Colocolo

52.—Pampas Cat.

Optimizations on Ocelot - a Relational Logic Solver in Rosette

Ocelot

- An embedding of relational logic in Rosette

- Used in MemSynth, a tool for reasoning about memory consistency.

- Ocelot works fast and well in its domain, but does not implement certain general optimizations, and lacks benchmarks.

- Skolemization
- Optimized CNF-SAT translation

SKOLEMIZATION

$$\forall x\, \exists y.\ P(x,y) \Leftrightarrow \exists f\, \forall x.\ P(x,f(x))$$

Skolemization in Relational Logic

- In the case of a bounded universe, we translate ∀x∃y . P (x, y) to

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} P(A_i, A_j)$$

- After skolemizing,

$$\bigwedge_{i=1}^{n} P(A_i, f(A_i))$$

Skolemization in Relational Logic

- Skolemize existentially quantified declarations.
- Let **S** be the set of $\forall$ or $\neg\exists$ quantified variables in scope.
- For each $\exists$ or $\neg\forall$ quantified declaration v : mult R, where mult is a multiplicity (e.g. one, some) and R is a relation of arity k,
  - Introduce a new relation R_v of with arity equal to |**S**| + k.
  - Introduce a new expression E_v which will replace any occurrence of v down the tree.
  - $E_v = a_n.(a_{n-1}.(\ldots(a_1.R_v)\ldots))$, where $a_i \in \mathbf{S}$
  - upper-bound(R_v) = upper-bound(a_1) → … → upper-bound(a_n) → upper-bound(R)
  - Domain constraint := $R_v.U\underbrace{.\ \ldots\ .}_{\text{k times}}U \subseteq \{\, a_1 : mult_1\ R_1 , \ldots , a_n : mult_n\ R_n \mid \top \,\}$
  - Range constraint: (E_v in R) and (m E_v)

Translation to CNF

- $F_1 \land F_2 \land \ldots \land F_n \rightsquigarrow (F_1 \lor \neg o) \land \ldots \land (F_n \lor \neg o) \land (\neg F_1 \lor \ldots \lor \neg F_n \lor o)$.
- $F_1 \lor F_2 \lor \ldots \lor F_n \rightsquigarrow (\neg F_1 \lor o) \land \ldots \land (\neg F_n \lor o) \land (F_1 \lor \ldots \lor F_n \lor \neg o)$.
- o is a fresh auxiliary variable; true iff left hand formula is true.
- Apply recursively on each F_i and substitute new CNF-SAT variable
- Used internally by Kodkod

Meow: KodKod AST to Colocolo

- Written in Java

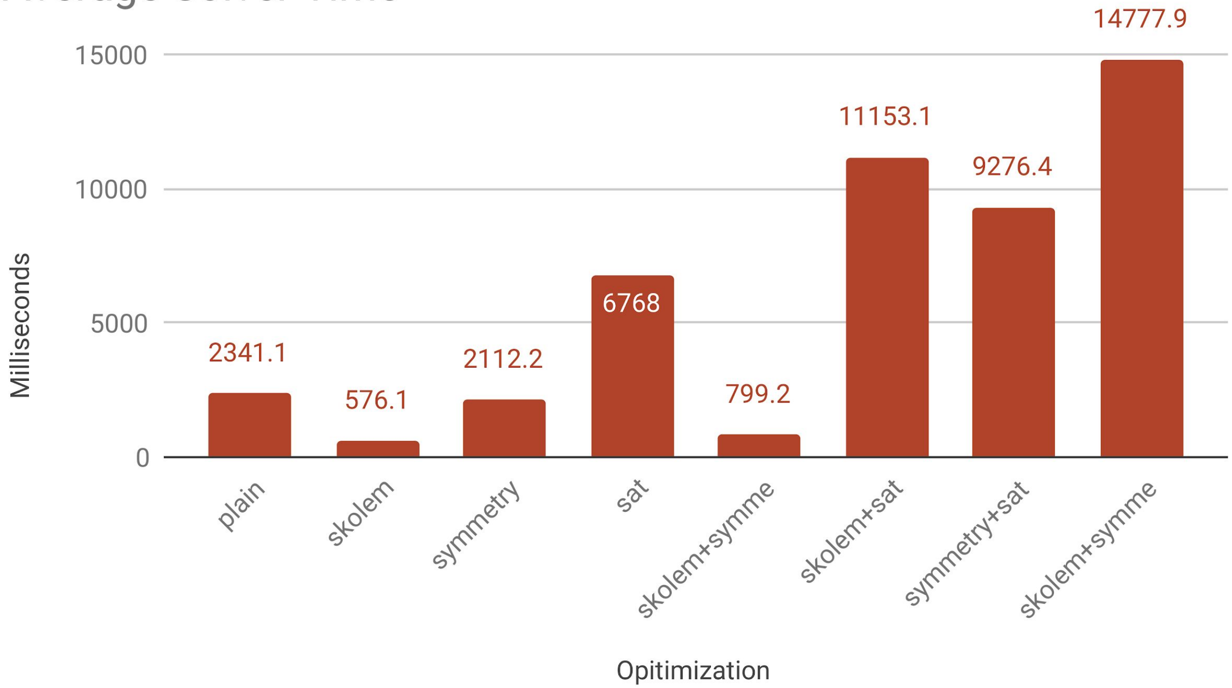
- Traverses Kodkod AST, which is actually a DAG due to node-sharing
    - DAG structure is preserved by caching nodes
    - Sort the nodes topologically to ensure correct compiled declaration order.

- Used to compile Kodkod benchmarks into Colocolo code

Demo/Reflection

- An engineering problem
  - Focused on performance improvement
  - Ocelot focused specifically on MemSynth
    - Difficulty with higher arity relations, especially when quantifying
    - No Integers
    - No Compact Boolean Circuits
- Performance Improvements
  - Reduction to SAT - not as good as we hoped
  - Skolemization: 2 - 4x improvement
  - Symmetry Breaking
    - Mixed bag

Average Solver Time


Milliseconds
plain
skolem
symmetry
sat
skolem+symme
skolem+sat
symmetry+sat
skolem+symme
Opitimization