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)

\begin{align*}
x_1 & \geq 1.0 \\
x_2 & \geq 1.0 \\
x_1 + x_2 & \leq 5.0
\end{align*}

\begin{itemize}
  \item $x_1 \geq 1.0$
  \item $x_2 \geq 1.0$
  \item $x_1 + x_2 \leq 5.0$
\end{itemize}
max \ x_1 - x_2 \\
\ x_1 \geq 1.0 \\
\ x_2 \geq 1.0 \\
\ x_1 + x_2 \leq 5.0
Linear programs (LP)

Cannot represent multiple polyhedra.

\[
\text{max } x_1 - x_2 \\
x_1 \geq 1.0 \\
x_2 \geq 1.0 \\
x_1 + x_2 \leq 5.0
\]
Declaring discrete choice – with disjunction

\[
\begin{bmatrix}
x_1 \geq 1 \\
x_2 \geq 1 \\
x_1 + x_2 \leq 5
\end{bmatrix}
\]

\[
\begin{bmatrix}
5 \leq x_1 \leq 8 \\
4 \leq x_2 \leq 7
\end{bmatrix}
\]
Declaring discrete choice – with disjunction

Language of DP extends LP with disjunction
Declaring discrete choice – with disjunction

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 $x \leq 2.0$
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) \( \rightarrow \) 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) \(\lor\) (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

\[
\begin{align*}
\text{var } & x: [10.0, 100.0] \\
\text{var } & w: [2.0, 50.0] \\
\text{min } & x + w \text{ subject_to} \\
(x & \leq w) \lor (x \geq w + 4.0)
\end{align*}
\]

Output

\[
\begin{align*}
\text{var } & x: [10.0, 100.0] \\
\text{var } & w: [2.0, 50.0] \\
\text{min } & x + w \text{ subject_to} \\
\exists y_1: [0, 1], & \exists y_2: [0, 1], \\
\exists x_1: [0, 100.0], & \exists x_2: [0, 100.0], \\
\exists w_1: [0, 50.0], & \exists w_2: [0, 50.0], \\
& w = w_1 + w_2, \\
& x = x_1 + x_2, \\
& y_1 + y_2 = 1, \\
10.0 \times y_1 & \leq x_1, \\
x_1 & \leq 100.0 \times y_1, \\
2.0 \times y_1 & \leq w_1, \\
w_1 & \leq 50.0 \times y_1, \\
x_1 & \leq w_1, \\
10.0 \times y_2 & \leq x_2, \\
x_2 & \leq 100.0 \times y_2, \\
2.0 \times y_2 & \leq w_2, \\
w_2 & \leq 50.0 \times y_2, \\
x_2 & \geq w_2 + 4.0 \times y_2
\end{align*}
\]

- Output generated in MPS and AMPL formats
- Implemented as a DSL embedded in OCaml
Switched flow process, comparison to ILOG Concert

<table>
<thead>
<tr>
<th>Method</th>
<th>#vars (#binary)</th>
<th>#constr. (#IC)</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow-Concert</td>
<td>1061 (874)</td>
<td>1080 (718)</td>
<td>36.85</td>
</tr>
<tr>
<td>flow-IC</td>
<td>477 (291)</td>
<td>1001 (438)</td>
<td>11.60</td>
</tr>
<tr>
<td>flow-BM</td>
<td>477 (291)</td>
<td>1198</td>
<td>3.37</td>
</tr>
<tr>
<td>flow-CH</td>
<td>1194 (631)</td>
<td>2747</td>
<td>1.09</td>
</tr>
</tbody>
</table>

- All 3 of our methods improve on state-of-the-art.
Strip packing, comparison to expert

