Rökfræðimálið Alloy

Inngangur

Alloy er skilgreiningar- og lýsingarmál[1] sem búið var til 1997 við MIT háskólann í Boston. Alloy byggir á umsagnarökfræði og með því má líkja eftir ýmsum kerfum þ.á.m. hugbúnaðarkerfum. Í málinu má skilgreina grunnmengi, föll og umsagnir, og setja fram umsagnarformúlur af fjölbreyttu tagi. Um Alloy er fjallað í kafla 2.7 í kennslubók. Það er að vísu útgáfa 2 sem þar er fjallað um, en nýjasta útgáfan er númer 4 og það hefur sumt breyst (sjá aftast á þessum blöðum).

Mikilvægt verkefni sem leysa má með Alloy er að kanna hvort formúlur séu réttar. Búin er til Alloy-lýsing (eða „forrit“) með einni eða fleiri formúlum. Síðan er svonefndur *Alloy Analyzer* keyrður, og hann athugar hvaða líkön (eins og þau sem lýst er á bls. 124 í kennslubók) passa við formúlurnar. Oft koma óendanlega mörg líkön til greina, en Alloy prófar bara endanlega mörg líkön, þau sem hafa óðal með takmarkaðan staka-fjölda *n*, sem tiltekinn er í forritinu. Ef formúlurnar eru réttar fyrir öll prófuð líkön, og *n* er ekki of lítið, þá má með nokkru (eða stundum miklu) öryggi fullyrða að formúlurnar séu réttar (sbr. bls. 143). Annars finnur Alloy mótdæmi, og a.m.k. ein formúla er sem sé röng. Alloy athugar formúlurnar með því að nota SAT-leysi (sbr. kafla 1.6 í kennslubók).

Einn mikilvægasti möguleikinn í Alloy er að geta teiknað líkönin sem finnast. Góð aðferð til að búa til Alloy lýsingu á kerfi er að byrja með fáar og einfaldar skilgreiningar og formúlur, teikna niðurstöðuna, og bæta svo smám saman við lýsinguna með leiðsögn af teikningum í hverju skrefi. Einfalt dæmi um þennan framgangsmáta er biðraðarlýsing sem gefin er í *A Guide to Alloy* (sjá http://alloy.mit.edu/community/tutorials), og annað dæmi er keðjulistadæmið hér að aftan.

Með Alloy er hægt að líkja eftir bæði stöðugum og kvikum kerfum (*static* og *dynamic*). Hér verður látið duga að fjalla um þau stöðugu. Kennslubókin tekur hins vegar dæmi um kvikt kerfi í kafla 2.7.3.

Á þessum blöðum er helstu atriðum Alloy málsins lýst. Fyrst er sagt frá því hvar finna má forritið og hvernig það er keyrt eftir að búið er að ná í það. Síðan eru tvö einföld dæmi, sem gefa forsmekk af því hvað hægt er að gera. Þá kemur umfjöllun um einstaka þætti málsins, og svo er endað með tveimur dæmum til viðbótar, sem sýna hvernig beita má Alloy á formúlur í umsagnarökfræði.

Uppsetning og keyrsla Alloy

Aðal-Alloy-vefsíðan er http://alloy.mit.edu, og þaðan má komast að forritinu (á http://alloy.mit.edu/alloy4) og ýmsum tengdum síðum og notkunarleiðbeiningum. Alloy-forritið er í einni *jar*-skrá (sem m.a. inniheldur sjálfan forritstexta Alloy), og það er keyrt með því að smella á hana.

Alloy lýsing er slegin inn í ritilinn sem birtist vinstra megin (hann er reyndar mjög frumstæður), og „keyrð“ með því að smella á *Execute* eða slá á ctrl-E. Til að sjá mynd af líkani er smellt á *Show* eða slegið á ctrl-L og þá birtist sérstakur gluggi með mynd. Hann skyggir reyndar á aðra glugga (hann er *always-on-top*) svo það þarf að velja hentuga staðsetningu og stærð á honum. Reyndar má fá myndina sjálfkrafa með því að velja (í eitt skipti fyrir öll) *Options–Visualize Automatically: Yes*, og þar sem aðal-úttakið eru líkanmyndir er þægilegt að gera það. Ég hef ekki breytt öðrum stillingum. Oftast passa fleiri en eitt líkan við lýsingu og þau má skoða með því að smella endurtekið á *Next* í myndaglugganum (eða slá á ctrl-N). Ennfremur getur verið gagnlegt að skoða mynd af lýsingunni sjálfri með því að velja *Execute–Show Metamodel* (eða ctrl-M).

