Opgaven over bolstapelingen en over de bewijsassistent HOL Light

Freek Wiedijk

Het Hales-Ferguson-bewijs van het vermoeden van Kepler is te ingewikkeld om er met opgaven dieper op in te gaan. Daarom geven we hier alleen opgaven over twee van de stapelingen waarvan het bewijs laat zien dat ze maximale dichtheid hebben, en opgaven over de bewijsassistent HOL Light door middel waarvan de correctheid van het bewijs met de computer is geverifieerd.

Er zijn enkele opgaven die niet echt om een antwoord vragen, maar meer het karakter hebben van 'bekijk dit eens.' Deze opgaven zijn met een sterretje in de kantlijn gemarkeerd.

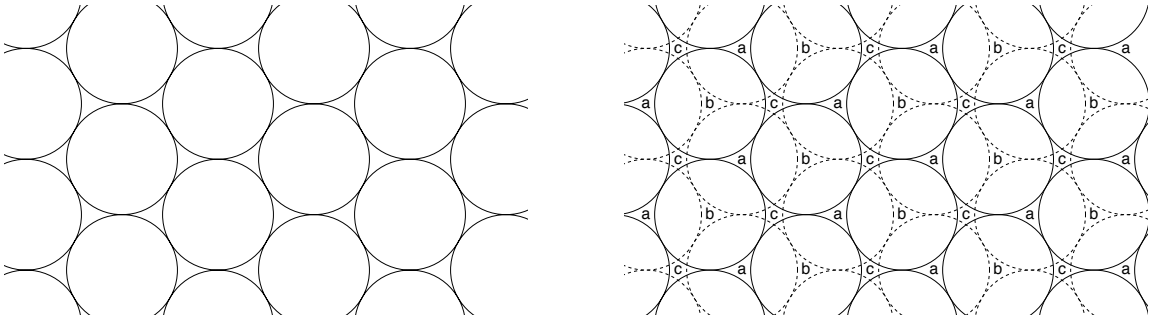
Bolstapelingen

We beschrijven kort twee bolstapelingen die maximale dichtheid hebben: de FCC-stapeling en de HCP-stapeling. FCC staat voor 'Face Centered Cubic' ofwel 'kubisch vlakgecentreerd', en HCP staat voor 'Hexagonal Close-Packed' ofwel 'hexagonale dichtst'. De eerste wordt altijd uitgelegd als hoe een groenteboer sinaasappelen opstapelt, of hoe kanonskogels naast een kanon worden opgestapeld.

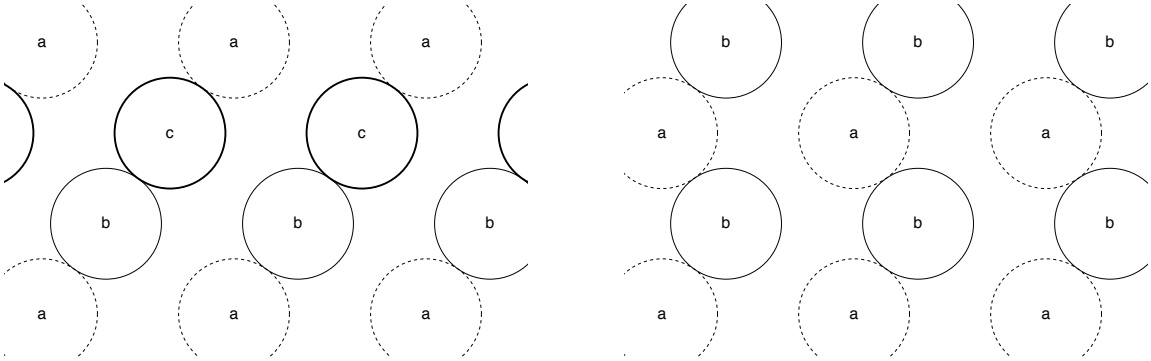
In beide gevallen gaat het om het stapelen van *laagjes* bollen, waarbij in iedere laag de bollen in een honingraatpatroon liggen. Elke keer als er een nieuwe laag op wordt gelegd zijn er *twee* verschillende posities waarin dit kan worden gedaan, zie figuur 1. Als de bollen uit een laag op de posities a liggen, dan kan voor de volgende laag tussen de positie b en de positie c worden gekozen.

