CvxLean
a convex optimization modeling framework based on the Lean 4 proof assistant

Ramon Fernández Mir **Paul Jackson**
Alex Bentkamp Jeremy Avigad

ICCOPT
July 21st, 2025

Proof assistants

A *proof assistant* provides an environment for

- Formally expressing mathematical definitions and theorems
- Describing proofs.
- Having those proofs formally checked
- Automating expression manipulation & proof generation

State of the art:

- Large libraries (e.g. UG math and more)
- Applications to
  - math teaching & research
  - formal verification of hardware and software
  - programming language foundations
- Industrial take-up (e.g. Intel, AMD, Apple, AWS, Meta, IOG)

Examples: *Lean*, Coq, Isabelle/HOL, ACL2, Agda

DCP example in CvxLean

```
def p :=
  optimization (x y : ℝ)
    minimize -sqrt (x - y)
    subject to
      c₁ : y = 2*x - 3
      c₂ : x^2 ≤ 2
      c₃ : 0 ≤ x - y

equivalence eqv/q : p := by dcp
```

q now bound to

```
optimization (x y t.0 t.1 : ℝ)
  minimize -t.0
  subject to
    c₁' : zeroCone (2*x - 3 - y)
    c₂' : nonnegOrthCone (2 - t.1)
    c₃' : rotatedSoCone t.1 0.5 ![x]
    c₄' : rotatedSoCone (x - y) 0.5 ![t.0]
```

DCP example in CvxLean (continued)

CvxLean interfaces to the MOSEK conic solver

```
solve p
```

```
#print p.conicForm -- shows the problem in conic form
#eval p.status     -- "PRIMAL_AND_DUAL_FEASIBLE"
#eval p.value      -- -2.101003
#eval p.solution   -- (-1.414214, -5.828427)
```

Atom library

Currently, CvxLean defines 107 atoms, including:

- 22 classes of affine atoms, including elementary operations ($+$, $-$, $\cdot$, and $/$) and various operations to manipulate vectors and matrices.
- 11 classes of convex atoms: absolute value, exponential, Huber loss, positive inverse, Kullback-Leibler divergence, log-sum-exp, max, ℓ^2-norm, some powers (2, -2, and -1), quadratic-over-linear, and $x \cdot \exp(x)$.
- 6 classes of concave atoms: entropy, geometric mean, logarithm, log-det, min, and square root.

Atom declaration for square-root

- curv := ConcaveFn
- domain := $\mathbb{R}$, and args := $(x : \mathbb{R})$ with inputKind(1) = Increasing.
- vconds(x) := $0 \leq x$
- expr(x) := $\sqrt{x}$
- impDomain(x) := $\mathbb{R}$, and impVars := $(v : \mathbb{R})$
- impObj(x, v) := v
- impConstrs := $\left[\lambda(x, v).\ (x, 0.5, v) \in \mathcal{Q}_r^3\right]$

CvxLean generates statements of desired properties

- feasibility : $\forall x : \mathbb{R}.\ 0 \leq x \Rightarrow \left(x, 0.5, \sqrt{x}\right) \in \mathcal{Q}_r^3$
- monotonicity : $\forall x, y : \mathbb{R}.\ 0 \leq x \Rightarrow 0 \leq y \Rightarrow x \geq y \Rightarrow \sqrt{x} \geq \sqrt{y}$
- bounds : $\forall x, v : \mathbb{R}.\ (x, 0.5, v) \in \mathcal{Q}_r^3 \Rightarrow v \leq \sqrt{x}$
- vcondElim : $\forall x, y, v : \mathbb{R}.\ (x, 0.5, v) \in \mathcal{Q}_r^3 \Rightarrow y \geq x \Rightarrow 0 \leq y$

which all must be proven.

Optimization problem equivalences

Let P be a minimization problem defined over domain D
Let Q be defined over E.

P and Q are *equivalent* if there exist maps
$\varphi : D \to E$ and $\psi : E \to D$ such that:

(φ_{opt}) $\forall x.\ \text{optimal}_P(x) \Rightarrow \text{optimal}_Q(\varphi(x))$
(ψ_{opt}) $\forall y.\ \text{optimal}_Q(y) \Rightarrow \text{optimal}_P(\psi(y))$

Can also add

(φ_{feas}) $\forall x.\ \text{feasible}_P(x) \Rightarrow \text{feasible}_Q(\varphi(x))$
(ψ_{feas}) $\forall y.\ \text{feasible}_Q(y) \Rightarrow \text{feasible}_P(\psi(y))$

Putting optimization problems into Dcp form

Sometimes initial user problems need transformations to put them into DCP form.

$$\begin{array}{ll} \text{Minimize} & x \\ \text{subject to} & 0.001 \leq x \\ & \frac{1}{\sqrt{x}} \leq \exp(x) \end{array} \quad \equiv \quad \begin{array}{ll} \text{Minimize} & x \\ \text{subject to} & 0.001 \leq x \\ & \exp(-x) \leq \sqrt{x} \end{array}$$

Not in DCP form | **In DCP form**

A possible sequence of rewrites:

$$\frac{1}{\sqrt{x}} \leq \exp(x) \quad \rightsquigarrow_1 \quad 1 \leq \exp(x)\sqrt{x} \quad \rightsquigarrow_2 \quad 1 \leq \sqrt{x}\exp(x)$$

$$\rightsquigarrow_3 \quad \frac{1}{\exp(x)} \leq \sqrt{x} \quad \rightsquigarrow_4 \quad \exp(-x) \leq \sqrt{x}$$

Rewrite rules applied bidirectionally (steps 1 & 3).
No obvious cost metric being reduced.

