Two Functional MDD's for the Price of One - Part 1

Curtis d'Alves, Nhan Thai, Nassim Khoonkari,
Padma Pasupathi, Tanya Bouman, Christopher Anand

November 6, 2019

Outline

Proto Model Driven Development

- Computer-aided software engineering (CASE) is the domain of software tools used to design and implement applications.
- ISDOS project started in 1968 at the University of Michigan
- Lots of tools.
  - DB-centric tools (e.g., Object Relation Mapping tools)
  - OO-oriented tools (e.g., Eclipse Modeling Framework)
- UML, standardized 1997
  - large-scale processes
  - documenting user interaction
- skeleton generation

Model Driven Development for Numerical Computation

- linear algebra
  - VOP - vector-oriented programming
  - APL - Array Programming Language, 1966
  - Matlab - "mathematics for engineering"
  - Maple - symbolic computation, symbolic code generation
- optimization
  - AMPL, 1985
  - GAMS
- mathematical interface to some very powerful software
- great for design exploration

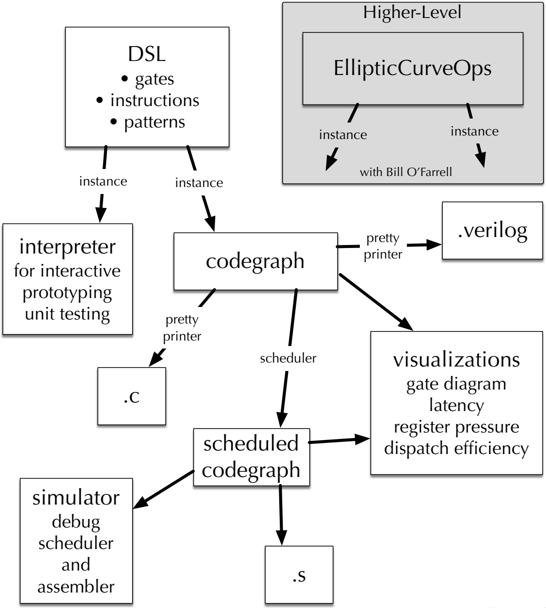
Model Driven Development

- some problems with CASE tools
  - generating skeletons, makes it hard to regenerate
  - suffers same problem as documentation
  - people who can update model are too busy to do so
  - easier to hack on generated code
- new vision (e.g. Selic, 2003)
  - primary focus and products are models rather than computer programs
  - technology-independent specifications
  - requires
    - generating complete programs from models
    - verifying models on a computer
  - looking at code should be as rare as looking at assembly

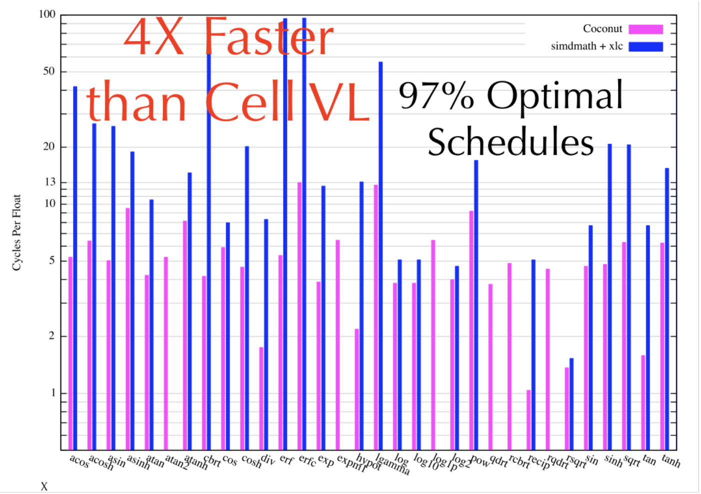
MDD - Coconut

- COde CONstruction User Tool, 2004
  - nested Domain-Specific Languages
  - functional assembly language
  - higher-level patterns
  - principled graph transformations

MDD - Coconut


Higher-Level
EllipticCurveOps
instance
instance
with Bill O'Farrell
DSL
• gates
• instructions
• patterns
instance
instance
interpreter
for interactive
prototyping
unit testing
codegraph
pretty
printer
.verilog
pretty
printer
.c
scheduler
visualizations
gate diagram
latency
register pressure
dispatch efficiency
scheduled
codegraph
simulator
debug
scheduler
and
assembler
.s

MDD - Coconut


4X Faster
than Cell VL
97% Optimal
Schedules
Coconut
simdmath + xlc
Cycles Per Float
100
50
20
13
10
5
2
1
acos
acosh
asin
asinh
atan
atan2
atanh
cbrt
cos
cosh
div
erf
erfc
exp
expm1
hypot
lgamma
log
log10
log1p
log2
pow
qdrt
rcbrt
recip
rqdrt
rsqrt
sin
sinh
sqrt
tan
tanh
X

