CHALMERS

Learn You a Physics for Great Good

Using domain spedific languages to learn physics

DAVID FRISK

Instutitionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2018

Rätt namn på "dep ment"?

Kandidatarbete DATX02-18-08

An Informative Headline describing the Content of the Report

A Subtitle that can be Very Much Longer if Necessary

Björn Werner
Erik Sjöström
Johan Johansson
Oskar Lundström

Sortera efter efter namn?

CHALMERS

Instutionen för Data- och Informationsteknik
Division of Division name
Chalmers tekniska högskola
Göteborg, Sverige 2018

An Informative Headline describing the Content of the Report
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME

© NAME FAMILYNAME, 2018.

Supervisor: Name, Company or Department
Examiner: Name, Department

Master's Thesis 2018:NN
Department of Some Subject or Technology
Division of Division name
Name of research group (if applicable)
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wind visualization constructed in Matlab showing a surface of constant wind speed along with streamlines of the flow.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2018

An Informative Headline describing the Content of the Report
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME
Department of Some Subject or Technology
Chalmers University of Technology

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: lorem, ipsum, dolor, sit, amet, consectetur, adipisicing, elit, sed, do.

Sammanfattning

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: lorem, ipsum, dolor, sit, amet, consectetur, adipisicing, elit, sed, do.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Name Familyname, Gothenburg, Month Year

Innehåll

Figurer

Tabeller

Ordlista

I bokstavsordning när vi har med all ord vi ska.

DSL Förkortning av *Domain Specific Language*, engelska för domänspecifikt språk.

Grundläggande område Ett område i läromaterialet som inte bygger på något annat område. Till exempel vektorer.

Komposit område Ett område i läromaterialet som bygger på andra områden. Till exempel lutande plan, som använder sig av vektorer.

Åke Fäldt Föreläsare och examinator i kursen Fysik för ingenjörer.

Fysik för ingenjörer En fysikkurs som är obligatorisk för studenter på Datateknik civilingenjör på Chalmers. Den ges i årskurs 2 och innehåller grunderna i mekanik, termodynamik och vågrörelselära.

Univerisity Physics Den fysikbok som används i Fysik för ingenjörer.

Läromaterialet Syftar på den pedagogiska text som projektet resulterat i.

1

Introduktion

Ska in nånstans i bakgrund eller diskussion

Vilka är projektet relevant för: Projektet är relevant för datateknologer som läser en fysikkurs. Men det kan också bli relevant för en fysikstudent som är ute efter en inkörsport till funktionell programmering. Förhoppningsvis blir det också relevant för de som är intresserade av domänspecifika språk i stort, pedagoger och föreläsare inom de berörda områdena och kanske till och med programledningen som ser vår rapport som ett skäl att introducera innehåll av detta slag i till exempel fysikkursen.

1.1 Bakgrund

På civilingenjörsprogrammet Datateknik på Chalmers ingår den obligatoriska fysikkursen *Fysik för ingenjörer*. Tentastatistiken för denna kurs är inte jättebra[1]. Vi tror att många studenter på datateknik ("datateknologer") finner denna kurs svår eller ointressant, och att detta leder till att ungefär en tredjedel av kursdeltagarna får underkänt på tentamen.

Examinatorn för kursen "Fysik för ingenjörer, TIF085 (2016)" Åke Fäldt, tycker att studenter i allmänhet verkar ha svårt för att sätta upp egna modeller. De baserar sina mentala modeller helt eller delvis på intuition och felaktiga antaganden, istället för definitioner och bevisade satser som man är säker på gäller. Detta leder till att de tar genvägar som ofta är fel.

Detta tror projektgruppen skulle kunna lösas med avstamp från kursen "Domain Specific Languages of Mathematics" ("DSLsofMath") eller "Matematikens domänspecifika språk" vilket är en valbar kurs på kandidatnivå för studenter på Chalmers och Göteborgs Universitet.

Konkret så presenterar kursen matematik så som derivator, komplexa tal och matriser ur ett funktionellt programmeringsperspektiv i det funktionella programmeringsspråket Haskell. Dessa för studenterna bekanta verktyg används för att lösa matematiska problem så som modellering av syntax, evaluering till semantiska värden och datorassisterad bevisföring.

ka nämnas en
ång tidigt och en
ång sammanfatt-
ingsvist nedan,
ler beskriver vi
ad vi ska göra
ngst upp innan
akgrund?

För vidare läsning rekommenderas *DSL for the Uninitiated*.[2] Projektgruppen ämnar att implementera domänspecifika språk för att beskriva fysik ur en alternativ vinkel.

I kursen DSLsofMath, var år 2016 var Cezar Ionescu huvudföreläsare, och från 2017 är Patrik Jansson huvudföreläsare, vilka har beskrivit avseendet med kursen genom en artikel [3]. Det direkta målet med kursen DSLSofMath är att förbättra den matematiska utbildningen för datavetare och den datavetenskapliga utbildningen för matematiker, där den grundläggande idéen bakom kursen är:

> "[. . .] att uppmuntra studenterna att närma sig matematiska domäner från ett funktionellt programmeringsperspektiv: att ge beräkningsbevis (calculational proofs); att vara uppmärksamma på syntaxen för matematiska uttryck; och, slutligen, att organisera de resulterande funktionerna och typerna i domänspecifika språk."[4][3]

Vi vill alltså med hjälp av domänspecifika språk som vi implementerar presentera fysik ur ett alternativt perspektiv, likt på det sättet DSLsofMath presenterar kopplingar mellan matematik och programmering. Förhoppningen är att detta ska göra kopplingen mellan datateknikprogrammet och fysik tydligare och därmed underlätta lärandet.

En analogi:

Studenterna hade svårt för matte → DSLsofMath.
Studenterna har svårt för fysik → Learn you a physics.

Angående tidigare forskning och studier, finns även på MIT har en kurs inte helt olik DSLsofMath tidigare givits som berör både fysik och domänspecifika språk ("DSL"). "Classical Mechanics: A Computational Approach" gavs av Prof. Gerald Sussman och Prof. Jack Wisdom bl.a. år 2008.[5] Denna kurs på avancerad nivå studerar de fundamentala principerna av klassisk mekanik med hjälp av beräkningsidéer för att precist formulera principerna av mekanik, med början i Lagranges ekvationer och avslut i perturbationsteori (teori för approximationer av matematiska lösningar). I kursen används boken "Structure and Interpretation of Classical Mechanics" av Sussman, Wisdom och Mayer, vilken förklarar fysikaliska fenomen genom att visa datorprogram för att simulera dem, skrivna i språket Scheme.[6]. Denna typen av kurser ter sig ovanliga, och är, till våran kännedom, den enda kursen bortsett från DSLsofMath på Chalmers som knyter samman matematik, fysik och programmering.

Utöver DSLsofMath-kursen har det även tidigare gjorts ett kandidatarbete om DSL här på Chalmers. Vårterminen 2016 utfördes kandidatarbetet "Programmering som undervisningsverktyg för Transformer, signaler och system. Utvecklingen av läromaterialet TSS med DSL" av fem studenter från Datateknik och Teknisk Matematik på Chalmers. Arbetet bestod av utveckling av läromaterial med tillhörande programmeringskod, uppgifter och lösningar, som komplement till existerande kurser i signallära.[7]

1.2 Projektets mål

Tanken med detta kandidatarbete är att angripa fysik från ett funktionellt programeringsperspektiv. På detta sätt ska ämnet bli både roligt och intressant för datastudenter, och därmed förhoppningsvis också enklare. Detta är likt premissen bakom kursen DSLsofMath och kandidatarbetet från 2016, som istället för fysik behandlade matematik respektive signallära.

