Formal Mathematical Languages

Ashish Agarwal

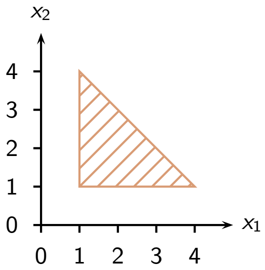
Yale University

CScADS Summer Workshop
Libraries and Autotuning for Petascale Applications
Snowbird, Utah
August 11, 2010

Motivation

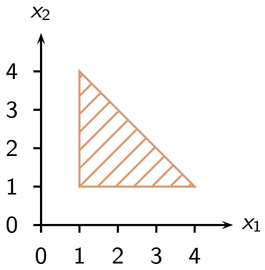
- Bring more of mathematics to more scientists and engineers
- New Language: Express mathematical problems elegantly *and* formally
- Syntactic Transformations: Mechanically generate algorithms

Linear programs (LP)



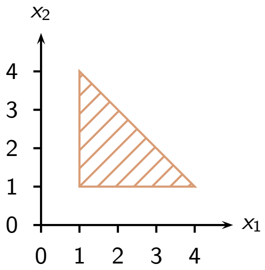
$$x_1 \geq 1.0$$
$$x_2 \geq 1.0$$
$$x_1 + x_2 \leq 5.0$$

Linear programs (LP)



$$
\begin{aligned}
\texttt{max}\ & x_1 - x_2 \\
& x_1 \geq 1.0 \\
& x_2 \geq 1.0 \\
& x_1 + x_2 \leq 5.0
\end{aligned}
$$

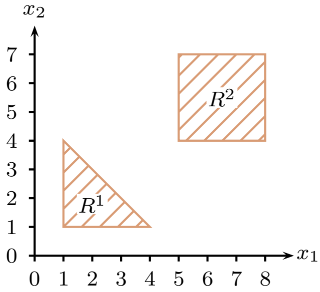
Linear programs (LP)



$$
\begin{aligned}
\texttt{max}\ & x_1 - x_2 \\
& x_1 \geq 1.0 \\
& x_2 \geq 1.0 \\
& x_1 + x_2 \leq 5.0
\end{aligned}
$$

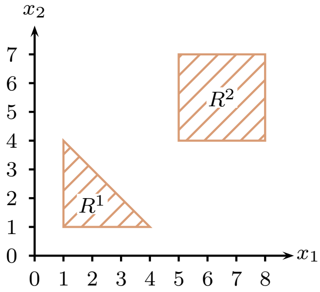
Cannot represent multiple polyhedra.

Declaring discrete choice – with disjunction



$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \quad \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

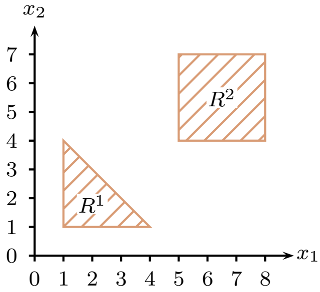
Declaring discrete choice – with disjunction



$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \vee \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

- Language of DP extends LP with disjunction

Declaring discrete choice – with disjunction



$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \vee \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

- Language of DP extends LP with disjunction
- Few algorithms for solving DPs directly.

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

- Basic idea: multiply terms by $y \in \{0, 1\}$

$$0 \leq y \leq 1$$
$$x \leq 3.0y + 2.0(1 - y)$$

- if $y = 1$, then $x \leq 3.0$
- if $y = 0$, then $x \leq 2.0$

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

- Basic idea: multiply terms by $y \in \{0, 1\}$

  $$0 \leq y \leq 1$$
  $$x \leq 3.0y + 2.0(1 - y)$$

- if $y = 1$, then $x \leq 3.0$
- if $y = 0$, then $x \leq 2.0$
- Language of MILP extends LP with the integer type

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

- Basic idea: multiply terms by $y \in \{0, 1\}$

$$0 \leq y \leq 1$$
$$x \leq 3.0y + 2.0(1 - y)$$

- if $y = 1$, then $x \leq 3.0$
- if $y = 0$, then $x \leq 2.0$
- Language of MILP extends LP with the integer type

Goal: Express as DP (intuitive) $\longrightarrow$ Convert to MILP (accepted by solvers)