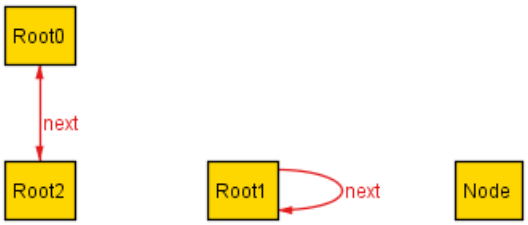
[1] *declarative specification language* (skilgreiningarmál er gagnstætt stefjuðu eða *procedural* máli)

Dæmi 1. Keðjulisti.

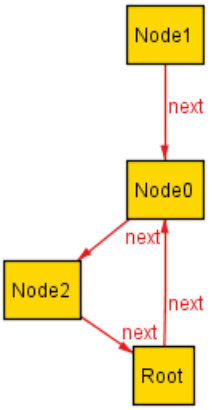
Lítum hér á líkan af keðjulista (*linked list*). Hann er gerður úr safni staka (*nodes*) sem hvert um sig nema það aftasta bendir á næsta stak. Fremsta stakið er kallað rót.

```
sig Node {next: lone Node}
sig Root extends Node {}
fact one_root           {one n: Node | n in Root}
fact next_not_reflexive {no n: Node | n = n.next}
fact next_not_cyclic    {no n: Node | n in n.^next}
fact all_in_list        {all n:Node | n in Root.*next}
run {} for 4
```

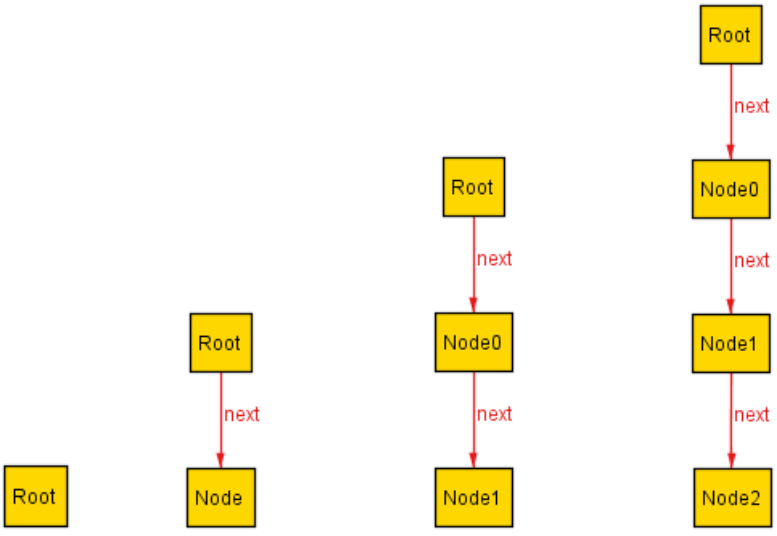
Hægt er að byggja líkanið smám saman upp. Ef öllum staðreyndunum er sleppt (t.d. með því að „kommentera út“ allar línur sem byrja á `fact`) fást ýmis líkön sem ekki eru keðjulistar, t.d.



Síðan má bæta smám saman við staðreyndum. Það er aðeins ein rót, og ekkert stak á að vera venslað við sjálft sig. Fyrstu tvær staðreyndirnar segja þetta, og ef þær eru hafðar með en ekki hinar fæst m.a. líkanið:



Ef allar staðreyndirnar eru með fást svo raunverulegir keðjulistar. Þriðja staðreyndin segir að það séu engir hringir (`^next` er gagnvirk lokun `next`-venslanna svo `n.^next` er mengi þeirra punkta sem hægt er að komast í úr `n`), og fjórða staðreyndin segir að hægt sé að komast í alla punkta úr rótinni (`n.*next` er `n.^next` að viðbættu sjálfu stakinu `n`). Með þessum tveimur staðreyndum fást myndirnar:



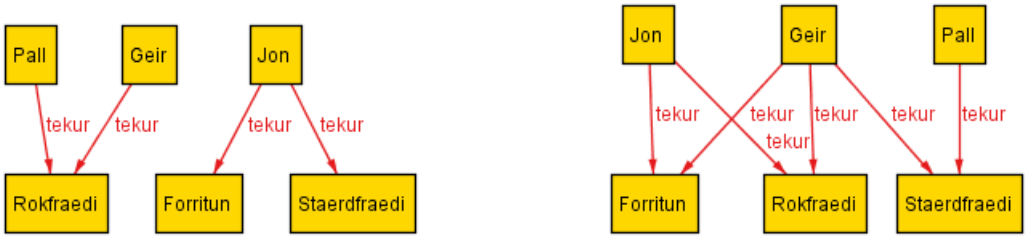
Dæmi 2. Nemendur og námskeið.

Hér er dæmi um einföld vensl:

```
module NemNamsk
abstract sig Nem, Namsk {}
one sig Jon, Pall, Geir extends Nem {}
one sig Rokfraedi, Forritun, Staerdfraedi extends Namsk {}
one sig nnvensl{tekur: Nem some -> some Namsk}
run {#nnvensl.tekur > 3}
```

Þessi lýsing gefur alla möguleika á skráningum Jóns, Páls og Geirs í Rökfræði, Forritun og Stærðfræði. Orðið `abstract` í annarri línu segir að það séu engir hlutir, hvorki af taginu `Nem` né `Namsk`, nema þeir sem erfa frá þeim (með `extends`). Orðið `one` fremst í línum 3–5 segir að þær skilgreini stök en ekki hlutmengi (eins-staks-mengi eru samsömuð stökum). Sem sé: `Nem = {Jon, Pall, Geir}` og `namsk = {Rokfraedi, Forritun, Staerdfraedi}`. Magnararnir `some` í næstneðstu línunni tryggja að hver nemandi taki a.m.k. eitt námskeið, og að ekkert námskeið sé nemendalaust, og með skilyrðinu í `run` skipuninni er beðið um líkön með fleiri en 3 pörum í venslunum.

Til að fá góða mynd þarf að útrýma `nnvensl` kassa úr henni. Það er gert með því að smella á *Projection: none* og haka við `nnvensl`. M.a. fást eftirfarandi líkön:



Snið Alloy-lýsingar

Lýsing líkans með Alloy byrjar á *module*-línu og síðan er röð af `open`-línum, þá `sig`-skilgreiningar og loks klausur:

```
module nafn
-- skjölun
open nafn
⋮
sig nafn … {skilgreiningar sviða}
⋮
lykilorð nafn … {skilgreining}
⋮
```

Reyndar má sleppa `module`-línunni og/eða `open`-línunum og `sig`-skilgreiningar mega koma inn á milli klausa. Á undan `sig` má rita `abstract`, og ennfremur mega koma þar *magnarar*: `one`, `lone` eða `some`. Á eftir „`sig` *nafn*“ má koma „`extends` *nafn*“ eða „`in` *nafn*“. Lykilorðin sem geta byrjað klausurnar eru `fact`, `pred`, `fun`, `assert`, `check` og `run`. Línur sem byrja á `open` notast til að vísa í aðrar lýsingar (þær svara til *import* í Java). Alloy gerir engan greinarmun á bili og línuskiptatákni.

Hægt er að setja skjölun (*documentation, comment*) í Alloy-lýsingar á eftir `//` eða milli `/*` og `*/` eins og í C, og það má líka byrja athugasemd með `--` (eins og gert er í kennslubók).

Skilgreiningar og vensl

Í umsagnarrökfræði er upptalning á föstum, föllum og umsögnum (sem táknuð eru með parinu ($\mathcal{F}$, $\mathcal{P}$) í kennslubók) ásamt viðfangafjölda þeirra (*arity*) stundum kallað *signature*[2]. Í Alloy eru talsvert fleiri möguleikar en í þeirri umsagnarrökfræði sem við höfum skoðað. Hluti sem fjallað er um má t.d. skilgreina með:

`sig` *nafn* `{}` (1a)
`sig` *nafn* `{`*svið* `:` *nafn*, …`}` (1b)
`sig` *nafn* `{`*svið* `:` *M nafn*, …`}` (1c)

[2] þetta er auðvitað náskylt notkun á orðinu *signature* um það hvernig kalla skal á fall í venjulegri forritun

þar sem … táknar að endurtaka má „*svið* `:` [*M*] *nafn*“ eins oft og vill, og *M* getur verið `set` (= hlutmengi), `one` (= eins-staks hlutmengi = eitt stak), `lone` (= núll eða eitt stak; *less-than-or-equal-to-one*) eða `some` (= eitt eða fleiri stök). Jafngilt er að skrifa „*svið* `:` *nafn*“ og að skrifa „*svið* `:` `one` *nafn*“.

