Synthesis of Domain Specific Encoders for Bit-Vector Solvers

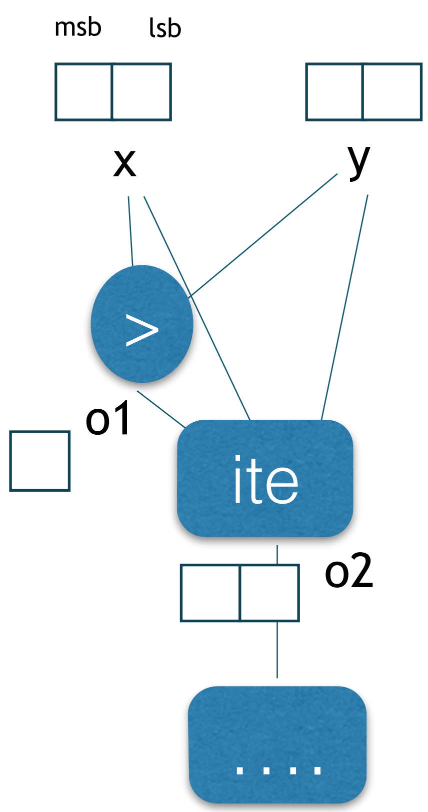
Jeevana Priya Inala
with
Rohit Singh, Armando Solar-Lezama

Appears in SAT'16

MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

High-level constraint to CNF clauses

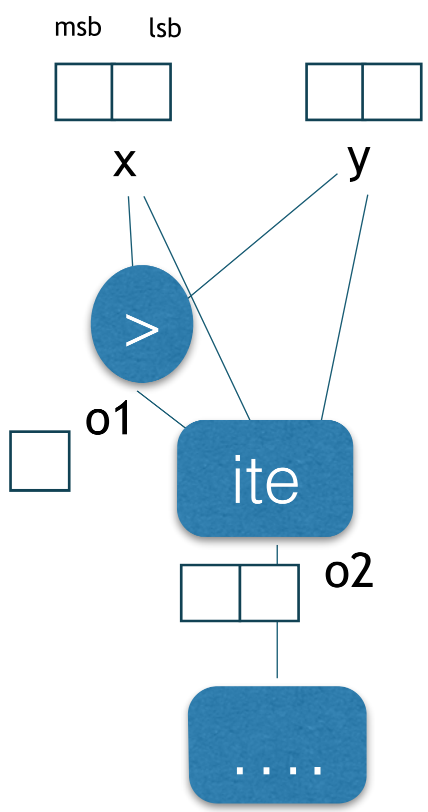
SMT solver
High-level constraint



SAT solver
CNF clauses

High-level constraint to CNF clauses

SMT solver
High-level constraint



Is this the "best" encoding?

SAT solver
CNF clauses

$\overline{y_1} \vee x_1 \vee \overline{t_1}$
$y_1 \vee \overline{x_1} \vee \overline{t_1}$
$\overline{y_1} \vee \overline{x_1} \vee t_1$
$y_1 \vee x_1 \vee t_1$
$\overline{t_2} \vee t_1$
$\overline{t_2} \vee \overline{y_0}$
$\overline{t_2} \vee x_0$

$\overline{t_1} \vee y_0 \vee \overline{x_0} \vee t_2$
$\overline{t_3} \vee \overline{y_1}$
$\overline{t_3} \vee x_1$
$y_1 \vee \overline{x_1} \vee t_3$
$o_1 \vee \overline{t_2}$
$o_1 \vee \overline{t_3}$
$t_2 \vee t_3 \vee \overline{o_1}$

$\overline{o_1} \vee x_1 \vee \overline{o2_1}$
$\overline{o_1} \vee \overline{x_1} \vee o2_1$
$o_1 \vee y_1 \vee \overline{o2_1}$
$o_1 \vee \overline{y_1} \vee o2_1$
$x_1 \vee y_1 \vee \overline{o2_1}$
$\overline{x_1} \vee \overline{y_1} \vee o2_1$

$\overline{o_1} \vee x_0 \vee \overline{o2_0}$
$\overline{o_1} \vee \overline{x_0} \vee o2_0$
$o_1 \vee y_0 \vee \overline{o2_0}$
$o_1 \vee \overline{y_0} \vee o2_0$
$x_0 \vee y_0 \vee \overline{o2_0}$
$\overline{x_0} \vee \overline{y_0} \vee o2_0$

....

Goal: Synthesize better code for this translation

How SAT solvers work?

- SAT solvers use unit propagation to infer variable assignments

How SAT solvers work?

- SAT solvers use unit propagation to infer variable assignments

Current variables assignment

$$x_1 = F, x_2 = T, x_3 = F, x_4 = F$$

How SAT solvers work?

- SAT solvers use unit propagation to infer variable assignments

Current variables assignment

$$x_1 = F, x_2 = T, x_3 = F, x_4 = F$$

$$\overbrace{x_1 \vee \overline{x_2} \vee x_4}^{\textit{false}} \vee x_5$$

Unit clause

How SAT solvers work?