Example: single disjunctive constraint

Agarwal, Bhat, Gray, Grossman (PADL 2010)

Input

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to
(x <= w) \/ (x >= w + 4.0)
```

Output

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to

exists y1:[0, 1]
exists y2:[0, 1]
exists x1:<0.0, 100.0>
exists x2:<0.0, 100.0>
exists w1:<0.0, 50.0>
exists w2:<0.0, 50.0>
  w = w1 + w2,
  x = x1 + x2,
  y1 + y2 = 1,

  10.0 * y1 <= x1,
  x1 <= 100.0 * y1,
  2.0 * y1 <= w1,
  w1 <= 50.0 * y1,
  x1 <= w1,

  10.0 * y2 <= x2,
  x2 <= 100.0 * y2,
  2.0 * y2 <= w2,
  w2 <= 50.0 * y2,
  x2 >= w2 + 4.0 * y2
```

Example: single disjunctive constraint

Agarwal, Bhat, Gray, Grossman (PADL 2010)

Input

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to
(x <= w) \/ (x >= w + 4.0)
```

- Output generated in MPS and AMPL formats
- Implemented as a DSL embedded in OCaml

Output

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to

exists y1:[0, 1]
exists y2:[0, 1]
exists x1:<0.0, 100.0>
exists x2:<0.0, 100.0>
exists w1:<0.0, 50.0>
exists w2:<0.0, 50.0>
  w = w1 + w2,
  x = x1 + x2,
  y1 + y2 = 1,

  10.0 * y1 <= x1,
  x1 <= 100.0 * y1,
  2.0 * y1 <= w1,
  w1 <= 50.0 * y1,
  x1 <= w1,

  10.0 * y2 <= x2,
  x2 <= 100.0 * y2,
  2.0 * y2 <= w2,
  w2 <= 50.0 * y2,
  x2 >= w2 + 4.0 * y2
```

Switched flow process, comparison to ILOG Concert

Method	#vars (#binary)	#constr. (#IC)	time (sec)
flow-Concert	1061 (874)	1080 (718)	36.85
flow-IC	477 (291)	1001 (438)	11.60
flow-BM	477 (291)	1198	3.37
flow-CH	1194 (631)	2747	1.09

- All 3 of our methods improve on state-of-the-art.

Strip packing, comparison to expert

Method	#vars (#binary)	#constr. (#IC)	time (sec)
pack12-IC	289 (264)	342 (264)	1.83
pack12-BM	289 (264)	342	1.22
pack12-CH	1345 (264)	2718	168.38
pack12-BM-expert	289 (264)	342	1.82
pack12-CH-expert	1345 (264)	1662	149.57
pack21-IC	883 (840)	1071 (840)	24.44
pack21-BM	883 (840)	1071	55.01
pack21-CH	4243 (840)	8631	991.68
pack21-BM-expert	883 (840)	1071	29.56
pack21-CH-expert	4243 (840)	5271	$>$ 3600.00

- Our mechanizations perform just as well as expert encodings.

Indexing

Agarwal (2006)

- We solved a problem with 150,000 equations and 25,000 variables.
- How do you declare so many equations and variables?

Indexing

Agarwal (2006)

- We solved a problem with 150,000 equations and 25,000 variables.
- How do you declare so many equations and variables?
  Use index sets.

Indexing

Agarwal (2006)

- We solved a problem with 150,000 equations and 25,000 variables.
- How do you declare so many equations and variables?
  Use index sets.
- Indexed operators:

$$\sum_{i=1}^{n} x_i$$

- Families of equations:

$$\forall i \in \{1, \ldots, n\} \quad x_{i+1} = x_i + y_i$$

Indexing

Agarwal (2006)

- We solved a problem with 150,000 equations and 25,000 variables.
- How do you declare so many equations and variables?
  Use index sets.
- Indexed operators:

$$\sum_{i=1}^{n} x_i$$

- Families of equations:

$$\forall i \in \{1, \ldots, n\} \quad x_{i+1} = x_i + y_i$$

- Complex index sets in practice, *e.g.* job shop scheduling:

$$\forall j \in J \quad \forall s \in S_j \quad \forall j' \in Pre_{j,s} \quad t_{j',s} \leq t_{j,s}$$