Manual guidance of rewrites very tedious. Automation needed ...

A preDCP transformation tactic for Lean

```
def p : Minimization ℝ ℝ :=
  optimization (x : ℝ)
    minimize (x)
    subject to
      h1 : 1 / 1000 ≤ x
      h2 : 1 / (sqrt x) ≤ exp x

equivalence eqv/q : p := by pre_dcp

#print q
-- def q : Minimization ℝ ℝ :=
--   optimization (x : ℝ)
--     minimize x
--     subject to
--       h1 : 1 / 1000 ≤ x
--       h2 : exp (-x) ≤ sqrt x
```

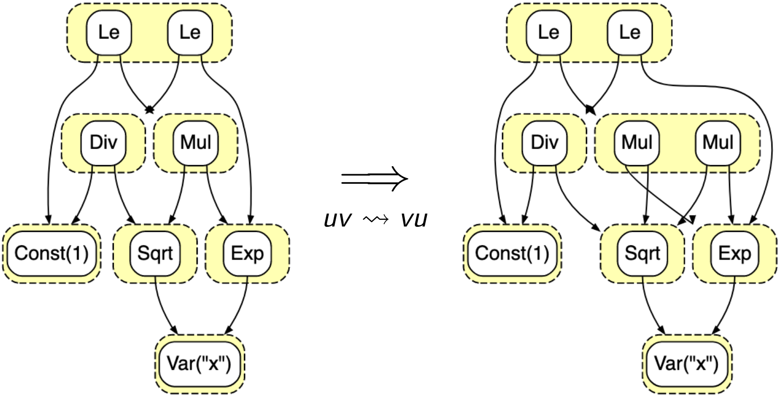
pre_dcp uses *e-graphs* and *e-graph rewriting* for an efficient breadth-first search of equivalent problems

E-graphs and e-graph rewriting

An *e-graph* represents a set of terms and a congruence relation on those terms. *E-graph rewriting* adds new terms equal to existing terms

$$\{1 \le \sqrt{x}\exp(x),\ 1/\sqrt{x} \le \exp(x)\}$$

$$\{1 \le \exp(x)\sqrt{x},\ 1 \le \sqrt{x}\exp(x),\ 1/\sqrt{x} \le \exp(x)\}$$



Example run of preDCP tactic

Started with geometric programming problem, after change of variables

- 4 variables, 8 constraints
- Problem size 97 to begin, 104 at end
- 37k rewrite rule matches
- 41k nodes in e-graph
- 123 rewrite steps to justify transformation
- 19 iterations of parallel rewriting
- 5s for e-graph computations,
  10s for verification in Lean

Related work

Formally verifying solver output

- ValidSDP (Coq)
- SDP-based non-linear-arithmetic prover (HOL Light)

Using interval arithmetic within solver: VSDP

Formally verifying convex optimization algorithms
(Peking University, using Lean)

Conclusions and questions

CvxLean implements & formally verifies

- automatic DCP form to conic form tranformations,
- manually-guided & automatic transformations into DCP form

Questions for DCP Community

- When do you care about problem transformation correctness, and how much?
- Are there concerns about correctness of convex solvers?
- What does correctness mean, when
  - using floating-point arithmetic?
  - problem parameters are approximate?
- How important is integration with e.g. CvxPy?
- What interaction expertise level(s) are appropriate?
- Is automatic transformation into DCP form useful?
- What further atoms or features are most desirable?
- Where can larger problem examples be found?

Further Information about CvxLean

- Code
  https://github.com/verified-optimization/CvxLean
- *Transforming optimization problems into Disciplined Convex Programming form.*
  R. Fernández Mir, P. Jackson, S. Bhat, A. Goens, T. Grosser.
  International Conference on Intelligent Computer Mathematics (CICM). 2024.
  https://link.springer.com/chapter/10.1007/978-3-031-66997-2_11
- *Verified reductions for optimization*
  A. Bentkamp, R. Fernández Mir, J. Avigad
  International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). 2023.
  https://link.springer.com/chapter/10.1007/978-3-031-30820-8_8
- *Verified transformations for convex programming*
  Ramon Fernández Mir. PhD Thesis. July 2024.
  https://era.ed.ac.uk/handle/1842/42057

Core Lean definitions

```
structure Minimization (D R : Type) where
  objFun : D → R
  constraints : D → Prop

variable {D R : Type} [Preorder R] (p : Minimization D R)

def feasible (x : D) : Prop := p.constraints x

def optimal (x : D) : Prop :=
  p.feasible x ∧
  ∀ y, p.feasible y → p.objFun x ≤ p.objFun y

structure Solution where
  point : D
  isOptimal : p.optimal point
```

preDcp tactic configuration

Atoms

unary: $-(\cdot)$, $(\cdot)^{-1}$, $|\cdot|$, $\sqrt{\cdot}$, log, exp, xexp, entr,
binary: $+$, $-$, $\times$, $/$, ^, min, max, qol, geo, lse, norm2

Rewrite rules (17 unidirectional, 51 bidirectional)

1. On $\mathbb{R}$-valued terms

$$\forall x \in \mathbb{R}.\ \frac{1}{\exp(x)} \leftrightsquigarrow \exp(-x)$$

2. On propositions

$$\forall a, b, c \in \mathbb{R}.\ c > 0 \Rightarrow \left(\frac{a}{c} \leq b \leftrightsquigarrow a \leq bc\right)$$

3. On whole problems

$$\forall f, cs.\ (\forall x.\ cs(x) \Rightarrow\ f(x) > 0) \Rightarrow (f, cs) \rightsquigarrow (\lambda x.\log(f(x)), cs)$$