- SAT solvers use unit propagation to infer variable assignments

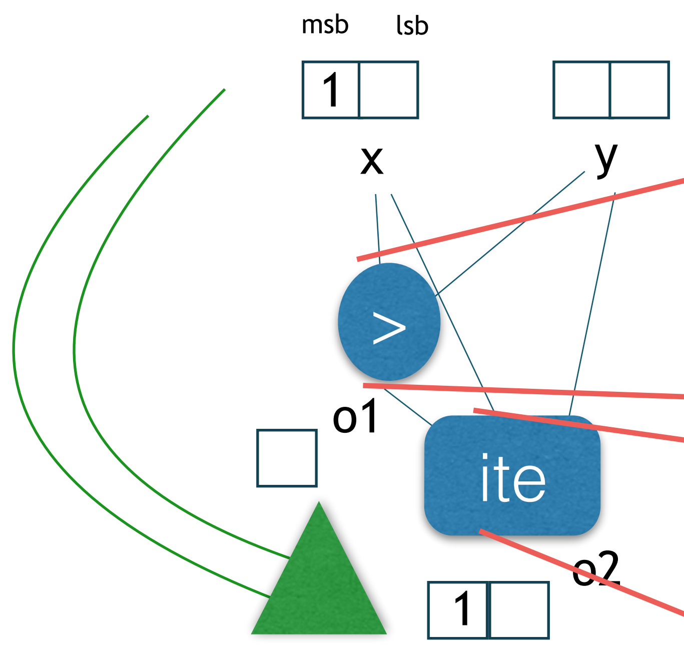
Current variables assignment

$$x_1 = F, x_2 = T, x_3 = F, x_4 = F, x_5 = T$$

$$x_1 \lor \overline{x_2} \lor x_4 \lor x_5$$

Criteria for a good encoding

- Maximal propagation
  - Maximize what we learn through unit propagations
- Fewer clauses
- Fewer variables



$$\overline{y_1} \vee x_1 \vee \overline{t_1}$$
$$y_1 \vee \overline{x_1} \vee \overline{t_1}$$
$$\overline{y_1} \vee \overline{x_1} \vee t_1$$
$$y_1 \vee x_1 \vee t_1$$
$$\overline{t_2} \vee t_1$$
$$\overline{t_2} \vee \overline{y_0}$$
$$\overline{t_2} \vee x_0$$

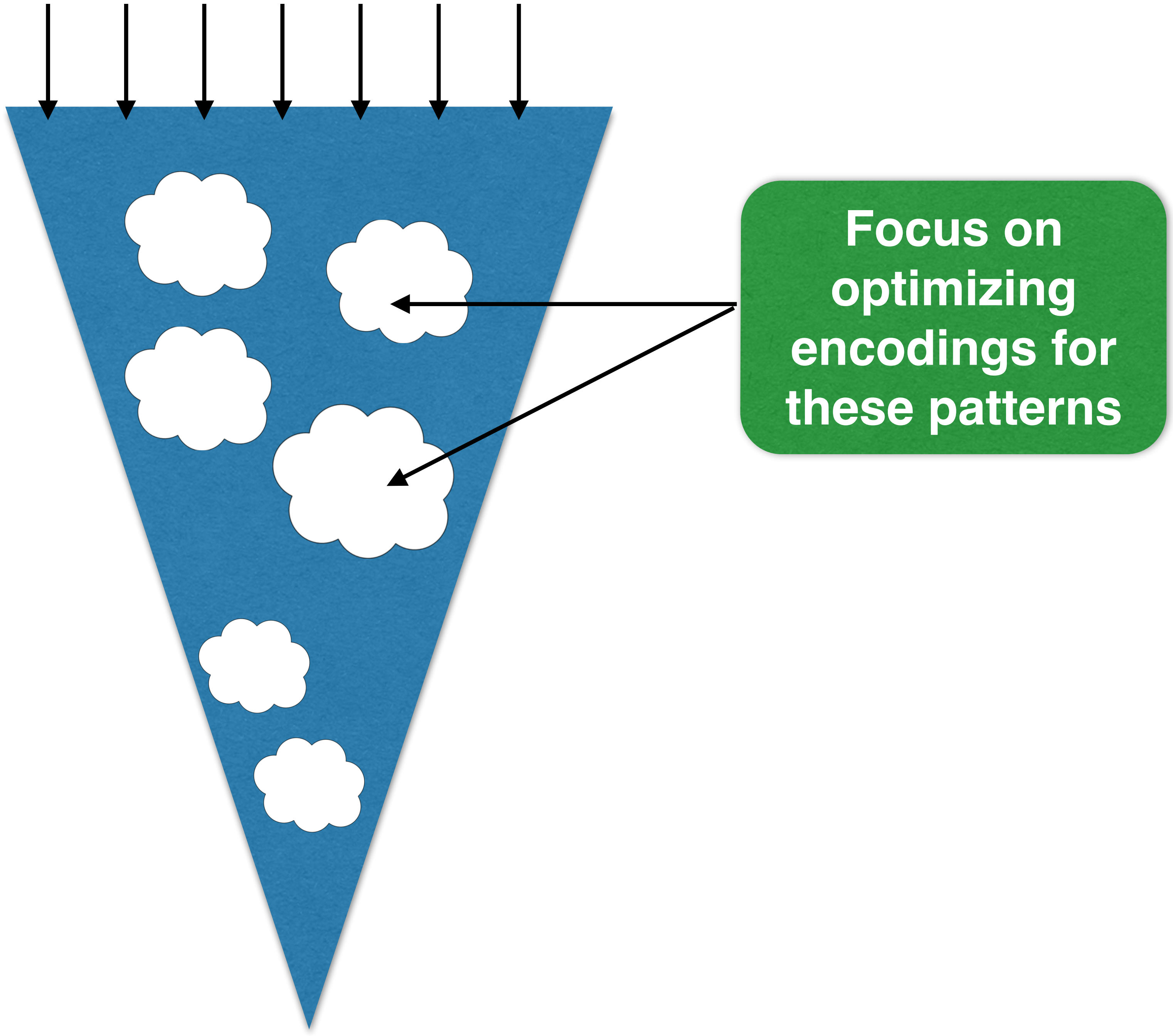
$$\overline{t_1} \vee y_0 \vee \overline{x_0} \vee t_2$$
$$\overline{t_3} \vee \overline{y_1}$$
$$\overline{t_3} \vee x_1$$
$$y_1 \vee \overline{x_1} \vee t_3$$
$$o_1 \vee \overline{t_2}$$
$$o_1 \vee \overline{t_3}$$
$$t_2 \vee t_3 \vee \overline{o_1}$$