Skilgreining (1a) svarar til skilgreiningar grunnmengis eða óðals, en það er sem sé hægt að hafa fleiri en eitt óðal, t.d. má rita:

```
sig Nem {}
sig Namsk {}
```
(2)

sem reyndar má stytta í „`sig Nem,Namsk {}`“ (sbr. dæmi 2)**.** Við sjáum að þetta er svolítið eins og að skilgreina tög í forriti, og skilgreiningar (1b) og (1c) minna um margt á skilgreiningu klasa með tiltekin svið eða eiginleika. Það er líka ritað *nafn*`.`*svið* til að auðkenna svið sem tengist nafni. Skilgreiningar eins og (1b,c) gefa eina leið til að skilgreina vensl eða umsagnir. Þannig mætti rita

```
sig Namsk {taka : some Nem}
```
(3)

Ef `n` er af taginu `Namsk` þá verður „`n.taka`“ mengi nemenda sem taka námskeiðið `n`. Við sjáum líka að með því að nota orðið „`one`“ má skilgreina fall (vensl með nákvæmlega eitt stak venslað við hvert stak):

```
sig Barn {bestivinur : one Barn}
```

En það eru fleiri leiðir til að skilgreina vensl í Alloy, og fyrsta dæmið í kennslubókinni, á bls. 143–145 sýnir það. Þar er kerfið sem verið er að lýsa látið vera hlutur (`StateMachine`) og lýsingin á óðalinu (`State`) er án sviða, en hinsvegar eru svið í `StateMachine` sem tilgreina upphafsstöðuna `i`, hvort staða sé lokastaða (`F`) og hvort hægt sé að færa sig úr stöðu í aðra tiltekna stöðu (`R`). Rithátturinn „`R : A -> A`“ táknar að `R` séu vensl milli staka í `A` (`R` er hlutmengi í $A \times A$). Til að búa til þau vensl að nemandi taki námskeið mætti t.d. rita:

```
one sig nnvensl{tekur : Nem->Namsk}
```
(4)

Hvert líkan sem Alloy býr til á aðeins að hafa ein vensl og því er sett `one` fremst. Almennt geta líkön sem búin eru til eftir lýsingu innihaldið engan, einn eða fleiri hluti af hverju tagi (t.d. mörg námskeið eða marga nemendur), en því má breyta með því að setja magnaraorð framan við `sig`, eitt af orðunum `one`, `lone` og `some` (`lone` þýðir 0 eða 1 og `some` þýðir ≥ 1). Og eins og sýnt var í dæmum 1 og 2 geta tög erft frá öðrum tögum:

```
abstract sig Madur {}
sig Karl extends Madur {eiginkona: lone Kona}
sig Kona extends Madur {eiginmadur: lone Karl}
```

Hér þýðir `abstract` að ekki sé hægt að búa til eintak af `Person`, sem sé allir eru annaðhvort menn eða konur. Eins og fyrr segir má líka skilgreina hlutmengi með `in`, t.d. „`sig A1,A2 in B {}`“, en það á ekki við hér því `A1` og `A2` þurfa þá ekki að vera sundurlæg (og einhverjir gætu sem sé orðið tvíkynja).

Að lokum skal nefndur til sögunnar möguleikinn að rita:

`sig` *nafn* `{`*skilgreining*`}{`*skorða*`}`

þar sem *skorða* er skilyrði (satt eða ósatt) sem hlutir af taginu nafn þurfa að uppfylla. Dæmi um þetta er skilgreining á `M` í dæmi 4 að aftan.

Tög

Allar breytur sem notaðar eru í Alloy þurfa að vera af tilteknu skilgreindu tagi, (þær tilheyra grunnmengi þess tags), og það eru tvær leiðir til að skilgreina tög: Með `sig` og með því að nota svið. Þannig skilgreinir (2) tagið `Nem`, (3) tögin `Namsk` og `Namsk.taka` (það síðara samanstendur af öllum nemendum sem taka eitthvert námskeið), og (4) skilgreinir tögin `nnvensl` og `nnvensl.tekur` (það síðara er mengið `Nem` × `Namsk`).