Indexing Is A Generalization of Matrix Notation

- Rows $R = \{1, \ldots, M\}$
- Columns $S = \{1, \ldots, N\}$
- Matrix $A : R \times S \to \mathbb{R}$

Indexing Is A Generalization of Matrix Notation

- Rows $R = \{1, \ldots, M\}$
- Columns $S = \{1, \ldots, N\}$
- Matrix $A : R \times S \rightarrow \mathbb{R}$
- Matrix multiplication:
- Consider vector $x : S \rightarrow \mathbb{R}$
- Then matrix multiplication is a higher-order function

$$\otimes : (R \times S \rightarrow \mathbb{R}) \times (S \rightarrow \mathbb{R}) \rightarrow (R \rightarrow \mathbb{R})$$

Beyond Matrices

```
set JOBS = {a,b,c}

set STAGES(i) = case i of
                    a => {s1,s2}
                  | b => {s1,s3,s4}
                  | c => {s3,s4}

set JOBS_STAGES = i:JOBS * STAGES[i]
```

Explicitly:

```
{(a,s1), (a,s2),
 (b,s1), (b,s3), (b,s4),
 (c,s3), (c,s4)}
```

Beyond Matrices

- Can now define non-rectangular data:

```
A : i:JOBS * STAGES[i] -> real
```

$$A = \begin{bmatrix} A_{a,s1} & A_{a,s2} & & \\ A_{b,s1} & & A_{b,s3} & A_{b,s4} \\ & & A_{c,s3} & A_{c,s4} \end{bmatrix}$$

Beyond Matrices

- Can now define non-rectangular data:

```
A : i:JOBS * STAGES[i] -> real
```

$$A = \left[\begin{array}{cccc} A_{a,s1} & A_{a,s2} & & \\ A_{b,s1} & & A_{b,s3} & A_{b,s4} \\ & & A_{c,s3} & A_{c,s4} \end{array}\right]$$

- Also support:
- Tensors
- Nested matrices
- Can have matrices with some elements as sub-matrices, and some as scalars or matrices of different dimensions

Beyond Matrices

- Can now define non-rectangular data:

```
A : i:JOBS * STAGES[i] -> real
```

$$A = \begin{bmatrix} A_{a,s1} & A_{a,s2} & & \\ A_{b,s1} & & A_{b,s3} & A_{b,s4} \\ & & A_{c,s3} & A_{c,s4} \end{bmatrix}$$

- Also support:
- Tensors
- Nested matrices
- Can have matrices with some elements as sub-matrices, and some as scalars or matrices of different dimensions

Types express exact nature of each value.

Memory Reduction

- Load this program:

$$\forall i \in \{1, \ldots, n\} \quad x_{i+1} = x_i + y_i$$

- Optimization software (AMPL, CPLEX, etc) expand this to:

$$\begin{aligned} x_2 &= x_1 + y_1 \\ x_3 &= x_2 + y_2 \\ x_4 &= x_3 + y_3 \\ x_5 &= x_4 + y_4 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \end{aligned}$$

Memory Reduction

- Load this program:

$$\forall i \in \{1, \ldots, n\} \quad x_{i+1} = x_i + y_i$$

- Optimization software (AMPL, CPLEX, etc) expand this to:

$$\begin{aligned} x_2 &= x_1 + y_1 \\ x_3 &= x_2 + y_2 \\ x_4 &= x_3 + y_3 \\ x_5 &= x_4 + y_4 \\ \vdots \quad & \quad \vdots \quad \vdots \end{aligned}$$

- We retain indexing structure:
  Memory requirements reduced from $O(n)$ to $O(1)$.