Er zijn nu twee voor de hand liggende mogelijkheden, zie figuur 2. Als je bij elke volgende laag dezelfde afstand opschuift, cykel je door de posities a, b en c, en krijg je de FCC-stapeling. Als je 'heen en weer' gaat tussen twee posities, zal de derde positie nooit aan de beurt komen, en krijg je de HCP-stapeling.

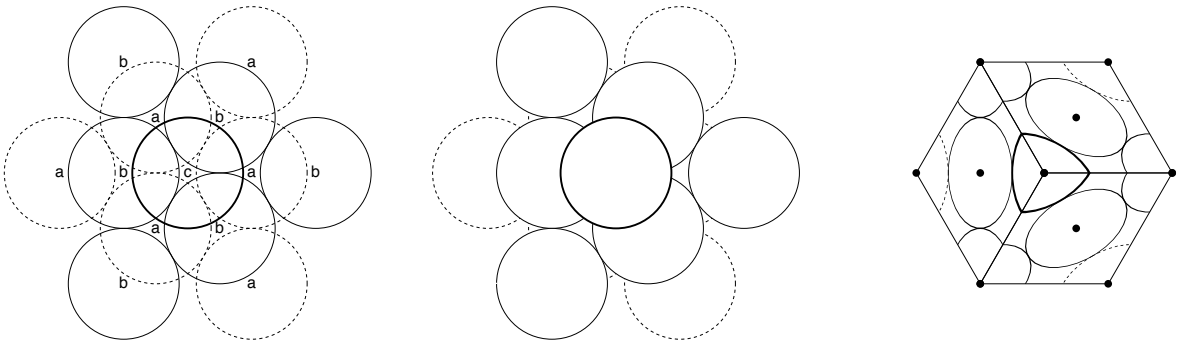
Om te begrijpen waarom de FCC-stapeling 'kubisch vlakgecentreerd' heet, bekijken we veertien bollen uit deze stapeling als in figuur 3. Het blijkt dat er acht op de hoekpunten van een kubus liggen, en zes op het midden van de zes zijvlakken van die kubus. In plaats van een structuur met laagjes, kun je deze stapeling dus ook begrijpen als een structuur van kubussen met bollen op de hoekpunten en middenop de zijvlakken. De lichaamsdiagonaal van deze kubussen staan loodrecht op de vlakken waarmee we de stapeling eerst beschreven.



Figuur 1: Links een laag elkaar rakende bollen in een honingraatpatroon. Rechts een tweede laag in positie 'b' (de oorspronkelijke laag gestippeld daaronder in positie 'a').



Figuur 2: Het verschil tussen de FCC-stapeling en de HCP-stapeling. In beide gevallen een zijaanzicht van een dwarsdoorsnede langs een lijn door de punten a-b-c, waardoor de posities van de lagen zichtbaar worden. Links de FCC-stapeling met de lagen achtereenvolgend cyclisch in posities a, b en c. Rechts de HCP-stapeling met twee posities alternerend. Merk op hoeveel ruimte er in deze doorsnede tussen de bollen zit!



Figuur 3: Veertien bollen van de FCC-stapeling uit vier opeenvolgende lagen. Onderaan een bol in positie c, daarop zes bollen (gestippeld) in een driehoek in positie a, dan zes bollen in een driehoek in positie b, en bovenop weer een bol in positie c. Links in de stijl van het rechterdiagram uit figuur 1. In het midden met de bollen ondoorzichtig getekend, zodat de kubusstructuur duidelijker is. Rechts een kubus waaruit deze stapeling kan worden opgebouwd, en die de naam van de FCC-stapeling verklaart

Opgave 1. Geef het aantal bollen dat een bol raakt in de FCC- en HCP-stapelingen. Dit heet het *kusgetal* van de bol.

Het Newton-Gregory probleem of het *dertienbollenprobleem* is de vraag of een bol in een bolstapeling in drie dimensies kusgetal 13 kan hebben. In 1953 werd door Schütte en van der Waerden voor het eerst correct bewezen dat dit niet mogelijk is. Het maximale kusgetal in drie dimensies is dus 12. Het corresponderende maximale kusgetal in vier dimensies is 24. Dit werd pas voor het eerst door Musin bewezen in 2003.