Formúlur og virkjar

Það eru þrjár gerðir af gildum sem koma fyrir í Alloy lýsingum: sanngildi, heiltölur, og gildi af skilgreindu tagi. Formúlurnar í Alloy eru samsettar úr breytum og virkjum, og hægt er tilgreina röð aðgerða með svigum. Dæmi um formúlu sem skilar sanngildi er:

```
#A = 3 + #B and (x in A + B or A in B)
```

Ef `x` er stak og `A` er mengi merkir hún: „`A` hefur þremur fleiri stök en `B` og annaðhvort er `x` stak í $A \cup B$ eða `B` hlutmengi í `C`“. Formúla sem skilar tagi er t.d. „`Karl - Kona.eiginmadur`“ (þ.e.a.s. allir piparsveinar).

Rökvirkjar

Þeir virkjar sem skila sanngildi eru hefðbundnir rökvirkjar, magnarar og samanburðarvirkjar. Fyrir suma virkjana geta menn valið um að nota orð eða tákn, t.d. `and` eða `&&`, og á sama hátt `or` eða `||`. Ath. að ekki er gerður greinarmunur eins-staks-mengi og stakinu sem það geymir, svo `in` getur táknað bæði $\in$ og $\subseteq$.

Alloy virki	Merking	Hefðbundin rökfræðitáknun
`&&`, `and`	og	$\wedge$
`||`, `or`	eða	$\vee$
`!`, `not`	ekki	$\neg$
`=>`, `implies`	leiðir til	$\rightarrow$, $\vdash$, $\models$, $\Rightarrow$
`<=>`, `iff`	er jafngilt og	$\dashv\vdash$, $\Leftrightarrow$
`all x:A|…`	fyrir öll `x` í `A` gildir …	$\forall x \in A\,(...)$, $\forall x(x \in A \rightarrow ...)$, $\forall x(...)$ (óðalið undanskilið)
`some x:A|…`	til er `x` í `A` þannig að …	$\exists x \in A\,(...)$, $\exists x\,(x \in A \wedge ...)$, $\exists x(...)$ (óðalið undanskilið)
`no x:A|…`	um ekkert `x` í `A` gildir …	$\neg\exists x(...)$ eða $\forall x \neg(...)$
`one x:A|…`	um nákvæmlega eitt `x` í `A` gildir …	
`lone x:A|…`	um 0 eða 1 `x` í `A` gildir …	
`in`	er stak í / er hlutmengi í	$x \in A$, $x \subseteq A$
`=`	er jafnt og	$=$
`<`, `>`, `=<`, `=>`	er minna en, stærra en…	$<, >, \leq, \geq$

Helstu rökvirkjar, magnara og samanburðarvirkjar í Alloy (virkjar sem skila sanngildi)

Virkjar sem skila tagi

Tögin í Alloy eru í raun mengi (eða a.m.k. má túlka þau sem mengi), og því svara virkjar sem þeim tengjast til mengjavirkja í stærðfræði. Til viðbótar hefðbundnum mengjafræðivirkjum eru síðan nokkrir virkjar sem tengjast venslum, endar skipa vensl mikilvægan sess í Alloy-málinu. Lokunarvirkjarnir `^` og `*` komu við sögu í keðjulistadæminu hér framar. Í eftirfarandi töflu tákna `R` og `Q` vensl og `A` og `B` frámengi og aðmengi.

Alloy virki	Merking	Hefðbundin mengjafræðitáknun
`+`	sammengi	$\cup$
`&`	sniðmengi	$\cap$
`–`	mengjamismunur	$-$
`R.B`	aðmengi vensla með frámengi `B`	$a \in$ `R.B` $\Leftrightarrow \exists b \in$ `B` með $(a,b) \in$ `R`
`A.R`	frámengi vensla með aðmengi `A`	$b \in$ `A.R` $\Leftrightarrow \exists a \in$ `A` með $(a,b) \in$ `R`
`R.Q`	tenging tveggja vensla (*join*)	$(a,b) \in$ `R.Q` $\Leftrightarrow \exists x$ með $(a,x) \in$ `R` og $(x,b) \in$ `Q`
`~R`	spegilvensl	$(a,b) \in$ `~R` $\Leftrightarrow (b,a) \in$ `R`
`^R`	gegnvirk lokun	$(a,b) \in$ `^R` $\Leftrightarrow \exists x_1, x_2, ..., x_n$ með $(a,x_1),(x_1,x_2),...,(x_n,b) \in$ `R`
`*R`	sjálfhverf gegnvirk lokun	`^R` $\cup$ (mengi allra tvennda (a,a) með $a \in$ frámengi `R`)