MDD - Coconut

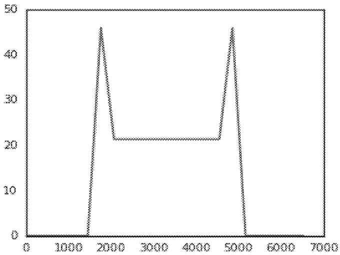
- COde CONstruction User Tool
  - used to generate MASS for all IBM platforms since Cell/B.E.
  - but not used by users outside McMaster / OCA
  - why?
    - have to learn Haskell
    - too many interfaces geared to abandoned research projects
  - fix = fix up the code
    - open source!
    - potential embarrassment will motivate clean up
  - open Coconut in pieces

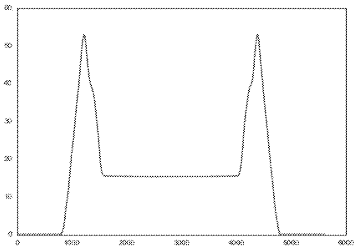
today	HashedExpression	calculus used for continuous optimization
later	CodeGraph library	Intermediate Lanugage
later	DSLs + interpreters	front-ends
later	scheduler + simulator	back-end

Optimization

- maximize benefit
- minimize cost
- satisfy constraints
- make anything better!
- not just instruction scheduling

Optimization - MRI





- big and little magnets manipulate proton spins
- antenna picks up signal
- little magnets controlled by "gradient waveforms"
- common designs (left) are not realizable
- reformulate as an optimization problem (producing right)

Optimization - MRI

- key insight: important waveforms are periodic
- Fourier Transform is discrete
- optimize over finite set of coefficients

$$
\begin{aligned}
\min_{\{h_f : f \in F\}} \quad & \delta && (1)\\
\text{subject to} \quad & |\mu - g(t)| \leq \delta, \qquad \forall t \in S && (2)\\
& |g(t)| \leq g_{\text{peak}}, \qquad \forall t && (3)\\
& |\partial g / \partial t(t)| \leq g_{\text{slew}}, \qquad \forall t && (4)\\
& \int_{t \in S} g(t) = A && (5)\\
\text{where} \quad & g(t) = \sum_{f \in F} h_f \sin(2\pi t f) && (6)
\end{aligned}
$$

Optimization - MRI

- developed with AMPL/neos
- needed parametrized family for production
- reimplement in Python
- want one tool for development and deployment

US 20190064301A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2019/0064301 A1**

Curtis et al. (43) **Pub. Date: Feb. 28, 2019**

(54) **METHOD AND SYSTEM OF FREQUENCY CONSTRAINED GRADIENT WAVEFORM PRODUCTION**

(71) Applicants: **Andrew Thomas Curtis**, Toronto (CA); **Christopher Kumar Anand**, Toronto (CA); **Chad Tyler Harris**, Toronto (CA); **Jeff Alan Stainsby**, Toronto (CA); **Phil J. Beatty**, Toronto (CA)

(72) Inventors: **Andrew Thomas Curtis**, Toronto (CA); **Christopher Kumar Anand**, Toronto (CA); **Chad Tyler Harris**, Toronto (CA); **Jeff Alan Stainsby**, Toronto (CA); **Phil J. Beatty**, Toronto (CA)

(21) Appl. No.: **15/690,795**

(22) Filed: **Aug. 30, 2017**

Publication Classification

(51) **Int. Cl.**
G01R 33/565 (2006.01)
G01R 33/389 (2006.01)

(52) **U.S. Cl.**
CPC ***G01R 33/56509*** (2013.01); ***G01R 33/389*** (2013.01)

(57) **ABSTRACT**

The present disclosure provides a method and system of magnetic resonance imaging using a constrained gradient waveform. The constrained gradient waveform is designed to have only predetermined frequencies, for example excluding one or more identified resonant frequencies associated with the MRI system. The application of such a constrained gradient waveform during imaging may aid in reducing noise, vibration and/or heating of the MRI system during imaging.