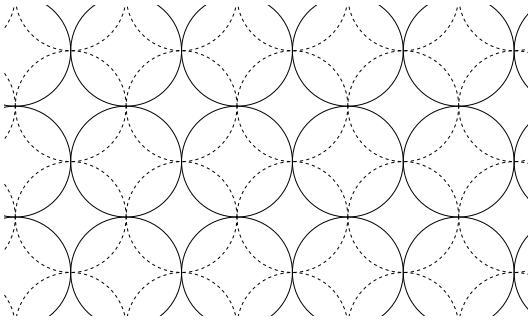
Vanaf nu bekijken we alleen bolstapelingen van bollen met straal $r = 1$, dus als twee bollen aan elkaar raken is de afstand tussen de middelpunten $2r = 2$.

Opgave 2. De middelpunten van de bollen in de FCC-stapeling vormen een *rooster*, d.w.z., ze zijn precies van de vorm $n_1\vec{v}_1+n_2\vec{v}_2+n_3\vec{v}_3$, met $n_1, n_2, n_3 \in \mathbb{Z}$. De vectoren $\vec{v}_1$, $\vec{v}_2$ en $\vec{v}_3$ heten de *basis* van het rooster.

(i) Geef een basis van het rooster van de middelpunten van de bollen in een FCC-stapeling, waarbij de coördinaten zo gekozen zijn dat een laag van de bollen in het x-y vlak ligt.

(ii) Geef vervolgens een basis van dit rooster, waarbij de coördinaten natuurlijk zijn voor de definitie in termen van kubussen.

(iii) Geef de rotatiematrix die het ene coördinatensysteem overvoert in het andere.



Figuur 4: De FCC-stapeling als twee alternerende lagen van bollen waarbij de lagen geen honingraatpatroon volgen, maar uit een patroon van vierkanten bestaan.

(iv) Onder deze rotatie wordt de ene basis niet noodzakelijkerwijs afgebeeld op de andere basis. Geef de (n_1, n_2, n_3) van de beelden van de vectoren uit de ene basis in termen van de andere basis.

Opgave 3. Laat zien dat je de FCC-stapeling ook nog kunt beschrijven met alternerende lagen zoals afgebeeld in figuur 4.

Opgave 4. Geef een definitie van de dichtheid van een (oneindige) bolstapeling. Is deze dichtheid altijd welbepaald?

Opgave 5. Laat zien dat de dichtheid van de bollen in de FCC-stapeling gelijk is aan:

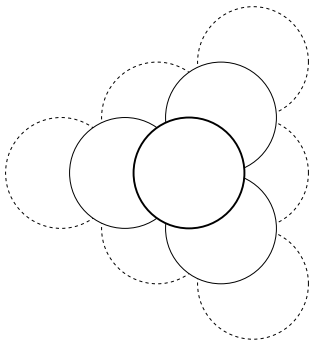
$$\frac{\pi}{\sqrt{18}} = 74{,}0480\ldots\,\%$$

Het volume van een bol met straal r is $\frac{4}{3}\pi r^3$, en het volume van het deel van de ruimte per bol in de FCC-stapeling kan worden uitgerekend als de determinant van de matrix corresponderend met één van de basissen uit de vorige opgave.

Opgave 6. De FCC- en HCP-stapelingen zijn maar twee mogelijkheden voor een bolstapeling met maximale dichtheid. Als we stapelingen die alleen verschillen in oriëntatie in de ruimte als identiek beschouwen, hoeveel verschillende stapelingen met maximale dichtheid zijn er dan? Zijn er ook stapelingen met maximale dichtheid waarin bollen voorkomen met kusgetal lager dan twaalf?

Opgave 7. De *Voronoi-cel* van een punt p uit een verzameling punten V, bestaat uit alle punten in de ruimte die dichter bij p liggen dan bij elk ander element van V. Als de verzameling V de middelpunten van de bollen in één van de twee genoemde bolstapelingen is, dan worden de Voronoi-cellen begrensd door een veelvlak.

Beschrijf de Voronoi-cel van de FCC-stapeling, en die van de HCP-stapeling. Deze heten respectievelijk een *rombische dodecaëder* en *trapezo-rombische dodecaëder*. Geef voor beide Voronoi-cellen de coördinaten van de hoekpunten van het veelvlak dat deze Voronoi-cellen begrenst.