Mer konkret ska ovanstående göras genom att skapa ett läromaterial. Läromaterialet ska bestå av domänspecifika språk, skrivna i Haskell, som modellerar fysik sammanvävt med förklarande lärotext. Läromaterialet ska i slutändan publiceras på en hemsida och all källkod ska finnas öppet tillgänglig.

En del av målet är också att ta reda på hur en kombination av domänspecifika språk och fysik kan se ut, samt om det finns en pedagogisk nytta i att kombinera dem.

1.3 Avgränsningar

Läromaterialet ska begränsa sig till att enbart beskriva de fysikalsika områden som ingår i kursen Fysik för ingenjörer. Denna avgränsning valdes dels för att det är den fysik gruppmedlemmarnas kunskapar begränsar sig till, dels för att det är för den kursen detta projekt kan bli mest relevant för, då kursen ingår i datastudenternas obligatoriska kursplan.

Vidare kommer en prioritering av innehållet i Fysik för ingenjörer att göras. Kursen behandlar grunderna inom klassisk mekanik, termodynamik och vågrörelselära. Det ingår även en stor mängd tillämpad matematik, exemeplvis differentilkalkyl. I första hand kommer mekaniken behandlas, för att sedan i mån av tid även behandla termodynamik och vågrörelselära. Fokuset kommer också att vara på de områden datastudenter haft svårt för.

För att utvärdera den pedagogiska nyttan kommer enbart en informell utvärdering att göras. Detta då en rigorös undersökning hade krävt mycket tid för att välja lämpliga testgrupper, analysera återkopplingen samt dokumentera testningsförloppet.

2

Teori

2.1 Domänspecifika språk

DSL ⇒ Modellera syntax.

Ett domänspecifikt programmeringsspråk är ett språk som är avgränsat till ett specifikt domän. Detta domän kan ta många former, det kan vara ett språk för att formatera text på en hemsida (HTML), det kan vara ett språk för att interagera med en databas (SQL), ett språk för ett beskriva hur text ska se ut på en skärm (typsnitt). Användningsområden för dessa språk är väldigt smala men detta smala fokus gör det möjligt att utveckla ett rikt och lättanvändligt språk för just detta område.

Motsatsen till ett domänspecifikt språk är ett generellt språk (C, Java, Python) vilka är turingkompletta, vilket betyder att det kan uttrycka alla beräkningsbara problem i dem och även lösa dem givet tillräckligt med tid och minnestillgångar. Begränsningen med dessa generella språk är just deras egen generaliserbarhet, eftersom de har stöd för alla typer av beräkningar så blir både läsbarheten och användarvänligheten lidande.

referens

referens/motiverin

Ett domänspecifikt språk kan antingen implementeras som ett fristående språk eller bäddas in i ett redan existerande språk. De domänspecifika språk som utvecklats inom detta projekt är inbäddade i språket *Haskell*.

2.2 Haskell och funktionell programmering

Haskell är ett funktionellt programmeringsspråk som lämpar sig bra för att implementera ett domänspecifikt språk i. Anledningen till detta är den lätthet som man kan skapa nya datatyper och klasser för att representera grundstenarna i det nya språk, och även dess mönstermatchning som gör det möjligt att på enkelt sätt bryta isär komplexa datatyper för evaluering.

TODO: Visa exempel på detta

```
data Expr = Expr :+: Expr
          | Expr :*: Expr
          | Const Double
          | VarX

exempel = (Const 7.0 :+: VarX) :*: ((VarX :+: Const 10.0)
          :*: VarX)
```

Figur 2.1: Bild!?

2.3 Syntaxträd och deras evaluering

Ska kanske slås ihop med DSL-avsnittet, då syntaxträd är ett så bra exempel på DSL

Ett typexempel på ett domänspecifkt språk är ett syntaxträd för algebraisk uttryck, här kodat i Haskell.

BYT TILL MONOSPACE

I detta exempel visar hur uttrycket $(7 + x) * ((x + 10) * x)$ kodas.

Ha med en bild på detta.

!!!Plus förklaring om evaluering!

2.4 Fysik för ingenjörer

Fysik för ingenjörer är en obligatorisk kurs för studenterna på det datatekniska programmet. Kursen täcker grunderna inom klassisk mekanik, termodynamik och vågrörelselära. Det ingår även en del tillämpad matematik såsom vektorer och differentialkalkyl. Kursen föreläsare och examinator är Åke Fäldt.

2.5 Hemsidan

a beskrivning
v individuella
eknologier

Literat och lhs m.m. står redan om i annat avsnitt. Programfilerna har skrivits i ett språk som kallas *Literate Kaskell* som kombinerar vanlig Haskell-kod med brödtext till något som kan tolkas av en kompilator som två separata saker. *Pandoc* är ett program som används för att generera html-filer med läromaterialet som är den slutgiltiga produkt som finns tillgänglig på hemsidan.

2.6 Litterat programmering och Literate Haskell

Litterat programmering (engelska *literate programming*) är ett alternativt sätt att programmera som introducerats av Donald Knuth.[8] Istället för att skriva ett program för en dator, skriver man ett program som ska läsas av människor. Det visar sig på följande två sätt.

Det en sättet är att jämfört med traditionella program får dokumentationen en ökad betydelse. I traditionella program är programkoden den viktiga delen. I litterata program däremot är dokumentationen minst lika viktig. Den anänds till att förklara koden, sätta den i relationen till andra delar och så vidare. Detta jämnbördiga förhållande syns konkret genom att titta på hur källkoden är skriven i ett literat program. Det kan till exempel se ut som i figur 2.2 där man ser att källkod och text för människor är sammanvävda på ett jämnbördigt sätt, där den ena inte är viktigare än den andra.

```
How does all this tie together? First the type is decided,
   for instance

< type ExampleType = Quantity T.Length Double

then a value of that type is created

< exampleValue :: ExampleType
< exampleValue = Quantity V.length 5.3

Note that the Quantity data type has both value-level and
   type-level dimensions. As previosuly mentioned, value-
   level in order to print prettily and type-level to only
   permit legal operations.
```

Figur 2.2: Ett exempel på hur en källfil till litterat programmering kan se ut. Exemplet är Litterate Haskell. Rader som börjar med `<` markerar att det är progamkod, medan rader utan markerar att det är text för människor.

Det andra sättet ett litterat program skiljer sig åt är ordningen progamkod står i. Traditionellt används top-down eller bottom-up. Dessa tillvägagånssätt står i kontrast till den som används i litterat programmering, där man betraktar ett program som ett nät av många delar som hänger ihop, utan att de nödvändigvis har en övergripande hierariki som de tradionella metoderna bygger på. Detta uttnyttjas genom att då presentera progamkoden i den ordning som är lättast att förstå för den mänskliga läsaren.