Optimizing A Simple Parabola

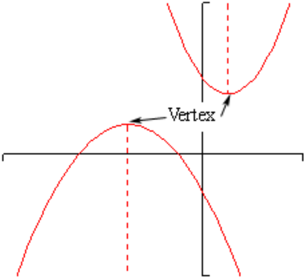


Figure: Example Parabola's: Solve for Vertex

$$\min_{x} \quad ax^2 + bx + c \tag{7}$$
$$\text{subject to } x \leq n \tag{8}$$
$$g(x) \leq m \tag{9}$$

AMPL - MDD Solution for Optimization

```
param a;
param b;
param c;
param n;
param m;

var x >= n;

minimize Obj:
   a * x*x + b * x + c;

subject to G:
   x * x <= m;
```

Figure: file: Quadratic.mod

```
data;

param a := 1 ;
param b := 0 ;
param c := 5 ;
param n := -10 ;
param m := 0 ;
```

Figure: file: Quadratic.dat

Categories of Optimization Problems / Solvers

- Many different categories of optimization
  - Linear / Non-Linear
  - Constrained / Un-Constrained
  - Integer / Mixed Integer
  - Semidefinite, stochastic, ...
- Many, many different solvers to choose from
- No one to rule them all! (Optimization is not Lord of the Rings)
- NEOS Server: fantastic resource for running AMPL models on a variety of different solvers
  `https://neos-server.org/neos/solvers/index.html`

Optimization Without MDD

- Variety of solvers provide implemented in C / Java / Python
- however require you to supply auxiliary functions manually
- commonly required auxillary functions include:
  - objective function
  - gradient of the objective function
  - hessian of the objective function
  - constraint function(s)
  - jacobian of the constraint function(s)

Optimization Without MDD - C Example

```c
void evaluate_partial_derivatives_and_objective() {
  ptr[OBJ] = ptr[A]*ptr[X]*ptr[X] + ptr[B]*ptr[X] + ptr[C];
  ptr[DERIVATIVE] = 2.0 *ptr[A]*ptr[X] + ptr[B];
}
void evaluate_objective() {
  ptr[OBJ] = ptr[A]*ptr[X]*ptr[X] + ptr[B]*ptr[X] + ptr[C];
}
void evaluate_partial_derivatives() {
  ptr[DERIVATIVE] = 2.0*ptr[A]*ptr[X] + ptr[B];
}
void evaluate_scalar_constraints() {
  ptr[CONSTRAINT] = pow(ptr[X],2);
}
void evaluate_scalar_constraints_jacobian() {
  ptr[CONSTRAINT_DERIVATIVE] = 2.0*ptr[X];
}
```

Symphony - Middleground MDD

```
variables:
  x = 0

constants:
  a = 1
  b = 0
  c = 5
  n = -10
  m = 0

constraints:
  x <= n
  x^2 <= m

minimize:
  a * x^2 + b * x + c
```

Why Symphony?

- no need to break out your Calculus 101 textbook
- generates just the auxiliary methods
- solver agnostic, plug into whatever c code you want

Vectors and dimension

- In Symphony, everything is vector.
- Vectors
  - Dimension (Shape): can be scalar, 1D, 2D, 3D, ...
    - Scalar is just a single number
    - 1D(n) variable is an array of n number, useful for problems in signal processing, sound processing, ..
    - 2D(m x n) variable is a 2D array of m x n numbers, useful for problems in image processing, ...
    - 3D(m x n x p) variable is a 3D array of m x n x p numbers, useful for problems in topology, image processing with voxels, ...
  - Numtype: can be real (R) or complex (C)
- We can manipulate vectors like adding, multiplying, doing inner product, ... to form new vectors (expressions).

Forming An Expression

- $(+), (-), (*), (/)$ Add/Subtract/Multiply/Divide (point-wise) two vectors having same shape and same numtype
- $(*.)$ Scale a vector with a scalar (if they form a vector space in Mathematics, i.e, real number can scale anything, but complex can only scale complex)
- $(<.>)$ Inner product (dot product) of two vectors
- $(\hat{})$ Power a vector with an integer
- Piecewise:

```
    case x:
      x <= 0      -> -x
      otherwise -> x

```

- sumElements, norm2square, normHuber

Structure

A valid symphony problem consists of:

- Variables
- Objective function
- Constants (optional)
- Constraints (optional)

Variable Declaration

- Variables are declared in a variable block
- For example:

```
  variables:
    x[100][100] = 10
    y[20][20][20]
    a, b = 2, c

```

- Assignment denotes an initial value
- Unassigned variables will be randomly iniatlized witha number between (0,1)

Objective Function

- Declared in a minimize block
- For example:

```
minimize:
  (x - y)^2

```

- Must evaluate to a scalar (one dimensional value)

Parts II and III

- More Optimization:
  - example: Integrating Blood Flow from PCA-MRI
  - example: MRI Image Reconstruction with Mask
  - example: Parallel MRI
  - example: Logistics
  - Hashed Expression (Symphony's Backend)
- PAL: Petri App Land
  - State Diagrams can model user interaction (grade 5+)
  - PALDraw: graphical modelling tool for State Diagrams
  - make a working app
  - Petri Nets can model multi-user interaction in web apps
  - PALDraw: graphical modelling tool for Petri App Land