Helstu virkjar í Alloy sem skila skilgreindu tagi (frámengi = domain, aðmengi = range)

Talnavirkjar

Heiltölur má m.a. búa til með því að telja stök í mengi (með virkjanum `#`), en annars er ekki mikið talað um tölur á þessum blöðum.

Alloy virki	Merking	Hefðbundin stærðfræðitáknun
`#A`	fjöldi staka í `A`	$\#(A)$, $\lvert A \rvert$
`+`	samlagning	$+$
`-`	frádráttur	$-$

Heiltöluvirkjar

Fastar

Alloy skilgreinir þrjá fasta:

`none`	tómamengið
`univ`	almengið (sammengi allra óðala sem skilgreind eru)
`iden`	einingarvenslin, sem vensla sérhvert stak við sjálft sig (þ.e.a.s. $\{(a,a) \mid a \in \texttt{univ}\}$)

Staðreyndir og staðhæfingar

Klausa með **staðreynd** sem öll líkönin sem Alloy býr til þurfa að uppfylla er rituð:

```
      fact nafn {formúla}
eða:  fact {formúla}
```

Hægt er að setja margar staðreyndir í sömu klausuna:

```
fact {
   formúla1
   formúla2
}
```

Að gefa staðreynd nafn gagnast sem skjölun (*documentation*) og auðveldar manni að tala um staðreyndina, en er að öðru leyti ekki nauðsynlegur hluti Alloy-lýsingar. Sem dæmi um staðreynd má t.d. taka:

```
fact {hjon[jon,gunna]}
```

Í forritum eru oft notað svonefnt *assert* til að athuga hvort einhver **staðhæfing** sem þarf að gilda (eða ætti að gilda) gildi í raun. Í Alloy má skilgreina slíkar staðhæfingar með:

```
assert nafn {formúla}
```

Síðan má prófa sannleiksgildi þeirra síðar með `check` (sjá neðar).

Umsagnir og föll

Oft má líta á vensl (hvort sem er af gerðinni (3) eða (4)) sem **umsögn** (*predicate*), en stundum hentar að skilgreina umsögn berum orðum með því að rita:

```
pred nafn[stiki1,…:tag,…] {formúla}
```

Slíka umsögn má nota í öðrum formúlum með því að rita *nafn*`[`*viðfang*`,…]`, líkt og þegar kallað er á föll í venjulegum forritunarmálum. Þannig mætti rita:

```
      pred systur[x,y:Kona] {fadir[x]=fadir[y] or modir[x]=modir[y]}
og:   pred hjon[x:Karl,y:Kona] {x.eiginkona=y}
```

Það er líka hægt að skreyta tagið með magnara, `one`, `lone`, `some` eða `set` og `one` er sjálfgefið). Í sumum löndum ætti við að rita „`pred hjon[x:Karl,y:some Kona]`“ eða „`pred hjon[x:some Karl,y:Kona]`“.

Umsagnir skila gildi sem er annaðhvort satt eða ósatt, en einnig má skilgreina **föll** sem skila gildi af einhverju tagi. Það er gert svona:

```
fun nafn[stiki1,…:tag,…]:skilatag {formúla}
```

Þetta fall má svo kalla á með *nafn*`[`*viðfang*`,…]` og það skilar niðurstöðu sem *formúla* gefur af tagi *skilatag*. Hér má líka bæta magnara við hvort sem er *tag* eða *skilatag*.

Keyrsla

Tvær skipanir eru notaðar til að keyra lýsingar (sem sé leita að líkönum sem passa við þær), `run` og `check`:

```
run umsögn
check staðhæfing
```

Skipanirnar verka svo hvor á sinn veginn: `run` býr til líkön sem gera umsögnina *sanna*, en `check` reynir að búa til líkön sem gera staðhæfinguna *ósanna*.