Litterate Haskell är literat programmering för Haskell.[9] Att progammera i Litterate Haskell går till på samma sätt som vanlig Haskell, med skillndaden att programkod och text vävs ihop i en och samma fil. Det kan se ut som i figur 2.2. Filen, med tillägget `.lhs`, går att använda direkt med Haskell-kompilatorn GHC. All text ignoreras och programkoden behandlas som om den tillhörde en vanlig Haskell-fil. `.lhs`-filen kan också kompileras till material avsett för mäniskor. Det finns flera verktyg som gör det men det

som används i detta projekt är *Pandoc*[10]. Med *Pandoc* kan texten märkas up med både `markdown` (används i projektet) och `Latex`. Det går att exportera till bland annat HTML och PDF.

2.7 ARCS/didaktik

Lärande Skola Bildning - kap 5 lärandeteorier

Knyta an materialets undervisningspotential till lärandeteorier.

Vygotskij - Sociokulturella lärandet (ex parprogrammering + diskussion emellan individer)

Jean Piaget - Kognitivismen (lära sig A, B, A + B -> C, alternativt att eleven utmanas med något den trodde var sant, och tvingas omformulera en lösning som stödjer den presenterade situationen).

Skinner - Behaviorism (ex glada tomater vs wall of text? Det är väll egentligen detta som hindrar läsare att läsa en tråkig, tung svårläst text") Vi väver in det som elever inte förknippar med tunga texter -> färre flyktförsök.

3

Metod

Projektets genomförande bestod till störst del av konstruktionen av själva läromaterialet, och där ingick skapandet av domänspecifika språk, skapandet av den tillhörande lärotexten och publiceringen av denna på en hemsida. Därefter följde en mindre del av utvärdering av materialet med en testgrupp.

Konstruktionen av läromaterialet har varit den största delen och har även fortgått genom hela projektets gång. När läromaterialet var halvvägs färdigt skapades en hemsida dit läromaterialet publicerats. Publiceringen skedde sedan dess löpande. I slutet av projektet gjordes utvärderingen, där en testgrupp fick svara på frågor om vad de tyckte om läromaterialet.

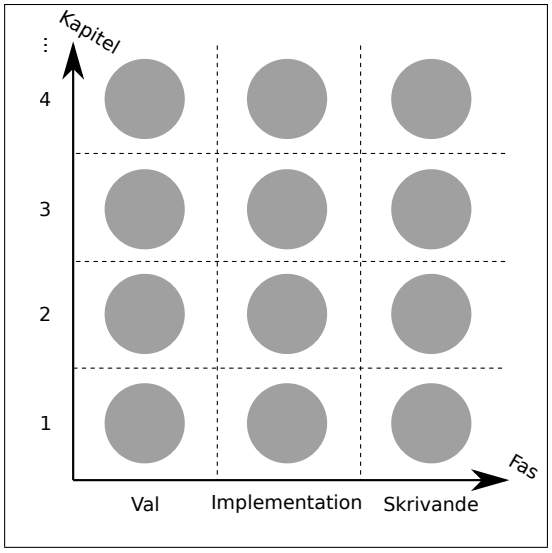
3.1 Konstruktion av läromaterialet

Läromaterialet består av ett antal kapitel som vardera behandlar separata områden. Skapande av varje kapitel skedde därför till största delen fristående från andra kapitel. Skrivandet av alla kapitel bestod i sin tur av tre faser, som såg likadana ut för alla kapitel. Dessa faser var val av område, implementation av domänspecifika språk för området samt skrivande av lärotext.

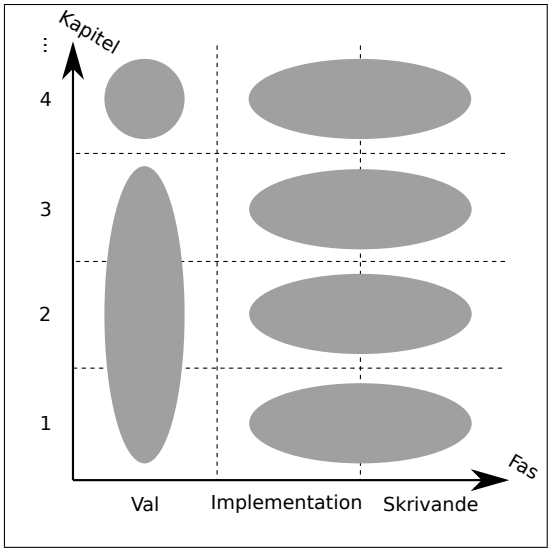
Denna skapandeprocess kan delas upp i två axlar: en utefter kapitel och en utefter fas. Det här illustreras i figur 3.1a. Figuren visar att varje kombination av kapitel och fas är en del i projektet som arbetades med.

Även om detta sätt att dela upp processen är översiktlig är den inte helt verklighetstrogen. I praktiken fanns det överlapp mellan de olika delarna, både med avseende på kapitel och fas. Det här illustereras i figur 3.1b. Där ser man att selektionen av områden skedde för flera kapitel samtidigt. Detta då arbetet med att hitta ett områden ofta gav flera områden samtidigt. I figuren ser man också att implementation av domänspecifika språk och skrivande av lärotext skedde samtidigt. Eftersom de i resultatet är sammanvävda är det högst naturligt att processerna med att skapa dem också var sammanvävda.

De tre följande avsnitten beskriver i mer detalj hur de tre faserna, val, implementation och skrivande såg ut. Det är viktigt att minnas att det, som nämndes ovan, finns överlapp



(a) Delarna visas distinkta och var för sig.



(b) Delarna visas med gränsöverskridande överlapp. Notera att implementation och skrivande *alltid* var överlappande medan selektion *ofta men inte alltid* var det.

Figur 3.1: Översikt över hur skapandeprocessen av läromaterialet såg ut. Processen kan delas upp utefter två axlar: kapitel och fas. Varje kombination är en del som arbetats med och är gråmarkerad.

både mellan faserna och mellan kapitlena.

3.1.1 Selektion av fysikaliska områden att behandla

Ett domänspecifikt språk modellerar ett specifikt och avgränsat område. Därför var det naturligt att söka och tänka i termer av avgränsade områden inom fysiken. För att mer konkret hitta områden att behandla kontaktades Åke Fäldt, examinator för *Fysik för ingenjörer*, samt så studerades fysikkursens bok, *Univeristy Physics*.

Fäldt befrågades om vilka områden han i allmänhet tycker studenter verkar ha svårt för. Detta för att i enlighet med projektets mål börja med de, för studenterna, problematiska områdena. Enligt Fäldt är ett allmänt problem att egna mentala modeller för problem är felaktiga eftersom studenter ofta tar genvägar, som inte bygger på saker man är säker gäller. En annan erfarenhet från honom är att så länge första raden i en uppgiftslösning är rätt, så är resten också rätt. Med andra ord, har studenten väl identifierat vilken typ av problem det rör sig om brukar det inte vara några svårigheter att lösa uppgiften.

Med hjälp av insikterna från Fäldt drogs två slutsatser. Den första slutsatsen var att matematisk analys var ett område värt att behandla i detalj. Den andra slutsatsen var att genom att ge struktur till olika typer av problem kan det förhoppningsvis underlätta för studenter att lära sig identifiera vilken typ av uppgift de handskas med.

En inledande selektion gjordes för att hitta mer specifika områden att arbeta. Detta

gjordes genom att studerade kursboken *University Physics*. Speciellt av intresse var de kapitel som behandlade mekanik (i enlighet med projektets mål att börja med klassisk mekanik), matematisk analys samt de kapitel som använde sig av en specifik syntax. Domänspecifik syntax var av intresse att finna då en betydlig del av domänspecifika språk är modellering av just syntaxen. Även områden som projektgruppen fann personligen intressanta, valdes ut.