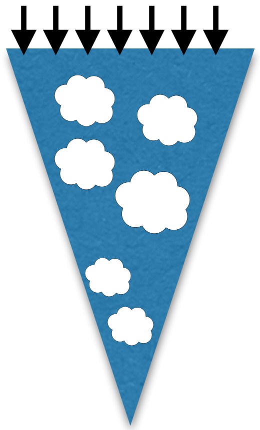
$$\overline{o_1} \vee x_1 \vee \overline{o2_1}$$
$$\overline{o_1} \vee \overline{x_1} \vee o2_1$$
$$o_1 \vee y_1 \vee \overline{o2_1}$$
$$o_1 \vee \overline{y_1} \vee o2_1$$
$$x_1 \vee y_1 \vee \overline{o2_1}$$
$$\overline{x_1} \vee \overline{y_1} \vee o2_1$$

$$\overline{o_1} \vee x_0 \vee \overline{o2_0}$$
$$\overline{o_1} \vee \overline{x_0} \vee o2_0$$
$$o_1 \vee y_0 \vee \overline{o2_0}$$
$$o_1 \vee \overline{y_0} \vee o2_0$$
$$x_0 \vee y_0 \vee \overline{o2_0}$$
$$\overline{x_0} \vee \overline{y_0} \vee o2_0$$

$$x_1 = 1 \xrightarrow{\cancel{Unit\ prop}} o2_1 = 1$$

Composing encodings does not preserve optimality

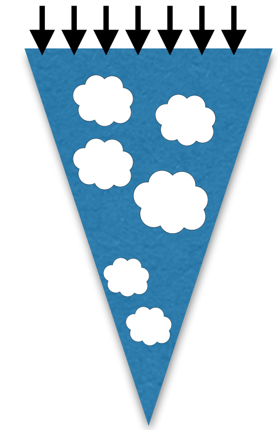

Focus on optimizing encodings for these patterns



What patterns to target?

How do we come up with “optimal” encoding for a pattern?

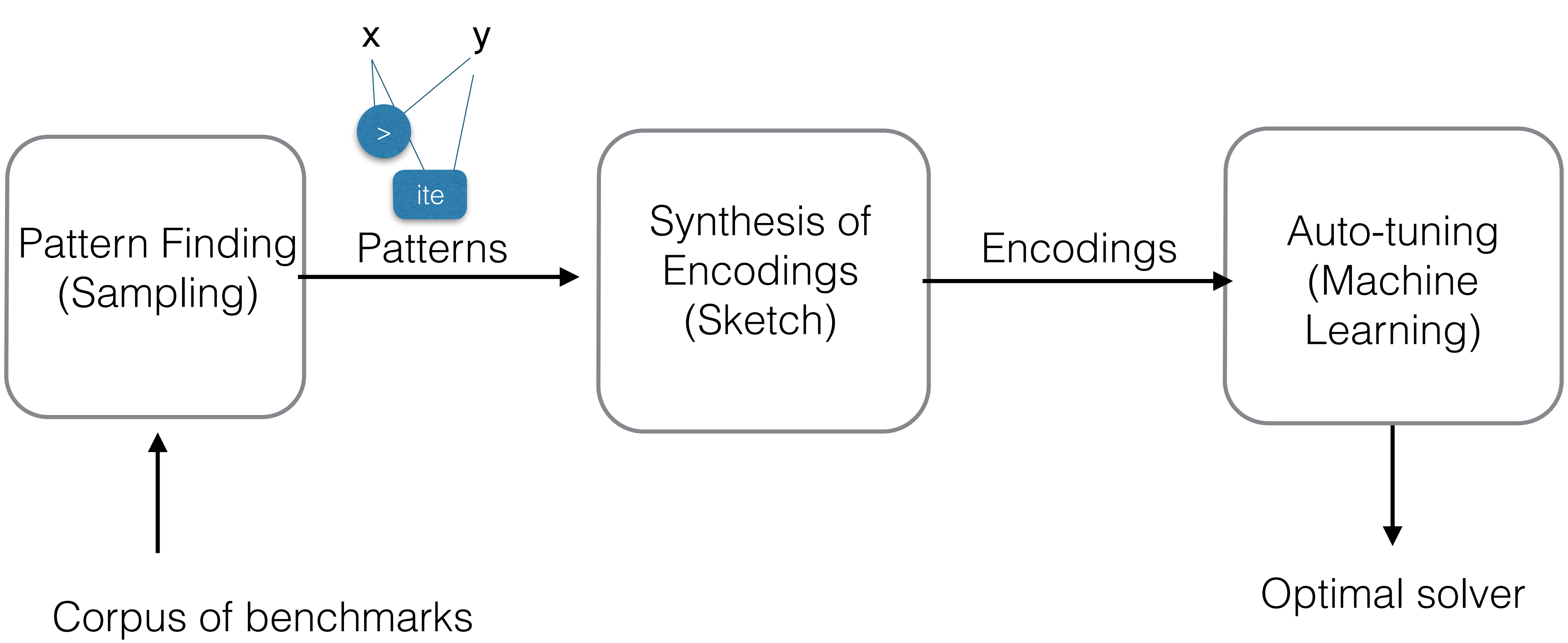
Do these encodings actually improve the performance?