Computational Improvements

- Input to our software:

$$\bigvee_{i:[1..10]} w \geq x_i + 4.0$$

- Our software's output:

$$\bigwedge_{i:[1..10]} \begin{bmatrix} 10.0 * y_i \leq w'_i \\ w'_i \leq 90.0 * y_i \\ \bigwedge_{d:[1..10]} \begin{bmatrix} 5.0 * y_i \leq x'_{i,d} \\ x'_{i,d} \leq 75.0 * y_i \end{bmatrix} \\ w'_i \geq x'_{i,i} + 4.0 * y_i \end{bmatrix}$$

Computational Improvements

- Input to our software:

$$\bigvee_{i:[1..10]} w \geq x_i + 4.0$$

- Our software's output:

$$\bigwedge_{i:[1..10]} \begin{bmatrix} 10.0 * y_i \leq w'_i \\ w'_i \leq 90.0 * y_i \\ \bigwedge_{d:[1..10]} \begin{bmatrix} 5.0 * y_i \leq x'_{i,d} \\ x'_{i,d} \leq 75.0 * y_i \end{bmatrix} \\ w'_i \geq x'_{i,i} + 4.0 * y_i \end{bmatrix}$$

Reformulation time reduced from $O(n)$ to $O(1)$.

“Loop” Optimization (Easy)

$\forall i \in \{1, \ldots, 5\} \quad \forall j \in \{1, \ldots, 10\} \quad x_{i,j} = y_{i-1,j}$

```
for i = 1 to 5 do
  for j = 1 to 10 do
    x[i, j] = y[i-1, j]
  done
done
```

$\forall j \in \{1, \ldots, 10\} \quad \forall i \in \{1, \ldots, 5\} \quad x_{i,j} = y_{i-1,j}$

```
for j = 1 to 10 do
  for i = 1 to 5 do
    x[i, j] = y[i-1, j]
  done
done
```

"Loop" Optimization (Easy)

$\forall i \in \{1, \ldots, 5\} \quad \forall j \in \{1, \ldots, 10\} \quad x_{i,j} = y_{i-1,j}$

```
for i = 1 to 5 do
  for j = 1 to 10 do
    x[i, j] = y[i-1, j]
  done
done
```

$\forall j \in \{1, \ldots, 10\} \quad \forall i \in \{1, \ldots, 5\} \quad x_{i,j} = y_{i-1,j}$

```
for j = 1 to 10 do
  for i = 1 to 5 do
    x[i, j] = y[i-1, j]
  done
done
```

Standard rules of first-order logic may apply.

“Loop” Optimization (Hard)

- What if there are dependent types?

  $\forall i \in \{1, \ldots, 5\} \quad \forall j \in \{1, \ldots, i\} \quad x_{i,j} = y_{i-1,j}$

```
for i = 1 to 5 do
  for j = 1 to i do
    x[i, j] = y[i-1, j]
  done
done
```

Conjunctive Normal Form with Dependent Types

- Indexed DNF expression

$$\bigvee_{i \in \sigma} \bigwedge_{i' \in \sigma'} e$$

- can be converted to indexed CNF

$$\bigwedge_{f \in (i:\sigma \rightarrow \sigma')} \bigvee_{i \in \sigma} \left\{ f\left(i\right) / i' \right\} e$$

- by introducing index over function space.
- Other solutions, *e.g.* introducing slack variables, also possible.

What Are Random Variables?

- Wasserman (2004) says:

  A random variable is a mapping

  $$X : \Omega \rightarrow \mathbb{R}$$

  that assigns a real number $X(\omega)$ to each outcome ω.

What Are Random Variables?

- Wasserman (2004) says:

  A random variable is a mapping
  $$X : \Omega \to \mathbb{R}$$
  that assigns a real number $X(\omega)$ to each outcome ω.

However:

- Treated as real: $\mathbb{P}(X \geq 5)$
- Not random:
  We write
  $$X \sim \texttt{Bernoulli}(p)$$
  to mean that X is *exactly* distributed as
  $$f(x) = p^x(1-p)^{1-x} \text{ for } x \in \{0, 1\}$$

What Are Random Variables?

Not variables:

- Cannot substitute occurrences of X for anything.
  e.g. In $\mathbb{P}(X \geq 5)$, cannot replace X with anything that preserves meaning of the statement.
- Dependence matters.
  e.g. Two random variables X and Y, both distributed as `Bernoulli`(0.5), each 0 or 1 with probability 0.5. What is $\mathbb{P}(X + Y = 2)$?

  Perhaps 0.25? But not if $Y = 1 - X$.

What Are Random Variables?

Not variables:

- Cannot substitute occurrences of X for anything.
  e.g. In $\mathbb{P}(X \geq 5)$, cannot replace X with anything that preserves meaning of the statement.
- Dependence matters.
  e.g. Two random variables X and Y, both distributed as `Bernoulli`(0.5), each 0 or 1 with probability 0.5. What is $\mathbb{P}(X + Y = 2)$?