<table>
<thead>
<tr>
<th>Method</th>
<th>#vars (#binary)</th>
<th>#constr. (#IC)</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack12-IC</td>
<td>289 (264)</td>
<td>342 (264)</td>
<td>1.83</td>
</tr>
<tr>
<td>pack12-BM</td>
<td>289 (264)</td>
<td>342</td>
<td>1.22</td>
</tr>
<tr>
<td>pack12-CH</td>
<td>1345 (264)</td>
<td>2718</td>
<td>168.38</td>
</tr>
<tr>
<td>pack12-BM-expert</td>
<td>289 (264)</td>
<td>342</td>
<td>1.82</td>
</tr>
<tr>
<td>pack12-CH-expert</td>
<td>1345 (264)</td>
<td>1662</td>
<td>149.57</td>
</tr>
<tr>
<td>pack21-IC</td>
<td>883 (840)</td>
<td>1071 (840)</td>
<td>24.44</td>
</tr>
<tr>
<td>pack21-BM</td>
<td>883 (840)</td>
<td>1071</td>
<td>55.01</td>
</tr>
<tr>
<td>pack21-CH</td>
<td>4243 (840)</td>
<td>8631</td>
<td>991.68</td>
</tr>
<tr>
<td>pack21-BM-expert</td>
<td>883 (840)</td>
<td>1071</td>
<td>29.56</td>
</tr>
<tr>
<td>pack21-CH-expert</td>
<td>4243 (840)</td>
<td>5271</td>
<td>&gt; 3600.00</td>
</tr>
</tbody>
</table>

Our mechanizations perform just as well as expert encodings.
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.
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:

$$\forall i \in \{1, \ldots, n\} \quad x_{i+1} = x_i + y_i$$
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 $A : R \times S \rightarrow \mathbb{R}$
Indexing Is A Generalization of Matrix Notation

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

$$\otimes : (R \times S \to \mathbb{R}) \times (S \to \mathbb{R}) \to (R \to \mathbb{R})$$
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)\}
Can now define non-rectangular data:

\[
A : i : \text{JOBS} \times \text{STAGES}[i] \rightarrow \text{real}
\]

\[
A = \begin{bmatrix}
A_{a,s1} & A_{a,s2} \\
A_{b,s1} & A_{b,s3} & A_{b,s4} \\
A_{c,s3} & A_{c,s4}
\end{bmatrix}
\]
Beyond Matrices

- Can now define non-rectangular data:
  \[ A : i:\text{JOBS} \times \text{STAGES}[i] \rightarrow \text{real} \]
  \[ A = \begin{bmatrix}
  A_{a,s1} & A_{a,s2} \\
  A_{b,s1} & A_{b,s2} \\
  A_{c,s3} & A_{c,s4}
\end{bmatrix} \]

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

\[ A : i:JOBS * STAGES[i] \rightarrow real \]

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

- 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:

\[
\begin{align*}
  x_2 &= x_1 + y_1 \\
  x_3 &= x_2 + y_2 \\
  x_4 &= x_3 + y_3 \\
  x_5 &= x_4 + y_4 \\
  \vdots &= \vdots \\
\end{align*}
\]
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:

\[
\begin{align*}
    x_2 &= x_1 + y_1 \\
    x_3 &= x_2 + y_2 \\
    x_4 &= x_3 + y_3 \\
    x_5 &= x_4 + y_4 \\
    \vdots &\quad \vdots &\quad \vdots
\end{align*}
\]

- 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:
  \[
  \bigwedge_{i:[1..10]} \left[ \begin{array}{l}
  10.0 \cdot y_i \leq w_i' \\
  w_i' \leq 90.0 \cdot y_i \\
  \end{array} \right] \quad \bigwedge_{d:[1..10]} \left[ \begin{array}{l}
  5.0 \cdot y_i \leq x_i',d \\
  x_i',d \leq 75.0 \cdot y_i \\
  \end{array} \right] \quad w_i' \geq x_i',i + 4.0 \cdot y_i
  \]
Computational Improvements

- Input to our software:

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

- Our software’s output:

\[ \bigwedge_{i: [1..10]} \left[ \begin{array}{l}
10.0 \times y_i \leq w_i' \\
w_i' \leq 90.0 \times y_i \\
w_i' \leq 90.0 \times y_i \\
5.0 \times y_i \leq x'_{i,d} \\
x'_{i,d} \leq 75.0 \times y_i \\
w_i' \geq x'_{i,j} + 4.0 \times y_i
\end{array} \right] \]

Reformulation time reduced from \(O(n)\) to \(O(1)\).
∀\(i\in\{1,\ldots,5\}\) ∀\(j\in\{1,\ldots,10\}\) \(x_{i,j} = y_{i-1,j}\)