What patterns to target?

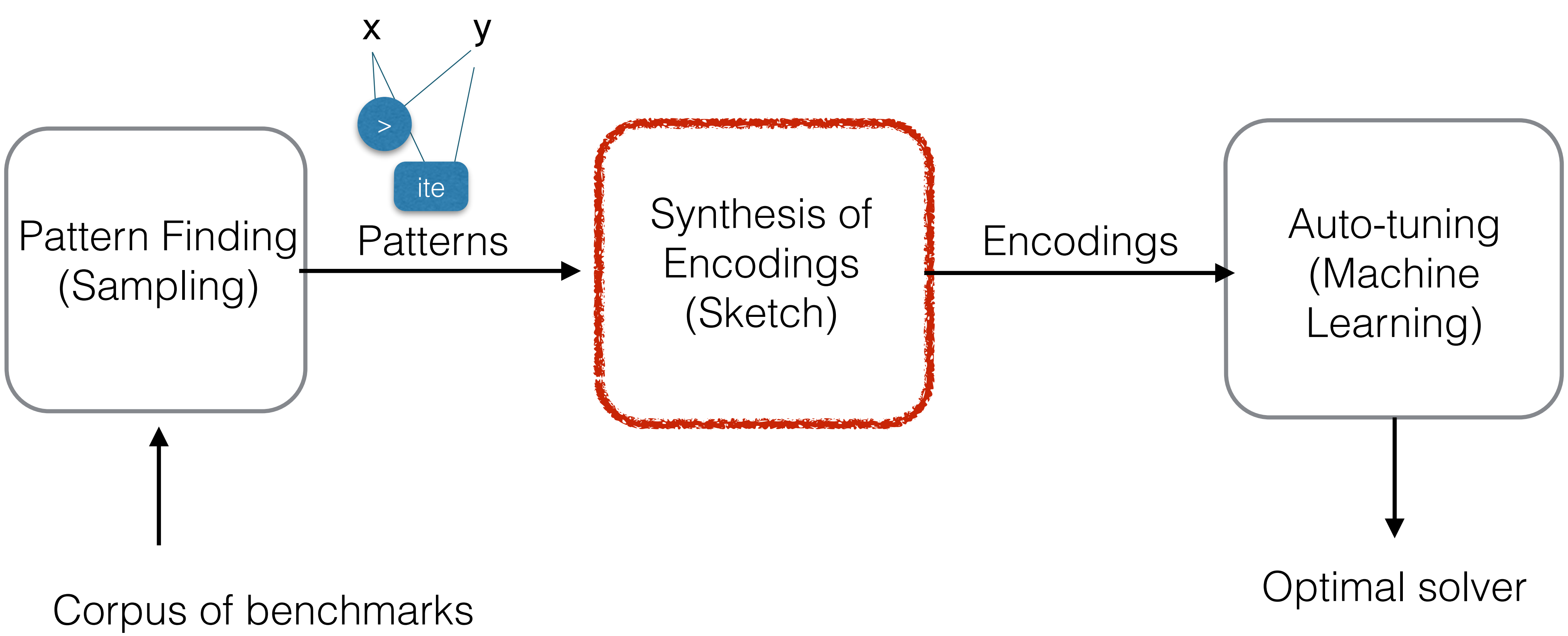
How do we come up with “optimal” encoding for a pattern?

Do these encodings actually improve the performance?



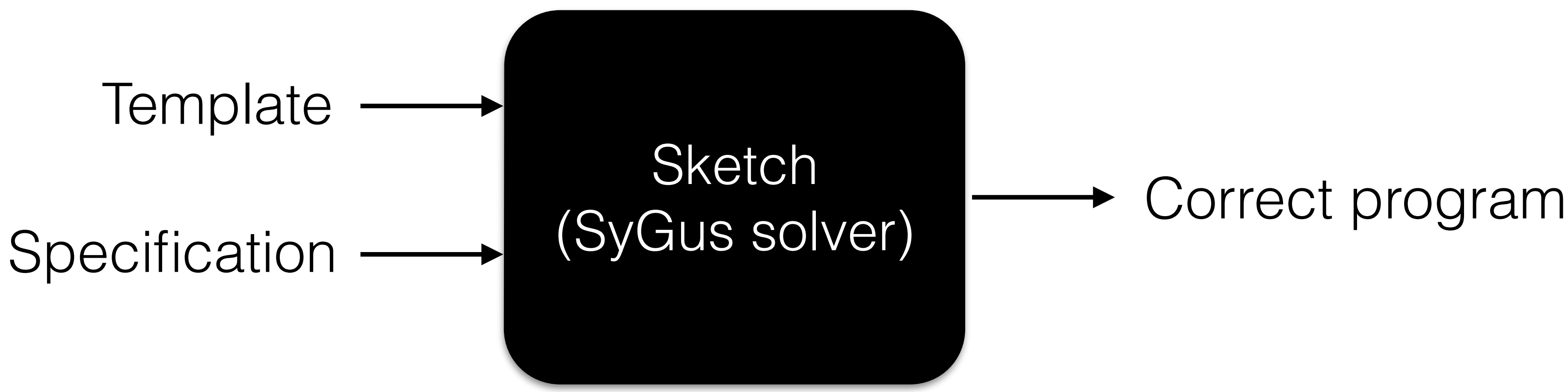
Related Work

- Automatic Generation of Propagation Complete SAT Encodings (VMCAI'16)
- Algorithm configuration for solver parameters
- Logic synthesis based SMT solvers


x
y
>
ite
Patterns
Pattern Finding
(Sampling)
Corpus of benchmarks
Synthesis of
Encodings
(Sketch)
Encodings
Auto-tuning
(Machine
Learning)
Optimal solver

