voor het beschrijven hoe een opgave opgelost wordt. Dit artikel introduceert en illustreert deze concepten.

Er zijn een aantal andere leeromgevingen die het probleem van het geven van feedback (gedeeltelijk) oplossen, zoals de Cognitive Tutor van Carnegie Learning. Onze aanpak verschilt van andere leeromgevingen door de expliciete beschrijving van de wiskundige concepten, zodanig dat deze beschrijvingen observeerbaar, herbruikbaar en aanpasbaar worden. In de toekomst hopen wij experimenten met docenten uit te voeren waarin we docenten feedback laten aanpassen.

Referenties

1 M.J. Beeson, Design principles of MathPert: Software to support education in algebra and calculus, in N. Kajler, ed., *Computer-Human Interaction in Symbolic Computation*, Springer, 1998, pp. 89–115.

2 C. Bokhove, Use of ICT for acquiring, practicing and assessing algebraic expertise, PhD thesis, Utrecht University, 2011

3 A. Cohen, H. Cuypers, E. Reinaldo Barreiro en H. Sterk, Interactive mathematical documents on the web, in *Algebra, Geometry and Software Systems*, Springer, 2003, pp. 289–306.

4 A.T. Corbett, K.R. Koedinger en J.R. Anderson, Intelligent Tutoring Systems, in M. Helander, T. K. Landauer, P. Prabhu (eds), *Handbook of Human-Computer Interaction, Second, Completely Revised Edition*, Chapter 37, Elsevier Science, 1997, pp. 849–874.

5 M. Doorman, P. Drijvers, P. Boon, S. van Gisbergen en K. Gravemeijer, Design and implementation of a computer supported learning environment for mathematics, in *Earli 2009 SIG20 invited Symposium Issues in designing and implementing computer supported inquiry learning environments*, 2009.

6 B. Heeren en J. Jeuring, Canonical forms in interactive exercise assistants, in *MKM'09*, Vol. 5625 of *LNCS*, Springer, 2009, pp. 325–340.

7 B. Heeren, J. Jeuringe en A. Gerdes, Specifying rewrite strategies for interactive exercises, *Mathematics in Computer Science* 3(3), 2010, pp. 349–370.

8 M. Hennecke, *Online Diagnose in intelligenten mathematischen Lehr-Lern-Systemen (in German)*, PhD thesis, Hildesheim University, 1999, Fortschritt-Berichte VDI Reihe 10, Informatik / Kommunikationstechnik; 605, Düsseldorf: VDI-Verlag.

9 E. Melis en J. Siekmann, ActiveMath: An intelligent tutoring system for mathematics, in *ICAISC*, Vol. 3070 of *LNCS*, Springer, 2004, pp. 91–101.