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

∀\(j\in\{1,\ldots,10\}\) ∀\(i\in\{1,\ldots,5\}\) \(x_{i,j} = y_{i-1,j}\)

for \(j = 1\) to \(10\) do
  for \(i = 1\) to \(5\) do
    \(x[i, j] = y[i-1, j]\)
  done
done
∀ \(i \in \{1, \ldots, 5\}\) \(\forall j \in \{1, \ldots, 10\}\) \(x_{i,j} = y_{i-1,j}\)

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

∀ \(j \in \{1, \ldots, 10\}\) \(\forall i \in \{1, \ldots, 5\}\) \(x_{i,j} = y_{i-1,j}\)

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.
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
Indexed DNF expression

\[ \bigvee_{i \in \sigma} \bigwedge_{i' \in \sigma'} e \]

can be converted to indexed CNF

\[ \bigwedge_{f \in (i:\sigma \rightarrow \sigma')} \bigvee_{i \in \sigma} \{ f(i) / i' \} e \]

by introducing index over function space.

Other solutions, e.g. introducing slack variables, also possible.
A random variable is a mapping

\[ X : \Omega \rightarrow \mathbb{R} \]

that assigns a real number \( X(\omega) \) to each outcome \( \omega \).
What Are Random Variables?

- Wasserman (2004) says:
  
  \[
  \text{A random variable is a mapping } \quad X : \Omega \rightarrow \mathbb{R} \\
  \text{that assigns a real number } X(\omega) \text{ to each outcome } \omega.
  \]

  However:

- Treated as real: \( \mathbb{P}(X \geq 5) \)

- Not random:
  
  We write

  \[
  X \sim \text{Bernoulli}(p)
  \]

  to mean that \( X \) is exactly distributed as

  \[
  f(x) = p^x(1 - p)^{1-x} \quad \text{for } x \in \{0, 1\}
  \]
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 $\text{Bernoulli}(0.5)$, 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

  Probability distributions are a monad.

- Kozen (1981)
  Formalized semantics.

- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.

  Continuous distributions also, but support only sampling.

- Erwig and Kollmansberger (2006)
  Provide Haskell library, but discrete distributions only, computational efficiency not optimized.
Previous Work

  Probability distributions are a monad.
- Kozen (1981)
  Formalized semantics.
- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.
  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)

\[ T ::= \text{Bool} \mid \text{Int} \mid \text{Real} \mid T_1 \times T_2 \mid \text{Prob } T \]

\[ E ::= x \mid \text{true} \mid \text{false} \]

\[ \mid r \mid E_1 + E_2 \mid E_1 \times E_2 \]

\[ \mid (E_1, E_2) \mid \text{fst } E \mid \text{snd } E \]

\[ \mid \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \mid E_1 = E_2 \mid E_1 \leq E_2 \]

\[ \mid \text{uniform} \mid \text{return } E \mid \text{let } x \sim E_1 \text{ in } E_2 \]
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:**

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

**Solution:**

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

**Formulation:**

\[ Z_i \sim \text{Bernoulli}(0.5) \]
\[ X_i \sim \text{Normal}((1 - Z_i)\theta_0 + Z_i\theta_1, 1) \]
\[
\hat{\theta} = \arg \max_{\theta} f(x \mid \theta)
\]
Trying alternative statistical models

Formulation:

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

Solution:

\[ \hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} X_i \]

Formulation:

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

Solution:

\( (\hat{\theta}_0, \hat{\theta}_1) := \text{rand}(); \)
\( \text{while} \ (\ldots) \)
\( \quad \text{for} \ i = 1 \ \text{to} \ n \ \text{do} \)
\( \quad \quad \gamma_i := \phi(x_i; \hat{\theta}_1, 1) / (\phi(x_i; \hat{\theta}_0, 1) + \phi(x_i; \hat{\theta}_1, 1)) ; \)
\( \quad \hat{\theta}_0 := \sum_{i=1}^{n} (1 - \gamma_i) * x_i / \sum_{i=1}^{n} (1 - \gamma_i) ; \)
\( \quad \hat{\theta}_1 := \sum_{i=1}^{n} \gamma_i * x_i / \sum_{i=1}^{n} \gamma_i ; \)
\( \text{return} \ (\hat{\theta}_0, \hat{\theta}_1) ; \)
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
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 probability distributions
  - simple computer algebra: e.g. $0 \cdot x \mapsto 0$
  - 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 probability 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 (Georgia Tech)*

- Linear algebra, HPC, Autotuning:
  - *Rich Vuduc (Georgia Tech)*