Synthesis as a SyGus problem

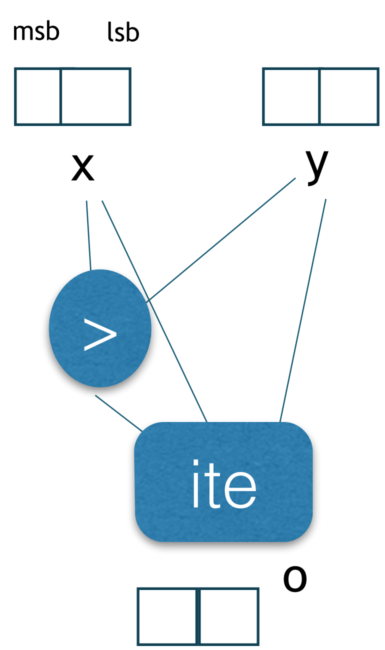
Boolean predicate P $\longrightarrow$ CNF clauses C



Templates

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$

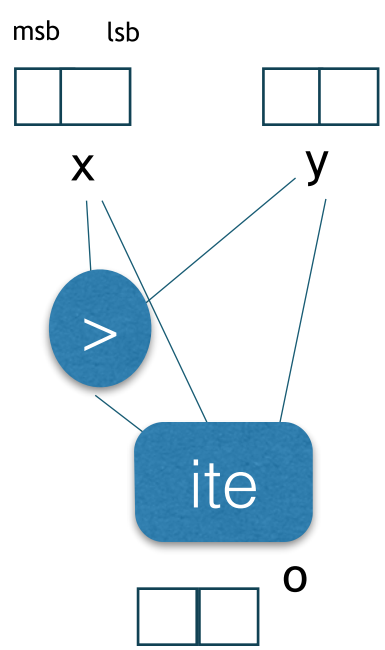


$$
\begin{aligned}
&t1 = true\\
&t2 = true\\
&for\ i\ from\ N\ to\ 1:\\
&\quad t3 = newVar\\
&\quad t4 = newVar\\
&\quad clause(\{x[i], y[i], \overline{o[i]}\})\\
&\quad clause(\{x[i], \overline{t1}, t3\})\\
&\quad clause(\{x[i], \overline{t2}, o[i], t4\})\\
&\quad clause(\{x[i], \overline{o[i]}, \overline{t3}\})\\
&\quad clause(\{\overline{x[i]}, y[i], o[i]\})\\
&\quad clause(\{\overline{x[i]}, \overline{t2}, o[i]\})\\
&\quad clause(\{\overline{x[i]}, \overline{t2}, t4\})\\
&\quad clause(\{y[i], \overline{t2}, t4\})\\
&\quad clause(\{y[i], \overline{o[i]}, \overline{t3}\})\\
&\quad clause(\{\overline{y[i]}, \overline{t1}, t3\})\\
&\quad clause(\{\overline{y[i]}, o[i], \overline{t3}\})\\
&\quad clause(\{t1, \overline{t3}\})\\
&\quad clause(\{\overline{t1}, o[i], t3\})\\
&\quad clause(\{t2, \overline{t4}\})\\
&\quad clause(\{t3, \overline{t4}\})\\
&\quad\ t1 = t3\\
&\quad\ t2 = t4
\end{aligned}
$$

Templates

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$

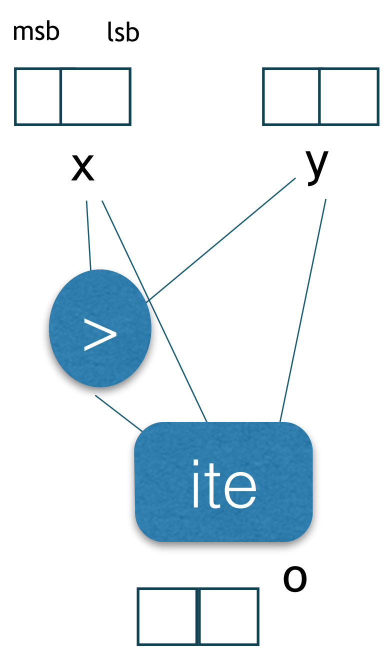


$t1 = true$
$t2 = true$
$for\ i\ from\ N\ to\ 1:$
$\quad t3 = newVar$
$\quad t4 = newVar$
$\quad clause(\{x[i], y[i], \overline{o[i]}\})$
$\quad clause(\{x[i], \overline{t1}, t3\})$
$\quad clause(\{x[i], \overline{t2}, o[i], t4\})$
$\quad clause(\{x[i], \overline{o[i]}, \overline{t3}\})$
$\quad clause(\{\overline{x[i]}, y[i], o[i]\})$
$\quad clause(\{\overline{x[i]}, \overline{t2}, o[i]\})$
$\quad clause(\{\overline{x[i]}, \overline{t2}, t4\})$
$\quad clause(\{y[i], \overline{t2}, t4\})$
$\quad clause(\{y[i], \overline{o[i]}, \overline{t3}\})$
$\quad clause(\{\overline{y[i]}, \overline{t1}, t3\})$
$\quad clause(\{\overline{y[i]}, o[i], \overline{t3}\})$
$\quad clause(\{t1, \overline{t3}\})$
$\quad clause(\{\overline{t1}, o[i], t3\})$
$\quad clause(\{t2, \overline{t4}\})$
$\quad clause(\{t3, \overline{t4}\})$
$\quad t1 = t3$
$\quad t2 = t4$