Umsögnin sem keyrð er með `run` má vera hvort sem er *nafn* sem áður hefur verið skilgreint með `pred` eða nafnlaus umsögn, tilgreind sem formúla innan `{}`:

```
      run hjon
eða:  run {x.eiginkona = y}
```

Oft hentar að láta umsögnina vera tóma:

```
run {}
```

Staðhæfingin sem prófuð er með `check` má líka vera hvort sem er *nafn* áður skilgreint með `assert` eða nafnlaus formúla innan `{}`.

Fyrir bæði `run` og `check` má tilgreina umfang leitarinnar. Sjálfgefið er að prófa alla möguleika á grunnmengjum (sem sé tögum sem skilgreind eru með `sig`) sem hafa 3 eða færri stök, nema lýsingin festi stakafjölda grunnmengjanna (t.d. með `one`). Svo eru prófaðir allir möguleikar á tögum sem erfa frá grunntögum (grunnmengjum), eru hlutmengi í þeim, eða vensl milli þeirra. Umfanginu má svo breyta með því að hengja aftan við keyrslskipanirnar viðauka, sem byrjar á `for`. Ef lýsingin hefur tögin `Karl`, `Kona` og `Barn` (og ekki önnur), þá má rita

```
run umsögn for 5
run umsögn for 5 Karl, 5 Kona, 5 Barn
run umsögn for 5 Karl, 5 Kona, 3 Barn
run umsögn for 5 but 3 Barn
run umsögn for 5 Karl, 5 Kona, exactly 3 Barn
```

Fyrstu tvær skipanirnar eru jafngildar, og sömuleiðis næstu tvær. Samskonar viðauka má hengja við `check`.

Í Alloy Analyzer er sjálfgefið að ef smellt er á *Execute* eða slegið á ctrl-E þá er aðeins fyrsta keyrsluskipun lýsingarinnar framkvæmd (hvort sem hún er `run` eða `check`), en með því að velja *Execute All* í *Execute*-valmyndinni má keyra þær allar. Það er líka hægt að keyra tiltekna skipun með valmyndinni (og sú skipun verður þá sjálfgefin fyrir hnappinn og ctrl-E).

Dæmi 3. Athugun á sanngildi yrðinga

Lítum á tvo liði úr dæmi 2.4.12 í kennslubókinni (sjá bls. 164). Í b-lið skal kanna skal sanngildi formúlunnar

$$\exists y((\forall x\, P(x)) \rightarrow P(y))$$

Við rifjum upp úr skilgr. 2.14 á bls. 124 í kennslubókinni að líkan fyrir svona formúlu samanstendur af grunnmengi A sem er ekki tómt, og hlutmengi P^M í A af stökum sem gera yrðinguna sanna. Ef formúlan inniheldi frjálsar breytur þyrfti svo að gefa þeim gildi, en hér eru engar slíkar. Þetta gefur:

```
sig A {}
fact {some A}
sig PM in A {}
pred P[x:A] {x in PM}
pred fi {some y:A | (all x:A | P[x]) => P[y]}
```

(fyrstu tvær línurnar mætti stytta í „`some sig A {}`“). Ef þessi lýsing er keyrð með „`run fi`“ finnst strax líkan sem gerir formúluna sanna, og Alloy skrifar „Instance found. Predicate is consistent“. Næsta skref er þá að kanna hvort hægt sé að finna mótdæmi. Það má t.d. gera með:

```
run {not fi} for 100
```

(einnig mætti láta `fi` vera `assert`, og rita svo „`check fi for 100`“). Nú skrifar Alloy: „No instance found“. Það þýðir að ekkert líkan með ≤ 100 stökum í A gerir formúluna ósanna, og það bendir sem sé allt til þess að hún sé gild setning (enda er ekki erfitt að sanna hana með náttúrulegri leiðingu eða rökstyttingu).

Lítum næst á g-lið dæmisins, með

$$\phi = \forall x \exists y \big(S(x, y) \wedge (S(x, y) \wedge S(y, x) \rightarrow x = y)\big) \rightarrow \neg\, \exists z \forall w (S(z, w)).$$

Þar sem S tekur tvö viðföng þarf að láta S^M vera vensl, þ.e. hlutmengi í $A \times A$. Til þess eru tvær leiðir eins og nánar verður vikið að í dæmi 4 að aftan, en við látum aðra duga, þá sem pakkar venslunum inn í séstakt tag. Ennfremur notum við `check` í stað `run` (því við höfum komist að því að formúlan er ekki gild).

```
some sig A {}
one sig M {SM: A -> A}
pred S[x,y: A] {x->y in M.SM}
assert fi {
  (all x:A|some y:A|S[x,y] && (S[x,y]&&S[y,x] => x=y)) => !(some z:A|all w:A|S[z,w])
}
check fi
```

Og Alloy skrifar að bragði út „Counterexample found“ og teiknar fyrir okkur mynd af mótdæminu.