  Perhaps 0.25? But not if $Y = 1 - X$.

Random variables are neither random nor variable.

Previous Work

- Giry (1981), Jones and Plotkin (1989)
  Probability distributions are a monad.
- Kozen (1981)
  Formalized semantics.
- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.
- Park, Pfenning, and Thrun (2004)
  Continuous distributions also, but support only sampling.
- Erwig and Kollmansberger (2006)
  Provide Haskell library, but discrete distributions only, computational efficiency not optimized.

Previous Work

- Giry (1981), Jones and Plotkin (1989)
  Probability distributions are a monad.
- Kozen (1981)
  Formalized semantics.
- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.
- Park, Pfenning, and Thrun (2004)
  Continuous distributions also, but support only sampling.
- Erwig and Kollmansberger (2006)
  Provide Haskell library, but discrete distributions only, computational efficiency not optimized.

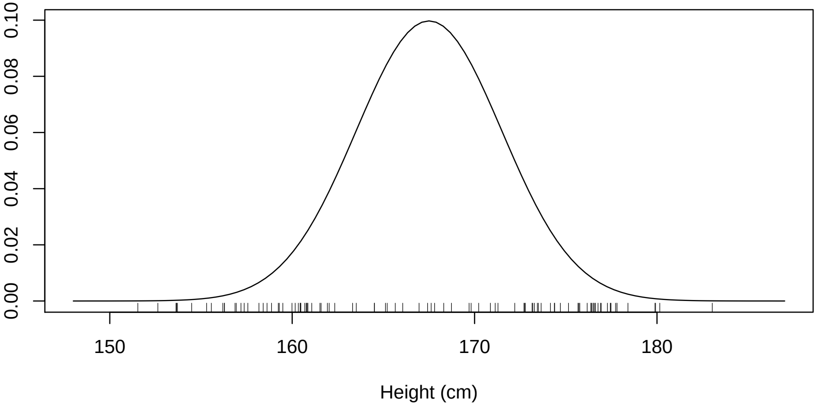
Our goal: Unify these results in a single system.

Syntax: Probability Language

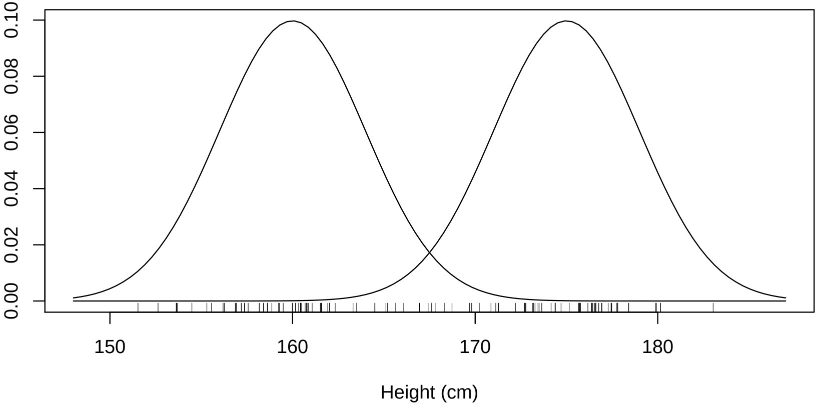
Bhat, Agarwal, Gray, Vuduc (2010)

$$
\begin{aligned}
T &::= \texttt{Bool} \mid \texttt{Int} \mid \texttt{Real} \mid T_1 \times T_2 \mid \texttt{Prob}\ T \\
E &::= x \mid \texttt{true} \mid \texttt{false} \\
&\quad \mid r \mid E_1 + E_2 \mid E_1 \times E_2 \\
&\quad \mid (E_1, E_2) \mid \texttt{fst}\ E \mid \texttt{snd}\ E \\
&\quad \mid \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \mid E_1 = E_2 \mid E_1 \leq E_2 \\
&\quad \mid \texttt{uniform} \mid \texttt{return}\ E \mid \texttt{let}\ x \sim E_1\ \texttt{in}\ E_2
\end{aligned}
$$