Kanske förtydliga/utveckla varför detta är av intresse

Skriv om

Allteftersom arbetet fortskred blev det tydligt att det fanns en distinktion mellan *grundläggande* och *komposita* områden. De grundläggande områdena är fristående och lägger grunden till mer komplexa områden och är i sig (dimensioner undantaget) inte av direkt intresse för fysik. De krävs däremot som ett bakomliggande matematiskt ramverk till de komposita områdena. De komposita områdena skiljer sig genom att vara närmare fysiken, vara av tillämpad natur och bygga vidare på mer underliggande koncept. Ett exempel är rörelseproblem som använder sig av både matematisk analys och dimensioner.

Grundläggande områden

Skriv om paragraf nedan

Grundläggande områden hittades främst genom den tidigare nämnda inledande selektionen (se avsnitt 3.1.1). Det omvända förhållandet gäller också. Den inledande selektionen gav främst grundläggande områden.

Det är väl inte om vänt, utan samma sak?

Fler grundläggande områden har hittats i efterhand. Avslutande av den inledande selektionen innebar alltså inget stopp för sökandet av grundläggande områden, även om de flesta redan hittats . Fler grundläggande områden hittades bland annat även genom att studera kursmaterialet som tillhörde *Fysik för ingenjörer*.

Hur vet vi att vi redan har hittat de flesta?

Sökandet i kursboken och kursmaterialet gav viktiga kunskaper om områden att behandla. Men minst lika viktiga var de experiment som gjordes för att se huruvida ett område lämpade sig till att göra ett domänspecifikt språk av. När områden kodades upp till domänspecifika språk var det långt ifrån alla gånger det kunde göras på en meningsfullt sätt . De områden som inte gick att modellera fick skrotas eller behandlas annorlunda (se ”Komposita områden” nedan). Förenklat sagt var de områden som innehöll tydliga data och operationer väl lämpade att för domänspecifika språk. Lämpligheten hos olika områden diskuteras utförligare i avsnitt 5.3.2.

Vad betyder meningsfullt?

När de grundläggande områdena valdes ut definerades de så att det i största möjliga utsträckning var fristående. Detta för att kunna arbetas med parallellt. De grundläggande områdena som under projekets gång valdes ut blev bevis, dimensioner, matematisk analys och vektorer.

OCH FLER?

Bevis eftersom det ger en insikt i hur de formler man använder faktiskt uppstår. För att genomföra bevis krävs också att formler och beteckningar görs rigorösa, vilket ger en bättre förståelse av dem.

Dimensioner eftersom det är viktigt för studenter att förstå sig på hur dimensioner påverkas av algebraiska operationer . Det kan också vara hjälpsamt att kunna utföra automatisk, datorassisterad dimensionsanalys på beräkningar.

arför är det vik-
gt?

Matematisk analys eftersom alla koncept i klassisk mekanik är relaterade genom matematisk analys. Mer specifikt används differenser för att beskriva medelrörelse, och infinitesimalkalkyl för att beskriva momentanrörelser. Vidare var infinitesimalkalkyl just det område som Fäldt pekade ut som speciellt viktigt och något som studenter har svårt för.

Vektorer eftersom det är en viktig grundsten inom mekanik. Alla krafter, hastigheter och accelerationer betraktas som vektorer i planet eller rummet, och dessa är alla fundamentala element inom klassisk mekanik.

OCH FLER

Komposita områden

Komposita områden uppstod då det framgick att inte alla områden var lämpliga att göra domänspecifika språk av (se avsnitt 5.3.2). Eller om området var en vidareutveckling och/eller kombination av de grundläggande områdena. Lösningen blev därför att tillämpa domänspecifika språk från de grundläggande områdena för att modellera dessa.

Ett sätt som komposita områden hittades var därmed som ett resultat av sökandet och experimenterandet med grundläggande områden. Ett annat sätt var att även komposita områden direkt söktes efter. Då användes precis som tidigare främst kursboken och kursmaterialet tillhörande *Fysik för ingenjörer*. Komposita områden inkluderades för att täcka områden med mer direkt koppling till fysik. Till exempel har lutande plan en mer direkt koppling än vad analys har.

Skriv om som ett exempel på ett kompositområde

Momentan- och medel-rörelse eftersom de direkt utgör en stor delmängd av alla problem inom mekanik i *Fysik för ingenjörer*. Väldigt många av de uppgifter studenter lär sig lösa inom mekanik är sträcka/hastighet/acceleration/kraft problem. Hur lång tid tar det att åka en sträcka om man har en viss medelhastighet? Om ett objekt med massa m påverkas av en kraft som varierar enligt $sin(t)$, vad är då momentanhastigheten vid $t = 10$?

3.1.2 Implementation av DSL för områdena

Implementationen av ett domänspecifikt språk till ett område började alltid med experimentering. Även om en del experiment redan gjorts i selektionsfasen behövdes mer göras.

Det finns ingen absolut kanonisk form att skriva ett domänspecifikt språk på, framförallt inte för fysik . Experimentering var därför viktigt för att finna lämpliga representationer av syntax och andra element av domänspecifika språk.

förklara kanonisk form

Varför är det fram förallt för fysik?

Framförallt eftersom vi inte vet hur den kanoniska formen ska se ut, det finns ingen almänn bestämd form. Och det är också det som är motivationen till vårt projekt.

Utveckla och skriv om

Vad som ansågs vara en bra, eller åtminstone tillräckligt bra, representation var i huvudsak baserat på gruppmedlemmarnas intuition om Haskell och diskussion med handledaren. Implementationerna skulle vara både naturliga, i den mening att de gick att använda smidigt rent tekniskt. Men de skulle även vara lättförståliga. Den progamtekniskt elegantaste implementationen användes därför inte alltid, utan den mer verbosa versionen föredrogs för att göra läromaterialet så lättläst som möjligt. Dock avstod vi inte från de mer avancerade funktionerna i Haskell när de var motiverade av materialet som vi beskrev, men då alltid med en utömmande förklaring av hur det fungerade och utan krav på tidigare kunskap hos läsaren.

De olika områdena krävde i varierande mån inläsning och studering av Haskell, fysik, matematik och domänspecifika språk. Till exempel krävde kapitlet om dimensioner inläsning om typnivå-programmering i Haskell.

HÄR BEHÖVER JAG HJÄLP ATT EXPANDERA JOHANS STYCKEN

I allmänhet implementerades DSL i Haskell som en kombination av syntaxträd, funktioner för att manipulera dessa träd, och evalueringsfunktioner till något slags semantisk domän. Komposita DSL var istället mer . Varje DSL, grundläggande och komposita, parades även ihop med en mängd fysikproblem att appliceras på, sådant att läsaren skulle få förståelse för hur DSLen kan brukas utöver hur de implementerades.

Vad är ett seman- tiskt domän

beskrivning

HUR DETALJERAT SKA NEDANSTÅENDE VARA? FRÅGA PATRIK OCKSÅ

Syntax analyserades och modellerades i syntaxträd. För vissa områden (enheter?) definierades även ett semantiskt värde som representerade en slags kanonisk form.

För vektorer gjordes ...

För enheter gjordes ...

För analys gjordes ...

För annat gjordes ...