Templates

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$

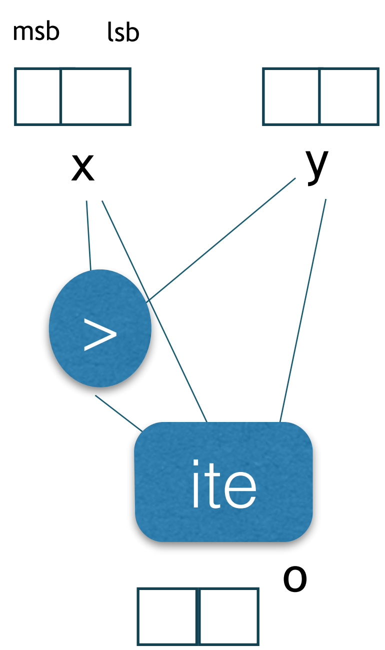


$t1 = true$
$t2 = true$
$for\ i\ from\ N\ to\ 1:$
  $t3 = newVar$
  $t4 = newVar$
  $clause(\{x[i], y[i], \overline{o[i]}\})$
  $clause(\{x[i], \overline{t1}, t3\})$
  $clause(\{x[i], \overline{t2}, o[i], t4\})$
  $clause(\{x[i], \overline{o[i]}, \overline{t3}\})$
  $clause(\{\overline{x[i]}, \overline{y[i]}, o[i]\})$
  $clause(\{\overline{x[i]}, \overline{t2}, o[i]\})$
  $clause(\{\overline{x[i]}, \overline{t2}, t4\})$
  $clause(\{y[i], \overline{t2}, t4\})$
  $clause(\{y[i], \overline{o[i]}, \overline{t3}\})$
  $clause(\{\overline{y[i]}, \overline{t1}, t3\})$
  $clause(\{\overline{y[i]}, o[i], \overline{t3}\})$
  $clause(\{t1, \overline{t3}\})$
  $clause(\{\overline{t1}, o[i], t3\})$
  $clause(\{t2, \overline{t4}\})$
  $clause(\{t3, \overline{t4}\})$
  $t1 = t3$
  $t2 = t4$

Templates

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$



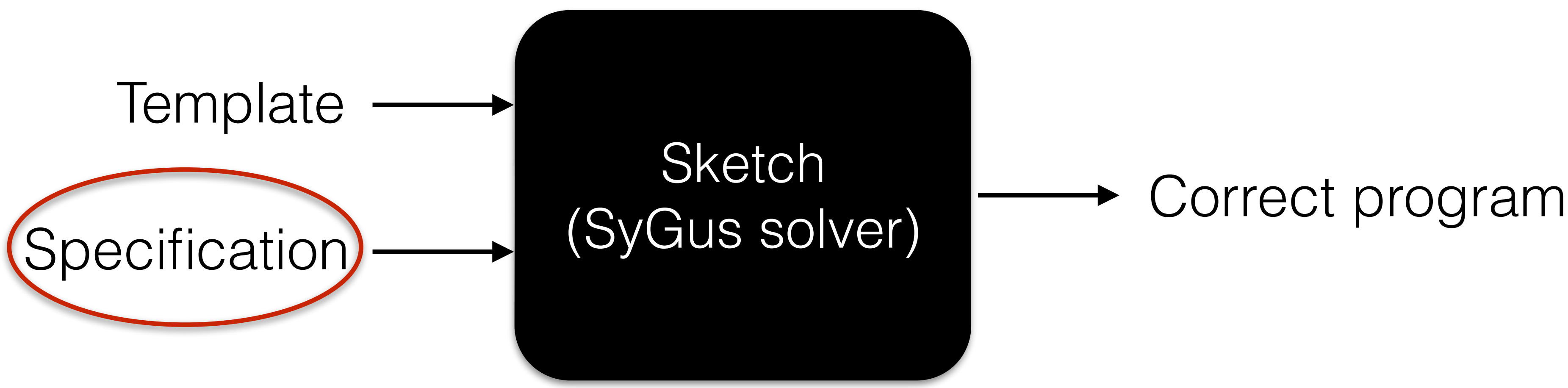
$$
\begin{array}{l}
t1 = true \\
t2 = true \\
for\ i\ from\ N\ to\ 1: \\
\quad t3 = newVar \\
\quad t4 = newVar \\
\quad clause(\{x[i], y[i], \overline{o[i]}\}) \\
\quad clause(\{x[i], \overline{t1}, t3\}) \\
\quad clause(\{x[i], \overline{t2}, o[i], t4\}) \\
\quad clause(\{x[i], \overline{o[i]}, \overline{t3}\}) \\
\quad clause(\{\overline{x[i]}, \overline{y[i]}, o[i]\}) \\
\quad clause(\{\overline{x[i]}, \overline{t2}, o[i]\}) \\
\quad clause(\{\overline{x[i]}, \overline{t2}, t4\}) \\
\quad clause(\{y[i], \overline{t2}, t4\}) \\
\quad clause(\{y[i], \overline{o[i]}, \overline{t3}\}) \\
\quad clause(\{\overline{y[i]}, \overline{t1}, t3\}) \\
\quad clause(\{\overline{y[i]}, o[i], \overline{t3}\}) \\
\quad clause(\{t1, \overline{t3}\}) \\
\quad clause(\{\overline{t1}, o[i], t3\}) \\
\quad clause(\{t2, \overline{t4}\}) \\
\quad clause(\{t3, \overline{t4}\}) \\
\quad t1 = t3 \\
\quad t2 = t4
\end{array}
$$