Figuur 5: Een tetraeder van bollen uit de FCC-stapeling, met $n = 3$. Deze tetraeder bevat dus $\binom{3+2}{3} = 10$ bollen.

Opgave 8. Laat zien dat het volume van zowel de Voronoi-cellen van de FCC- als van de HCP-stapelingen gelijk is aan:

$$4\sqrt{2} = 5{,}65685\ldots$$

Opgave 9. Je kunt bollen in een tetraedervorm opstapelen volgens de structuur van de FCC-stapeling. Zie bijvoorbeeld figuur 5 voor zo'n tetraeder met een zijde van drie bollen. In dat geval bestaat die tetraeder uit tien bollen.

(i) Geef een formule voor het algemene geval van het aantal bollen in zo'n tetraeder met een zijde van n bollen.

(ii) Laat zien dat dit aantal gelijk is aan de binomiaalcoëfficient $\binom{n+2}{3}$. Driehoeksgetallen zijn van de vorm $\binom{n+1}{2}$, en tetraedergetallen zijn dus van de vorm $\binom{n+2}{3}$. Op deze manier is aan alle getallen in de driehoek van Pascal een meetkundige interpretatie te geven.

* **Opgave 10.** Bekijk het niet-computer deel van het Hales-Ferguson bewijs in het boek *Dense Sphere Packings*:

`https://github.com/flyspeck/flyspeck/blob/master/downloads/DenseSpherePackings.pdf`

Zie bijvoorbeeld pagina 10 van dit boek voor een plaatje van de Voronoi-cel van de FCC-stapeling.

De bewijsassistent HOL Light

Voor de volgende opgaven is een computer met een internetverbinding nodig. Zowel Windows, macOS als Linux zijn hiervoor geschikt, hoewel in Linux het installeren van HOL Light het makkelijkst gaat.

Een bewijsassistent of *interactieve stellingenbewijzer* (*Interactive Theorem Prover* of ITP-systeem; in tegenstelling tot Automated Theorem Prover of ATP-systeem) is een computerprogramma om met de computer bewijzen in volledig detail te construeren en deze zeer grondig op correctheid te controleren. De naam is enigszins misleidend, want de 'assistentie' betreft voornamelijk de triviale details. *Formal proof editor* zou wellicht een betere naam zijn. Ook lijken deze systemen erg op programmeeromgevingen, en de 'wiskundige' talen waarin de bewijzen worden opgeschreven lijken erg op programmeertalen.

Het installeren van een bewijsassistent bestaat over het algemeen uit het installeren van de vertaler van de programmeertaal waarin het systeem is geschreven, het installeren van het systeem zelf, en het installeren van de wiskundige bibliotheek voor het systeem. Om HOL Light te installeren komt dit neer op:

1. Installeer het OCaml systeem, te vinden op:

   `https://ocaml.org/`

   OCaml is een populaire Franse functionele programmeertaal.

2. Installeer de Campl5 preprocessor voor OCaml, te vinden op:

   `https://camlp5.github.io/`

3. Download het HOL Light systeem, te vinden op:

   `http://www.cl.cam.ac.uk/~jrh13/hol-light/`

   Als `git` is geïnstalleerd gaat dit met het commando:

   ```
   git clone https://github.com/jrh13/hol-light.git
   ```

4. In de HOL Light directory moet de relevante versie van `pa_j.ml` worden vertaald. Onder Linux gaat dit met een eenvoudig:

   ```
   make
   ```

   Op een niet-Linux systeem moet je zelf doen wat aan het begin van `Makefile` staat, namelijk met

   ```
   ocamlc -version
   ```

   uitzoeken wat de versie van OCaml is, dan de relevante file naar `pa_j.ml` kopiëren, en vervolgens het relevante `ocamlc` commando uit de `Makefile` uitvoeren.

5. Tenslotte, om HOL Light te laden, geef je in de HOL Light directory het commando:

```
ocaml
```

en binnen de OCaml interpreter die dan wordt gestart het commando:

```
#use "hol.ml";;
```

om de HOL Light code te laden. Dit duurt eventjes, want de basis-wiskundebibliotheek die dan wordt geladen wordt eerst in zijn geheel op wiskundige correctheid gecontroleerd.