TESTER!!! (quickcheck)

Skriv

Saker som vi väntar en stund med att skriva

KOMPOSITA ASPEKTER

Till de komposita områdena implementerades inga domänspecifika språk. Istället användes de tidigare språken som skapats för de grundläggande områdena. SKRIV MER HÄR NÄR VI GJORT NÅGRA KOMPOSITA

Importera DSLerna för varandra för att göra mer komplicerade grejer.

Eller, *Bruk av de mer fundamentala/teoretiska DSLerna för att angripa områden av mer "tillämpad" natur (såsom Krafter, Arbete, etc))*(?)

3.1.3 Skriva lärotext

I samband med att ett område implementerades skrevs lärotext till det. Till en början skrevs brödtext som fokuserade på att förklara programkoden. Detta var ofta ett naturligt val . Det var viktigt att programkoden gick att förstå innan kopplingar till matematik och fysik kunde förklaras. Det var nämligen brödtext av det slaget som skrevs senare. Avslutningsvis skrevs inledning och avslutning till kapitlet.

arför var det naurligt?

Generellt under skrivningen av lärotext togs det hänsyn till ARCS som presenterats i avsnitt 2.7.

TODO: skriv mer här när vet om ARCS.

Lättsamt språk och en gnutta humor för att hålla kvar uppmärksamhet. Relaterat till Attention i ARCS modellen (2016 använde den. Såg vettig ut).

TODO: Expandera. Vad exakt använder vi för didaktisk metod? Samma som appliceras i Learn You a Haskell, mer eller mindre. Fånga uppmärksamhet med lite humor och lättsamhet etc.

Brödtexten och programkoden skrevs sammanvävt i samma fil, i *Literate Haskell*, se avsnitt 2.6. Literat programmering passade bra ihop med hur läromaterialet skulle se ut då det betonade det jämnbördiga förhållandet mellan programkod och förklaringar. För att läromaterialet skulle vara lättförståeligt var det också viktigt att presentera materialet i den ordning som en mänsklig läsare, och inte datorn, tyckte var enklast.

Quantity så är etta tydligt. Då jordes en "taste f types" tidigt för tt läsaren skul- förstå Quantity ättre, innan mas- detaljer gicks in å. Frågan är dock m detta ska gås på i rapporten?

TODO: Skriva om övningar.

KOMPOSITA ASPEKTER

Vari vi visar att DSLerna både är praktiskt användbara, likt Wolfram Alpha, och att implementationen+applikationen hjälper oss förstå mekanik i allmänhet och probleminstanserna i synnerhet.

3.2 Skapande av och publicering på hemsidan

Läromaterialet publiceras på en internethemsida, varpå man kan läsa allt o ha skoj.

TODO: Slå ihop underrubriker till ett stycke.

3.2.1 Beskrivning

Ett build-script hämtar .lhs källfilerna, i vilka lärotexten är skriven med markdown. Rendrar med pandoc, och sätter in lite navigationselement etc. med hjälp av eget templating-system. Manuellt läggs sedan stoffet på gh-pages branchen för att automatiskt visas på dslsofmath.github.io/BScProj2018.

Förklara templating, rendrar, branch

Obs: Medan bygget är scriptat så är inte publiceringen det, och ingenting genereras/publiceras automatiskt kontinuerligt. Måste köra scriptet manuellt och lägga stoff på gh-pages branchen.

3.2.2 Build-script

I.e. implementation av python-build-scriptet i mer detalj.

TODO: Är detta ens intressant? Viktigt för att producera sidan såklart, men inte intressant ur varken matte eller haskell/DSL perspektiv.

3.2.3 Hemsidan

TODO: Nåt om design, läslighet, grafik(?), navigation, avsiktligt undvikande av javascript, etc.

Från resultat: ska integreras här

Hemsidan består av grundläggande HTML, CSS och javascript. På hemsidan finns en innehållsförteckning med klickbara länkar till de olika kapitlen. Hemsidan är öppen för alla och bör fungera i de flesta webläsare. Javascript är inget krav för hemsidan. Matematiska formler visas ändå, om än inte lika tydligt.

3.3 Test och återkoppling

Test på försöksstudenter. Återkoppling med Åke(?).

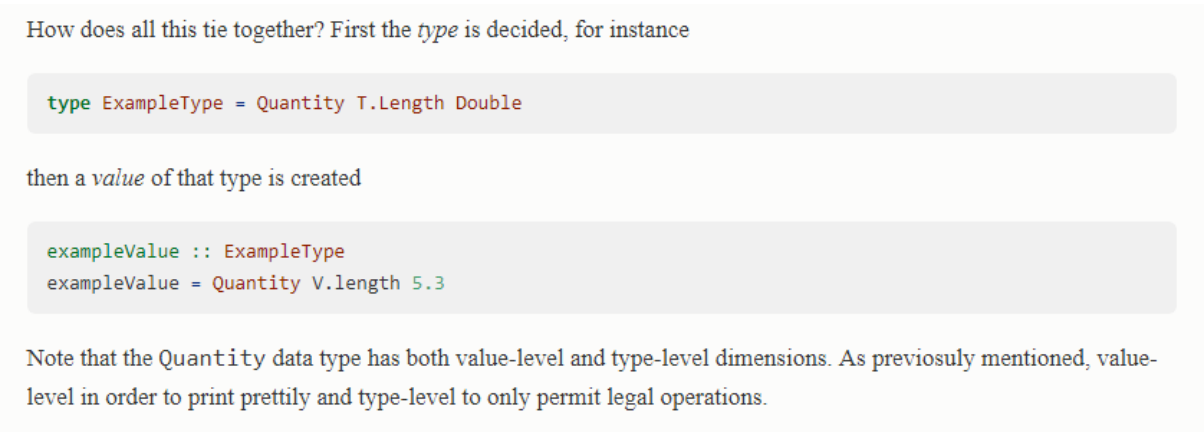
Nog bra att vara explicit här med att det inte är en rigorös empirisk studio, om inte det redan täckts väl i Avgränsningar.

4

Resultat

4.1 Läromaterialet

Läromaterialet blev i slutändan en sammanvävning av domänspecifika språk som modellerar fysik, och lärotext som förklarar kopplingen mellan fysiken och de domänspecifika språken. Figur 4.1 visar ett kort utdrag ur läromaterialet. Där ser man hur domänspecifika språk och lärotext är sammanvävda.


How does all this tie together? First the *type* is decided, for instance

```
type ExampleType = Quantity T.Length Double
```

then a *value* of that type is created

```
exampleValue :: ExampleType
exampleValue = Quantity V.length 5.3
```

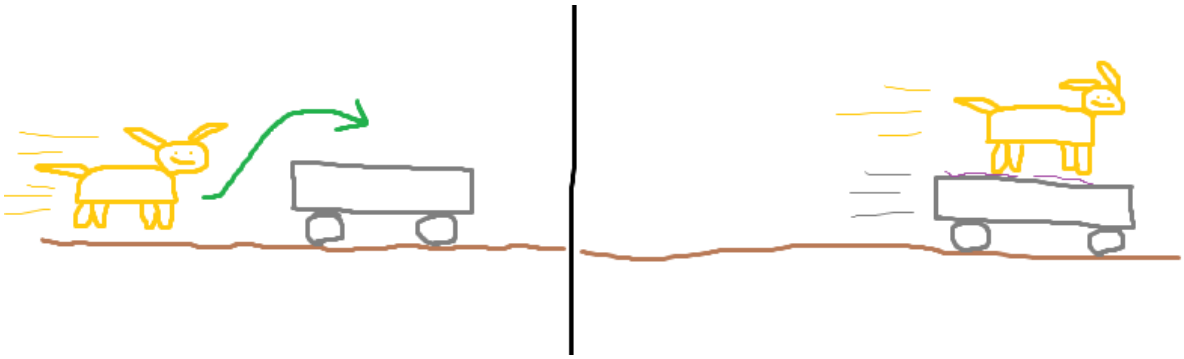
Note that the `Quantity` data type has both value-level and type-level dimensions. As previosuly mentioned, value-level in order to print prettily and type-level to only permit legal operations.