Templates

$$
\begin{aligned}
&t1 = ??\\
&t2 = ??\\
&for\ i\ from\ ??\ to\ ??:\\
&\quad t3 = newVar\\
&\quad t4 = newVar\\
&\quad genClauses(x[i], y[i], o[i], t1, t2, t3, t4)\\
&\quad t1 = t3\\
&\quad t2 = t4
\end{aligned}
$$

Synthesis as a SyGus problem

Boolean predicate P → CNF clauses C



Specification

Boolean predicate P → CNF clauses C

Correctness: $\forall \sigma.\ P(\sigma) = C(\sigma)$ σ = variables assignment

$d = ite(a, b, c) \wedge e = d$ $a = T, b = T, c = F, d = T, e = T$

$a \vee c \vee \overline{d}$	$T \vee F \vee F$
$a \vee \overline{c} \vee d$	$T \vee T \vee T$
$\overline{a} \vee b \vee \overline{d}$	$F \vee T \vee F$
$\overline{a} \vee \overline{b} \vee d$	$F \vee F \vee T$
$b \vee c \vee \overline{d}$	$T \vee F \vee F$
$\overline{b} \vee \overline{c} \vee d$	$F \vee T \vee T$
$d \vee \overline{e}$	$T \vee F$
$\overline{d} \vee e$	$F \vee T$

$P(\sigma) = C(\sigma) = T$

Specification

Boolean predicate P → CNF clauses C

Correctness: $\forall \sigma.\ P(\sigma) = C(\sigma)$ σ = variables assignment

$d = ite(a, b, c) \wedge e = d$ $a = F, b = T, c = F, d = T, e = T$

$a \vee c \vee \overline{d}$	$F \vee F \vee F$
$a \vee \overline{c} \vee d$	$F \vee T \vee T$
$\overline{a} \vee b \vee \overline{d}$	$T \vee T \vee F$
$\overline{a} \vee \overline{b} \vee d$	$T \vee F \vee T$
$b \vee c \vee \overline{d}$	$T \vee F \vee F$
$\overline{b} \vee \overline{c} \vee d$	$F \vee T \vee T$
$d \vee \overline{e}$	$T \vee F$
$\overline{d} \vee e$	$F \vee T$

$P(\sigma) = C(\sigma) = F$

Specification

Boolean predicate P $\longrightarrow$ CNF clauses C

Maximal Propagation $\forall \sigma.\ \mathit{satisfiable}(\sigma, P) \implies$

$$d = ite(a, b, c) \wedge e = d$$

$$b = T, c = T$$

$$a \vee c \vee \overline{d}$$
$$a \vee \overline{c} \vee d$$
$$\overline{a} \vee b \vee \overline{d}$$
$$\overline{a} \vee \overline{b} \vee d$$
$$b \vee c \vee \overline{d}$$
$$\overline{b} \vee \overline{c} \vee d$$
$$d \vee \overline{e}$$
$$\overline{d} \vee e$$

Specification

Boolean predicate P $\longrightarrow$ CNF clauses C

Maximal Propagation

$$\forall \sigma.\ satisfiable(\sigma, P) \implies$$
$$\forall x_i, b_i.\ (forces(\sigma, P, x_i, b_i) \implies$$

$$d = ite(a, b, c) \wedge e = d$$

$$b = T, c = T \implies e = T$$

$$a \vee c \vee \overline{d}$$
$$a \vee \overline{c} \vee d$$
$$\overline{a} \vee b \vee \overline{d}$$
$$\overline{a} \vee \overline{b} \vee d$$
$$b \vee c \vee \overline{d}$$
$$\overline{b} \vee \overline{c} \vee d$$
$$d \vee \overline{e}$$
$$\overline{d} \vee e$$

Specification

Boolean predicate P → CNF clauses C

Maximal Propagation

$$\forall \sigma.\ satisfiable(\sigma, P) \implies \forall x_i, b_i.\ (forces(\sigma, P, x_i, b_i) \implies UP(C, \sigma) \sqsubseteq extend(\sigma, x_i, b_i))$$

$$d = ite(a, b, c) \wedge e = d \qquad\qquad b = T, c = T \implies e = T$$

$a \vee c \vee \overline{d}$	$a \vee T \vee \overline{d}$
$a \vee \overline{c} \vee d$	$a \vee F \vee d$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee \overline{d}$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee d$
$b \vee c \vee \overline{d}$	$T \vee T \vee \overline{d}$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee d$
$d \vee \overline{e}$	$d \vee \overline{e}$
$\overline{d} \vee e$	$\overline{d} \vee e$

Specification

Boolean predicate P → CNF clauses C

Maximal Propagation

$$\forall \sigma.\ satisfiable(\sigma, P) \implies$$
$$\forall x_i, b_i.\ (forces(\sigma, P, x_i, b_i) \implies UP(C, \sigma) \sqsubseteq extend(\sigma, x_i, b_i))$$

$$d = ite(a, b, c) \wedge e = d$$

$$b = T, c = T, d = T \implies e = T$$

$a \vee c \vee \overline{d}$	$a \vee T \vee F$
$a \vee \overline{c} \vee d$	$a \vee F \vee T$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee F$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee T$
$b \vee c \vee \overline{d}$	$T \vee T \vee F$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee T$
$d \vee \overline{e}$	$T \vee \overline{e}$
$\overline{d} \vee e$	$F \vee e$