Opgave 11. Installeer HOL Light zoals hierboven beschreven, en laad het systeem. Controleer of het systeem goed functioneert door $1 + 1$ met OCaml uit te rekenen, en de bijbehorende gelijkheid in HOL Light te bewijzen:

```
1 + 1;;
NUM_REDUCE_CONV `1 + 1`;;
```

Merk op dat de aanhalingstekens om HOL Light formules *back*quotes zijn. De toets hiervoor bevindt zich op de meeste toetsenborden ergens aan de linkerkant.

Als HOL Light gestart is, kun je een bewijs gaan doen. HOL Light heeft geen aparte bewijstaal, het is uitsluitend een bibliotheek van OCaml-functies. Dus het 'doen' van een bewijs bestaat uit het aanroepen van OCaml functies uit de HOL Light bibliotheek. Dit kan op twee manieren: als een serie interactieve commando's en als een 'bewijsscript'.

Als voorbeeld hiervan geven we hier een bewijs met inductie dat voor alle $n \geq 0$ geldt dat:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

De inductie bestaat er uit dat je deze formule moet bewijzen voor $n = 0$ (de basisstap), en voor iedere n dat als deze formule geldt voor n, ze ook geldt voor $n + 1$ (de inductiestap).

Als losse commando's wordt dit bewezen door:

```
g `!n. nsum (1..n) (\i. i) = (n*(n + 1)) DIV 2`;;
e INDUCT_TAC;;
e (REWRITE_TAC[NSUM_CLAUSES_NUMSEG]);;
e ARITH_TAC;;
e (REWRITE_TAC[NSUM_CLAUSES_NUMSEG]);;
e (ASM_REWRITE_TAC[]);;
e ARITH_TAC;;
top_thm();;
```

De `g` functie initialiseert een bewijsverplichting of *bewijsdoel*. Vervolgens voert de `e` functie *tactieken* uit die telkens de eerste van de nog overgebleven bewijsverplichtingen terugbrengen tot nul of meer (hopelijk) eenvoudigere doelen. De tactieken die dit bewijs gebruikt zijn:

```
INDUCT_TAC
REWRITE_TAC
ASM_REWRITE_TAC
ARITH_TAC
```

Wat de syntax van de formules betreft, het uitroepteken `!` betekent 'voor alle', en de backslash `\` definieert een functie. Dus `\i. i` is de identiteitsfunctie die met input `i` het getal `i` zelf teruggeeft. Handige HOL Light functies bij het maken van een bewijs zijn voorts:

```
help
search
b
```

De eerste geeft uitleg over een HOL Light functie. Zo geeft

```
help "REWRITE_TAC";;
```

uitleg over de `REWRITE_TAC` tactiek. De tweede is handig om lemma's in de HOL Light bibliotheek te vinden. Zo vond ik het lemma `NSUM_CLAUSES_NUMSEG` door het commando

```
search [‘nsum (x..0)‘];;
```

te geven (iets dergelijks leek me handig voor de basisstap van het inductiebewijs). Dit lemma kan bekeken worden door de naam in te tikken. De input

```
NSUM_CLAUSES_NUMSEG;;
```

geeft als output:

```
val it : thm =
  |- (!m. nsum (m..0) f = (if m = 0 then f 0 else 0)) /\
     (!m n.
          nsum (m..SUC n) f =
          (if m <= SUC n then nsum (m..n) f + f (SUC n) else nsum (m..n) f))
```

Tenslotte gaat

```
b();;
```

één stap in het bewijs terug. Dit is de 'back' ofwel de 'undo' functie.

Opgave 12. Voer bovenstaand voorbeeldbewijs in HOL Light uit, en experimenteer met de verschillende HOL Light functies die hierboven genoemd zijn.

Opgave 13. Gebruik `NUM_REDUCE_CONV` om uit te vinden waarom dit bewijs niet werkt als je `n*(n + 1) DIV 2` schrijft in plaats van `(n*(n + 1)) DIV 2`.

Opgave 14. Bewijs met HOL Light de formule uit opgave 9:

$$\sum_{i=1}^{n} \binom{i+1}{2} = \binom{n+2}{3}$$

Geef eerst het commando

```
loads "Library/binomial.ml";;
```

om de definitie van binomiaalcoëfficiënten te laden. Mogelijk nuttige lemma's voor dit bewijs zijn:

```
GSYM ARITH_SUC
ADD_CLAUSES
NSUM_CLAUSES_NUMSEG
LE_SUC
LE_0
binom
```

Een subtiliteit bij dit bewijs is dat HOL Light niet zelf ziet dat `2` gelijk is aan `SUC (SUC 0)`. Om dit duidelijk te maken moet het bewijsdoel eerst herschreven worden met de eerste twee lemma's.