Figur 4.1: Ett smakprov över hur det resulterande läromaterialet ser ut. Lärotext är framför den ljusgrå bakgrunden medan Haskell-kod för domänspecifika språk är framför den mörkgrå.

I läromaterialet finns även bilder. Figur 4.2 är ett exempel på en bild ur läromaterialet. Speciellt att notera är den medvetet oseriösa ritningstekniken som är tänkt att vara rolig och muntra upp läsaren.

Läromatateriale behandlar ett flertal områden inom fysik och matematik som används inom fysik. Fokuset är på klassisk mekanik samt till det området tillhörande matematik. I sin fullständighet är de behandlade områdena

- Bevis
- Dimensioner
- Matematisk analys
- Vektorer



Figur 4.2: Exempel på en bild ur läromaterialet. Bilden visar hur en hund springer och hoppar upp på en stillastående vagn. När hunden landat rör sig hunden och vagnen med en ny, gemensam hastighet.

- Tillämpningar av ovanstående

I *bevis*-kapitlet presenteras bevisföringen med hjälp av Haskells typsystem. Det exemplifieras genom att kinematiska formler bevisas.

Dimensioner behandlar dimensioner, storheter och enheter inom fysiken. Dimensioner införs på typnivå i Haskell för att kunna visa på likheten mellan Haskells typsystem och hur man måste förhålla sig till dimensioner inom fysiken.

I *matematisk analys* konstrueras ett syntaxträd för algebraiska uttryck och funktioner. Därefter implementeras symbolisk derivering och integrering på syntaxträdet.

Vektorer implementerar vektorer och vektoroperationer i Haskell. Lagar som ska gälla för vektorer implementeras och testas.

I läromatateriale finns, förutom de fyra ovanstående grundläggande områdena, även tillämpningar av dem på exempelproblem. Till exempel används dimensioner till att lösa problem med fritt fall. De tillämpningar som behandlats är

- Gundbräda
- Krafter på lådor

Läromaterialet blev publicerat på en hemsida[11] och all källkod finns tillgänglig på projektets GitHub-repository.[12] Texten är skriven på engelska.

4.2 Återkoppling från testgrupp

För att utvärdera huruvida läromaterialet är intressant och hjälpsamt har vi haft en informell återkoppling med en testgrupp. TODO: Skriv mer när vi faktiskt gjort detta.

5

Diskussion

De två nedanstående ska vara nånstans i diskussion.

De domänspecifika språken modellerar områden snarare än att vara problemlösare. *Analys* exemplifierar detta väl. Det språket består av ett syntaxträd över algebraiska uttryck samt operationer som derivering och integration. Med hjälp av det kan man modellera uttryck och analytiska operationer på dem. Däremot löser det inte problem åt en. Man kan med andra ord inte mata in en differentialekvation och automatiskt få en lösning.

Hur bra våra moduler blev samt varför de inte är problemlösare utan modellerare istället.

5.1 Blev vi nöjda med läromaterialet?

blablabla

5.2 Tillvägagångssättet av skapandet

Tanke: I metod/teori varför vi valde att just Haskell o.s.v. Beskrivning av vårt iterativa arbetssätt, hur vi sökte i mörkret. Här i diskussion kanske mer varför det var nödvändigt...

Annan tanke: antingen separat resultat-kapitel för misslyckadeförsök, som DSL för lutande plan, eller att vi har det här..

Vi har kanske inte varit så strukturerade, utan bara valt ut något område på måfå som vi kände för, t.ex. valet av termodynmik.

Blev så för svårt att veta vad som DSL lämpligt för.

Största svårigheten var att välja områden som skulla passa bra för DSL. Som att famla i mörkret.

Kan skriva om misslyckade försök att skapa DSLer för vissa saker. T.ex. lutande plan (DSL för det, ej problemlösning av det området med andra DSL:er), bevis med Haskells typsystem...

5.3 Domänspecifika språk och fysik

Hur en kombination av domänspecifika språk och fysik ser ut är inget uppenbart. Går det att kombinera de två, och för vilka områden fungerar det bra för? Slutligen, är det lärorikt att presentera fysik ur ett domänspecifikt-språk-perspektiv?

5.3.1 Om läromaterialets fokus på matematik och Haskell snarare än fysik

En stor del av läromaterialets fokus ligger på matematik och Haskell snarare än fysik. Detta trots att läromaterialet är tänkt att handla om fysik. Varför är det så?

En orsak till att ett stort fokus ligger på matematik är att fysik egentligen bara är räkneexempel i matematik. Ta cirkulär centralrörelse som exempel. Att beskriva en sådan rörelse hos en kropp görs med hjälp av vektorer och matematisk analys. Fysik finns bara som en bakomliggande teori som dikterar vilka samband som gäller. Själva räknandet görs sedan med matematik.

Detta är inte hela sanningen. Inom fysik ingår en stor del problemlösning, till exempel att sätta upp och hitta krafter i mekanikproblem, utan att någon matematik behöver blandas in. Denna typ av fysik är dock svår att göra domänspecifika språk av, något som diskuteras djupare i avsnitt 5.3.2.

Läromaterialet innehåller även en hel del förklaringar av Haskell-teknisk karaktär. Orsaken till detta är tvåfaldigt. För det första används avancerade koncept inom Haskell som läsaren inte förväntas kunna sedan innan. Ett exempel är typnivå-programmering. För att göra kopplingen mellan Haskell och fysik så tydlig som möjlig behövdes fysikaliska dimensioner kunna likställas med, och hanteras på samma sätt, som Haskell-typer.

För det andra är läromaterialets syfte inte enbart att lära ut fysik i sig, utan även visa hur fysik kan modelleras i domänspecifika språk och dra paralleller mellan fysik och domänspecifika språk. Av detta skäl blir det naturligt att det ”även” ägnas en hel del förklaringar åt programkoden. Syftet är att väcka intresse för fysik, inte att lära ut fysik. Ett sätt att göra det är då att visa dessa paralleller.

Är det rätt eller fel att ha en relativt liten tyngd på fysiken som projekets läromaterial har? På den frågan har vi mer än ett svar. Man kan svara ja, med hänvisning till de ovanstående skälen att det direkta syftet inte är att lära ut fysik utan att väcka intresse. Man kan svara nej, och hävda att de domänspecifika språken kan utformas på ett annat

sätt som mer direkt knyter an till fysikaliska koncept, till exempel ett syntaxträd för sammansättningen av ett lutande plans komponenter. Vi tyckte dock det var svårt att göra på det sättet, se avsnitt 5.3.2. Detta är dock en vidareutvecklingspotential, att undersöka huruvida det går att göra på ett bra sätt.