Gaussian Model



Mixture of Gaussians Model



Trying alternative statistical models

Formulation:

$$X_i \sim \text{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Trying alternative statistical models

Formulation:

$$X_i \sim \text{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Trying alternative statistical models

Formulation:

$$X_i \sim \mathrm{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Formulation:

$$Z_i \sim \mathrm{Bernoulli}(0.5)$$
$$X_i \sim \mathrm{Normal}((1 - Z_i)\theta_0 + Z_i\theta_1, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Trying alternative statistical models

Formulation:

$$X_i \sim \mathrm{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Formulation:

$$Z_i \sim \mathrm{Bernoulli}(0.5)$$
$$X_i \sim \mathrm{Normal}((1 - Z_i)\theta_0 + Z_i\theta_1, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$(\hat{\theta}_0, \hat{\theta}_1)$:= rand();
while (...)
  for i = 1 to n do
    γ_i := $\phi(x_i; \hat{\theta}_1, 1) / (\phi(x_i; \hat{\theta}_0, 1) + \phi(x_i; \hat{\theta}_1, 1))$;
  $\hat{\theta}_0$:= $\sum_{i=1}^{n} (1-\gamma_i) * x_i \;/\; \sum_{i=1}^{n} (1-\gamma_i)$;
  $\hat{\theta}_1$:= $\sum_{i=1}^{n} \gamma_i * x_i \;/\; \sum_{i=1}^{n} \gamma_i$;
return $(\hat{\theta}_0, \hat{\theta}_1)$;

Interactive algorithm assistant

Features

- enter problems
- apply schemas
- undo/redo
- combinators

Status

- can solve several textbook examples of MLE, incl. via EM
- autotuning + more sophisticated code generation is planned

```
File  Edit  View  Terminal  Help
sooraj@lucy:~/mathProg/om$ om
        Objective Caml version 3.11.1

# load gaussian;;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  pdf
  (let pick = normal mu ss in
   var x1 ~ pick in var x2 ~ pick in var x3 ~ pick in return (x1, x2, x3))
  (9, 28, 11)
  | 0 <= ss}
# ap ( let_simpl <&> pdf_simpl );;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  ss^-1.500000 * %e^((9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000) * (2 * %pi)^-1.500000
  | 0 <= ss}
# ap ( argmax_log <&> log_simpl <&> argmax_add );;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  -1.500000 * log ss + (9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000
  | 0 <= ss}
# ap descartes;;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  -1.500000 * log ss + (9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000
  | 0 <= ss && 0 = -1.500000/ss + (9 - mu)^2 * ss^-2 * 0.500000 + (28 - mu)^
    2 * ss^-2 * 0.500000 + (11 - mu)^2 * ss^-2 * 0.500000 && 0 = 1/ss * (9 -
    mu) + 1/ss * (28 - mu) + 1/ss * (11 - mu)}
# ap ( rewrite undistr <&> rewrite factors_0 <&> simpl <&> back_solve None );;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  -1.500000 * log ss + (9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000
  | mu = 16.000000 && ss = 72.666667}
#
```

Conclusions

- Richly typed language covering:
  - linear algebra
  - indexing
  - Boolean logic
  - optimization
  - probability and statistics
- Library of transformations:
  - bigM and convex-hull methods for disjunctive constraints
  - Boolean propositions to pure integer constraints
  - several specific to probablity distributions
  - simple computer algebra: *e.g.* $0x \mapsto 0$
  - need many more

Conclusions

- Richly typed language covering:
  - linear algebra
  - indexing
  - Boolean logic
  - optimization
  - probability and statistics
- Library of transformations:
  - bigM and convex-hull methods for disjunctive constraints
  - Boolean propositions to pure integer constraints
  - several specific to probablity distributions
  - simple computer algebra: *e.g.* $0x \mapsto 0$
  - need many more
- Next step: autotuning!

Acknowledgments

- Optimization:

  Ignacio Grossmann (Carnegie Mellon)
  Nick Sawaya and Vikas Goel (Exxon Mobil)

- Indexing:

  Bob Harper (Carnegie Mellon)

- Statistics:

  Sooraj Bhat and Alex Gray (GeorgiaTech)

- Linear algebra, HPC, Autotuning:

  Rich Vuduc (GeorgiaTech)