


# Colocolo



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

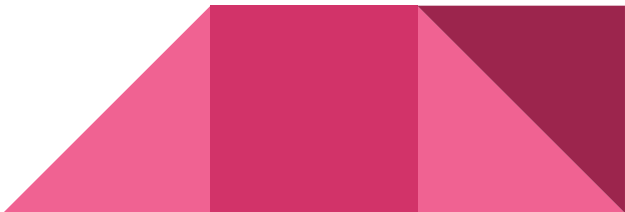
$$\forall x \exists y. P(x,y) \Leftrightarrow \exists f \forall x. P(x,f(x))$$

# Skolemization in Relational Logic

- In the case of a bounded universe, we translate  $\forall x \exists y . P(x, y)$  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  $\mathbf{S}$  be the set of  $\forall$  or  $\neg\exists$  quantified variables in scope.
- For each  $\exists$  or  $\neg\forall$  quantified declaration  $v : \text{mult } R$ , 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  $|\mathbf{S}| + k$ .
  - Introduce a new expression  $E_v$  which will replace any occurrence of  $v$  down the tree.
  - $E_v = a_n.(a_{n-1}.\dots(a_1.R_v)\dots)$ , where  $a_i \in \mathbf{S}$
  - $\text{upper-bound}(R_v) = \text{upper-bound}(a_1) \rightarrow \dots \rightarrow \text{upper-bound}(a_n) \rightarrow \text{upper-bound}(R)$
  - Domain constraint  $:= R_v.U \dots U \subseteq \{ a_1 : \text{mult}_1 R_1, \dots, a_n : \text{mult}_n R_n \mid \top \}$   
 $\underbrace{\hspace{1.5cm}}_{k \text{ times}}$
  - Range constraint:  $(E_v \text{ in } R)$  and  $(m E_v)$

# Translation to CNF

- $F_1 \wedge F_2 \wedge \dots \wedge F_n \rightsquigarrow (F_1 \vee \neg o) \wedge \dots \wedge (F_n \vee \neg o) \wedge (\neg F_1 \vee \dots \vee \neg F_n \vee o).$
- $F_1 \vee F_2 \vee \dots \vee F_n \rightsquigarrow (\neg F_1 \vee o) \wedge \dots \wedge (\neg F_n \vee o) \wedge (F_1 \vee \dots \vee F_n \vee \neg o).$
- $o$  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
    - Lack of Ocelot-expressible benchmarks with Quantifiers
  - Symmetry Breaking
    - Mixed b



## Average Solver Time