Man kan svara nej på frågan av ett annat skäl också, nämligen att fysiken i sig bör vara det främsta fokuset. Detta skulle dock innebära en helt annan form på läromaterialet. Vi menar att det skulle vara svårt att väva in domänspecifika språk på ett meningsfullt sätt om de samtidigt skulle vara i skymundan. Ett läromaterial med större fokus på fysik bör i så fall enbart handla om fysik. En utförligare diskussion om detta finns i avsnitt 5.3.3.

5.3.2 Vad för slags områden är DSLs lämpligt att göra för?

Under projektets gång gjordes flera experiment för att se hur man kunde göra ett domänspecifikt språk för ett område. Det visade sig att visa områden var lämpligare än andra. Till exempel var analys och vektorer områden som vi ansåg väl lämpade, medan andra som till exempel lutande plan och termodynamik var mindre lämpade.

Vad har de lämpliga områdena gemensamt? De gemensamma dragen är att de är matematiska, har en fix struktur och består av ”data och operationer”. Tabell 5.1 visar några exempel på områden med sina data och operationer.

Tabell 5.1: Exempel på data och operationer i några domänspecifika språk.

DSL / data	Exempel på operationer
Dimensioner	Multiplikation, division
Vektorer	Addition, skalärprodukt
Analys, funktioner	Derivera, multiplicera

Varför gör dessa drag att de är lämpliga att göra domänspecifika språk för? TODO: Matematiska: man modellerar syntaxen

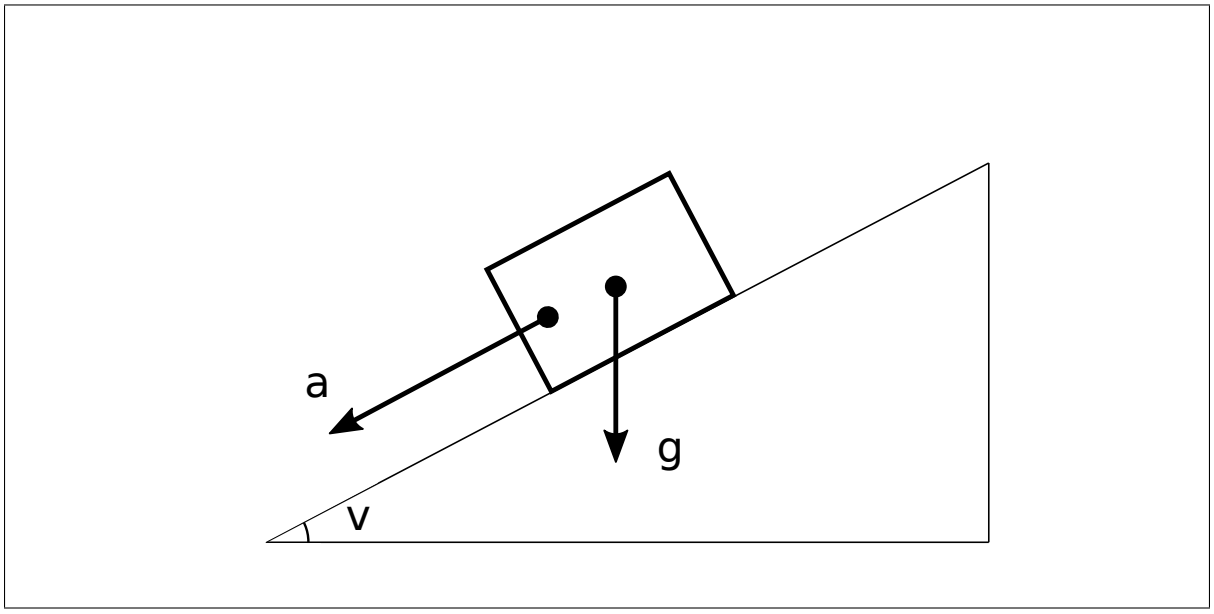
Den fixa strukturen kombinerat med data och operationer gör det enkelt att modellera med datatyper i Haskell. Datatyper har nämligen också en fix form. Dessutom blir relationen mellan å ena sidan data i fysik och datatyper i Haskell, å andra sidan operationer i fysik och funktioner i Haskell, väldigt tydlig. En operation/funktion resulterar sedan i data av samma slag som innan.

Denna strukturella likhet gör modellerandet och manipulerandet av data enkel att genomföra rent tekniskt. Men den innebär också en pedagogisk vinst. Genom att ha strukturerat upp fysik tydligt i Haskell blir det förhoppningsvis enklare för läsaren att förstå hur datan hänger ihop rent fysikaliskt.

I kontrast till dessa lämpliga områden står mindre lämpliga områden (eller åtminstone som vi inte lyckades göra något bra till). Som nämndes tidigare är termodynamik och

lutande plan exempel på mindre lämpliga områden. Vad har dessa områden för gemensamma drag?

Ett gemensamt drag som gör dem svåra att göra domänspecifika språk av är att de består av en samling teoretiska samband. För det lutande planet är det till exempel $a = g * sin(v)$, som illusteras i figur 5.1.



Figur 5.1: Den variant av lutande plan som referas till i exemplet i texten. a är en lådas acceleration längs med planet, g är tyngdacceleration och v är vinkeln. Friktionen antas vara försumbar.

Teoretiska samband av det slaget relaterar olika egenskaper i systemet. Visserligen kan man modellera samband och ekvationer som ett domänspecifikt språk, men vi menar att nyttan inte blir stor med det. Det man kan göra är att programmera en ekvationslösare. Men den hade behövt vara både mekanisk och komplex. Den skulle alltså skilja sig drastiskt från hur man löser problem för hand och skulle vara svår att förstå. Alldeles för mycket fokus skulle hamna på algoritmer istället för fysik.

När det kommer till lutande plan och liknande områden är nyckeln att visserligen känna till vilka samband som gäller, men det framförallt att veta när man ska använda dem och hur man tillämpar dem på olika typer av uppgifter. Vi behandlar därför områden som lutande plan genom att lösa exempeluppgifter modellerade i de tidigare domänspecika språk. De tidigare språken tillhandahåller de matematiska verktyg som behövs för att koda upp lösningar av problem.

TODO: Kanske avsluta med att poängtera skillnaderna mellan lämpliga och mindre lämpliga.

Och vad är skillnaden mellan dessa två kategorier av områden?

Den viktiga skillnaden är att t.ex. analys har tydlig data och operationer medan problemlösning som lutande plan har ett gäng samband som man använder beroende på

behov.

5.3.3 Gör DSLs så att fysik blir enklare att förstå?

Ex med lutande plan: man kan betrakta ett sådant problem som en samling ekvationer. Man har några kända värden och med hjälp av ekvationerna ska man hitta den sökta obekanta. Hjälper verkligen ett DSL till att man blir bättre på detta?

Med enheter, räcker det inte att förklara skillnaden mellan storheter, dimensioner och enheter på ett så grundligt sätt vi gjorde, utan att blanda in DSLs? Problemet i fysik för Data kanske är att det inte förklras grundligt och mycket är underförsått. Behöver man ens förstå enheter grundligare för att klara fysik bättre?

Man kan också se det som att DSLs och denna extrakunskap vi presenterat är för att göra fysik intressant genom att visa på vad för kopplingar till programmering man kan göra, även om områdena vi behandlat inte är direkt de områden som man behöver förstå för att klara kursen. Projektet kan ses som ren kuriosa som kan vara intressant.