Behalve als een rij tactieken kan een bewijs ook in scriptvorm worden gegeven. Dit is hoe bewijzen er in de HOL Light-bibliotheek uitzien. Wat daar gebeurt is dat tactieken worden gecombineerd met `THEN` en `THENL`. Het bewijs dat we hier als voorbeeld gebruiken wordt in deze vorm:

```
let TRIANGULAR_SUM = prove
 (`!n. nsum (1..n) (\i. i) = (n*(n + 1)) DIV 2`,
  INDUCT_TAC THENL
  [REWRITE_TAC[NSUM_CLAUSES_NUMSEG] THEN ARITH_TAC;
   REWRITE_TAC[NSUM_CLAUSES_NUMSEG] THEN ASM_REWRITE_TAC[] THEN ARITH_TAC]);;
```

Dit kan nog wat verkort worden door tactieken te combineren tot:

```
let TRIANGULAR_SUM = prove
 (`!n. nsum (1..n) (\i. i) = (n*(n + 1)) DIV 2`,
  INDUCT_TAC THEN ASM_REWRITE_TAC[NSUM_CLAUSES_NUMSEG] THEN ARITH_TAC);;
```

Opgave 15. Voer bovenstaand script in HOL Light uit. Zet ook je bewijs uit opgave 14 in deze vorm om.

Opgave 16. Zoek in de HOL Light bibliotheek de formulering van lemma `VOLUME_BALL` die de formule voor het volume van een bol geeft.

Opgave 17. In het lemma uit de vorige opgave komt de constante `pi` voor. Zoek in de HOL Light-bibliotheek de keten van definities die gebruikt is om deze constante te definiëren.

* **Opgave 18.** Download de Flyspeck-formalisatie van

`https://github.com/flyspeck/flyspeck`

en bekijk de bewijsscripts in de directory `text_formalization`.

Gebruikersvriendelijkere bewijsassistenten

De twee populairste ITP-systemen van het moment zijn de Franse bewijsassistent *Coq*, en de Brits-Duitse bewijsassistent *Isabelle*. De Britse bewijsassistent *HOL Light* waar we in de vorige sectie naar gekeken hebben kent niet zoveel gebruikers, en is ook lastig te leren. (Al is het wel één van de meest interessante computersystemen die er voor wiskunde bestaat.)

Evenwel, mocht je dieper in de bewijsassistenten willen duiken, dan is het dus waarschijnlijk handiger om Coq of Isabelle proberen te leren. Daarom nu nog drie opgaven om een beter beeld te krijgen van wat er aan alternatieven naast HOL Light bestaat:

* **Opgave 19.** Neem een kijkje bij het Coq systeem:

`https://coq.inria.fr/`

Deze webpagina is niet bijzonder aanlokkelijk, maar het Coq systeem heeft desondanks vele gebruikers. Een redelijke introductie is de syllabus *Logical Verification* door Femke van Raamsdonk:

`http://www.cs.ru.nl/~freek/courses/tt-2016/public/notes.pdf`

Het meest indrukwekkende Coq-project tot nog toe is de verificatie van de Feit-Thompson stelling, die zegt dat iedere groep van oneven orde oplosbaar is. Het bewijs van deze stelling kost op papier meer dan tweehonderd pagina's moeilijke groepentheorie. Zie de Mathoverflowdiscussie

`https://mathoverflow.net/questions/164959/`
`how-do-i-verify-the-coq-proof-of-feit-thompson`

over hoe dit bewijs met Coq te controleren.

* **Opgave 20.** Neem een kijkje bij het Isabelle systeem:

`http://isabelle.in.tum.de/`

Een lijst van allerhande formele bewijzen kan gevonden worden in het *Archive of Formal Proofs*:

`https://www.isa-afp.org/`

* **Opgave 21.** Kies een stelling uit een lijst met honderd belangrijke wiskundige resultaten

`http://www.cs.ru.nl/~freek/100/`

en bekijk hoe deze in de bewijstalen van de verschillende bewijsassistenten wordt geformuleerd en bewezen.