Dæmi 4. Athugað hvort líkan passi við yrðingu

Í dæmi 2.4.5a í kennslubók er gefin yrðingin $\phi = \forall x \forall y \exists z(R(x, y) \rightarrow R(y, z))$ og líkanið:

$$A = \{a, b, c, d\}$$
$$R^M = \{(b,c),(b,b),(b,a)\}$$

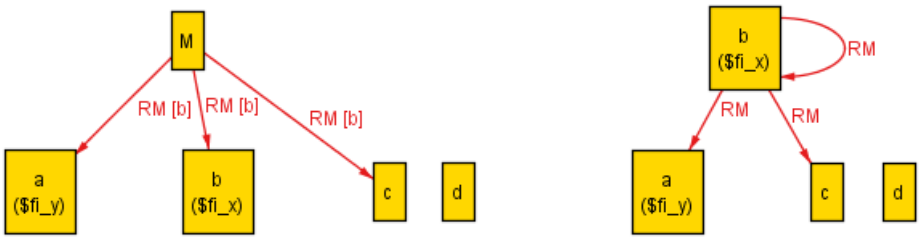
og kanna skal hvort það geri yrðinguna sanna (sem sé hvort $M \models \phi$). Mengið R^M er mengi para (x, y) sem gerir umsögnina $R(x, y)$ sanna. Paramengi er í raun vensla og eins og fyrr segir eru tvær leiðir til að skilgreina vensl í Alloy og gagnlegt er að skoða þær báðar. Eftirfarandi fjórar línur má samt setja í lýsinguna hvor leiðin sem valin er:

```
pred fi {all x,y : A | some z : A | R[x,y] => R[y,z]}
one sig a,b,c,d extends A{}
run {fi}
run {not fi}
```

Til að skilgreina venslin er líklega einfaldara að búa til sérstakt tag sem hefur þann tilgang einan að halda utan um þau, sbr (4). Ef þetta sérstaka tag heitir `M` og venslin heita `RM` þá bætist við lýsinguna:

```
abstract sig A {}
pred R[x,y: A] {x->y in M.RM}
one sig M {RM : A -> A}{RM = b->c + b->b + b->a}
```

Ef valið er *Execute–Execute All* skrifar Alloy fyrir fyrri `run` skipunina „No instance found“, sem þýðir að líkanið passar ekki, og seinni `run` skipunin skilar svo „Instance found“ og myndum sem sýna hvað x og y eru í mótdæminu:



(seinni myndin fæst með því að velja „Projected over M“ og eyða þannig M-kassanum).

Hin leiðin til að skigreina venslin er að bæta sviði við grunnmengið, eins og í (3). Þá bætist við fyrstu fjórar línur lýsingarinnar:

```
pred R[x,y: A] {x in A and y in x.r}
abstract sig A {r: set A}
fact {
  a.r = none
  b.r = a+b+c
  c.r = none
  d.r = none
}
```

Við sjáum að það er dálítið flóknara að skilgreina venslin ef þessi leið er valin.

Breytingar milli Alloy 2 og Alloy 4

Eins og segir hér fremst liggur útg. 2 af Alloy til grundvallar kennslubókinni, en þessi blöð lýsa útg. 4. Meðal þess sem hefur breyst er eftirfarandi:

1. Til að skilgreina umsagnir er nú notað `pred` en ekki `fun`.
2. Í skilgreiningu falla og umsagna (`fun` og `pred`) kemur stikalisti innan hornklofa, [ ], en ekki innan sviga, ( ).
3. Sömuleiðis skal nota hornklofa en ekki sviga þegar kallað er á umsagnir og föll.
4. Nú er ritað `lone` (*less-than-or-equal-to one*) í stað `option`
5. Ekki er lengur hægt að nota `with` eins og gert er sumsstaðar í dæmum bókarinnar.

Raunar hefur ýmislegt fleira breyst en við látum þennan stutta lista duga.

Kristján Jónasson, 1. mars 2011