5.4 Inte tänkt så mycket på didaktik

Fanns mycket i ARCS vi inte använder. Kanske därför inte kan förvänta oss att så bra pedagogiskt.

Men projekets fokus var snarare av teknisk karaktär.

Egentligen har det väl varit skapandet som varit det lärorika och inte användingen av det vi gjort?

5.5 Vidareutvecklingsmöjligheter

På något sätt DSL (och inte bara tillämpning av andra) för lutande plan, termodynamik osv. Ett DSL för ”fysikaliskt problemlösning” i sig. Ingen aning hur ett sådant skulle se ut, men skulle säkert vara intressant.

Det finns många områden inom fysik som vi inte täckt in. Finns möjlighet för ett annat projekt att ta vid där vi avslutade.

Att göra en ordentlig studie om DSLs+fysik gör fysik enklare.

5.6 Etiska aspekter

Hemsida med grundläggande html, css och frivilligt javascript. Anapassat för små datorskärmar. Även de med 10-tums laptop ska kunna ha LYAP på ena halvan av skärmen och ändå se fullständigt.

Engelska

Roligt

Öppen källkod

TODO: Varför en hemsida istället för PDF i diskussion om etik?

Om Hemsidan: Det är lite intressant ur pedagogik-aspekten. Kan det kanske vara lättare/roligare att lära sig om sidan är fin och lättläst? Att javascript inte krävs gör att sidan kan visas ordentligt även om man sitter i U-land med dålig/gammal/billig telefon.

6

Slutsatser

Beskriv vad vi kom fram till i diskussionen. Varför är det svårt med vissa områden. Tips till dem som vill testa något liknande.

Litteraturförteckning

[1] G. Johansson, "Statistik över kursresultat", 2018. [Online]. Tillgänglig: http://document.chalmers.se/doc/00000000-0000-0000-0000-00001C968DC6, hämtad: 2018-01-30.

[2] D. Ghosh, "DSL for the Uninitiated", *Queue*, vol. 9, nr. 6, s. 10-21, jun. 2011. [Online]. Tillgänglig: https://doi.org/10.1145/1989748.1989750, hämtad: 2018-03-07.

[3] C. Ionescu och P. Jansson, "Domain-Specific Languages of Mathematics: Presenting Mathematical Analysis Using Functional Programming", i *Proceedings of the 4th and 5th International Workshop on Trends in Functional Programming in Education*, Sophia-Antipolis, France and University of Maryland College Park, USA, 2015, ss. 1-15. [Online]. Tillgänglig: https://doi.org/10.4204/EPTCS.230.1, hämtad: 2018-01-30.

[4] C. Ionescu och P. Jansson, "Domain Specific Languages of Mathematics: Lecture Notes", 2018. [Online]. Tillgänglig: https://github.com/DSLsofMath/DSLsofMath/tree/master/L/snapshots, hämtad: 2018-01-30.

[5] G. J. Sussman och J. Wisdom, "Classical Mechanics: A Computational Approach", 2008. [Online]. Tillgänglig: urlhttps://ocw.mit.edu/courses/earth-atmospheric-and-planetary-sciences/12-620j-classical-mechanics-a-computational-approach-fall-2008/, hämtad: 2018-01-30.

[6] G. J. Sussman, J. Wisdom och M. E. Mayer, *Structure and Interpretation of Classical Mechanics*. Cambridge, MA, USA: MIT, 2001. [Online]. Tillgänglig: https://mitpress.mit.edu/sites/default/files/titles/content/sicm/book.html, hämtad: 2018-03-07.

[7] F. Lindahl, C. Rosvall, P. Ngo, J. Jonsson och J. Olsson, "Programmering som undervisningsverktyg för Transformer, signaler och system: Utveckling av läromaterialet TSS med DSL", Chalmers tekniska högskola, Göteborg, Sverige, 2016.

[8] D. Knuth, "Literate Programming", *The Computer Journal*, vol. 27, nr. 2, ss. 97-111, jan. 1984. [Online]. Tillgänglig: https://doi.org/10.1093/comjnl/27.2.97, hämtad: 2018-03-14.

[9] Haskell Wiki, “Literate Programming”, 2017. [Online]. Tillgänglig: https://wiki.haskell.org/Literate_programming, hämtad: 2018-03-14.

[10] Pandoc, “Pandoc: a universal document converter”, 2018. [Online]. Tillgänglig: https://pandoc.org/, hämtad: 2018-03-14.

[11] B. Werner, E. Sjöström, J. Johansson och O. Lundström, “Learn You a Physics for Great Good!”, 2018. [Online]. Tillgänglig: https://dslsofmath.github.io/BScProj2018/, hämtad: 2018-03-14.

[12] B. Werner, E. Sjöström, J. Johansson och O. Lundström, “BScProj2018”, 2018. [Online]. Tillgänglig: https://github.com/DSLsofMath/BScProj2018, hämtad: 2018-03-14.

A

Bilaga λ

Inläsning

- Identifikation av problemområden.
  - Kontakt med Åke Fäldt och DNS. Studera kursutvärderingar.
  - Reflektera över vad vi själva tyckt varit svåra områden då vi läst kursen.
- Studerande av existerande läromaterial, både inom ren fysik och liknande vårt material.
  - Fysikboken.
  - Åke Fäldts egna material.
  - Boken *Structure Interpretaton of Classical Mechanics*[6].
  - Kursboken till kursen *Matematikens domänspecifika språk*.
- Existerande implementationer.
  - OpenTA.
  - Hamilton.
  - MasteringPhysics.
- Tidigare forskning.
  - Cezar och Patriks 2015 forskningsartikel.
  - 2016 års kandidatarbete.
  - Artikeln *DSL for the Uninitiated*.
  - *Communicating Mathematics: Useful Ideas from Computer Science*

Implementation av domänspecifika språk

Vid implementationen av ett/flera domänspecika språk behöver nedanstående punkter genomföras.

- Hitta relevanta grundtyper inom fysik, exempelvis sträcka och massa.
- Hitta relevanta kompositttyper, exempelvis hastighet och tryck.
- Utförligt typsystem.
- Dimensionskontroll.
- Modellera fysikens syntax i språket.

- Pedagogiska syntaxträd.
- Kombinatorer och konstruktorer.
- Hålla våra typer polymorfa.

Skrivande av läromaterial

Vid skrivandet av läromaterialet kommer följande punkter ligga till grund.

- En gemensam vokabulär som fungerar när man skriver om både fysik och programmering (generics kontra polymorfism), och som gör det möjligt att prata om dem i samma mening utan att byta språk och på så sätt brygga det semantiska gapet mellan områdena.
- Övningar
  - Modellera ett fysikaliskt problem med vårt domänspecifika språk.
  - Lös ett ”vanligt” fysikaliskt problem med hjälp av vårt domänspecifika språk.
  - Simuleringar i stil med *Bouncing Balls*.
  - Delar av fysiken vi inte behandlat lämnas som övning att själv implementera.
- Gå igenom allmän teori (t.ex. Newtons lagar, krafter som verkar, etc) tillsammans med en parallell utveckling av ett domänspecifikt språk.
- Materialet ska vara enkelt att ta till sig.
- Verkligen exponera det DSL som vi gemensamt bygger för att påvisa kopplingen mellan fysik och programmering.