Specification

Boolean predicate P $\longrightarrow$ CNF clauses C

Maximal Propagation

$$\forall \sigma.\ satisfiable(\sigma, P) \implies \forall x_i, b_i.\ (forces(\sigma, P, x_i, b_i) \implies UP(C, \sigma) \sqsubseteq extend(\sigma, x_i, b_i))$$

$$d = ite(a, b, c) \wedge e = d \qquad b = T, c = T, d = T, e = T \implies e = T$$

$a \vee c \vee \overline{d}$	$a \vee T \vee F$
$a \vee \overline{c} \vee d$	$a \vee F \vee T$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee F$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee T$
$b \vee c \vee \overline{d}$	$T \vee T \vee F$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee T$
$d \vee \overline{e}$	$T \vee \overline{e}$
$\overline{d} \vee e$	$F \vee e$

Specification

Boolean predicate P → CNF clauses C

Maximal Propagation

$$\forall \sigma.\ satisfiable(\sigma, P) \implies \forall x_i, b_i.\ (forces(\sigma, P, x_i, b_i) \implies UP(C, \sigma) \sqsubseteq extend(\sigma, x_i, b_i))$$

$$d = ite(a, b, c) \wedge e = d$$

$$b = T, c = T, d = T, e = T \implies e = T$$

$a \vee c \vee \overline{d}$	$a \vee T \vee F$
$a \vee \overline{c} \vee d$	$a \vee F \vee T$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee F$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee T$
$b \vee c \vee \overline{d}$	$T \vee T \vee F$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee T$
$d \vee \overline{e}$	$T \vee \overline{e}$
$\overline{d} \vee e$	$F \vee e$

Specification

Boolean predicate P $\longrightarrow$ CNF clauses C

Maximal Propagation

$$\forall \sigma.\ \mathit{satisfiable}(\sigma, P) \implies \forall x_i, b_i.\ (\mathit{forces}(\sigma, P, x_i, b_i) \implies UP(C, \sigma) \sqsubseteq \mathit{extend}(\sigma, x_i, b_i))$$

Difficult for synthesis tools to reason about

Is there a better specification for maximal propagation?

Synthesis Friendly Spec

Boolean predicate P → CNF clauses C

1. $\forall \sigma.\ satisfiable(\sigma, P) \implies C(\sigma) \neq false$

$$d = ite(a, b, c) \wedge e = d$$

$$b = T, c = T$$

is satisfiable

$a \vee c \vee \overline{d}$	$a \vee T \vee \overline{d}$
$a \vee \overline{c} \vee d$	$a \vee F \vee d$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee \overline{d}$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee d$
$b \vee c \vee \overline{d}$	$T \vee T \vee \overline{d}$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee d$
$d \vee \overline{e}$	$d \vee \overline{e}$
$\overline{d} \vee e$	$\overline{d} \vee e$

No false clauses

Synthesis Friendly Spec

Boolean predicate P $\longrightarrow$ CNF clauses C

1. $\forall \sigma.\ satisfiable(\sigma, P) \implies C(\sigma) \neq false$
2. $\forall \sigma.\ maypropagate(\sigma, P) \implies C(\sigma)$ has a unit clause

$d = ite(a, b, c) \wedge e = d$

$b = T, c = T$

Can propagate e to T

$a \vee c \vee \overline{d}$	$a \vee T \vee \overline{d}$
$a \vee \overline{c} \vee d$	$a \vee F \vee d$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee \overline{d}$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee d$
$b \vee c \vee \overline{d}$	$T \vee T \vee \overline{d}$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee d$
$d \vee \overline{e}$	$d \vee \overline{e}$
$\overline{d} \vee e$	$\overline{d} \vee e$

Has a unit clause

Synthesis Friendly Spec

Boolean predicate P $\longrightarrow$ CNF clauses C

1. $\forall \sigma.\ \mathit{satisfiable}(\sigma, P) \implies C(\sigma) \neq \mathit{false}$
2. $\forall \sigma.\ \mathit{maypropagate}(\sigma, P) \implies C(\sigma)$ has a unit clause

$$d = ite(a, b, c) \wedge e = d$$

$$b = T, c = T, d = T$$

Can still propagate e to T

$a \vee c \vee \overline{d}$	$a \vee T \vee F$
$a \vee \overline{c} \vee d$	$a \vee F \vee T$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee F$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee T$
$b \vee c \vee \overline{d}$	$T \vee T \vee F$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee T$
$d \vee \overline{e}$	$T \vee \overline{e}$
$\overline{d} \vee e$	$F \vee e$

Has a unit clause

Synthesis Friendly Spec

Boolean predicate P $\longrightarrow$ CNF clauses C

1. $\forall \sigma.\ satisfiable(\sigma, P) \implies C(\sigma) \neq false$
2. $\forall \sigma.\ maypropagate(\sigma, P) \implies C(\sigma)$ has a unit clause
3. $\forall \sigma.\ unsatisfiable(\sigma, P) \implies C(\sigma) = false\ (or)\ C(\sigma)$ has a unit clause

$d = ite(a, b, c) \wedge e = d$ $\qquad$ $b = T, c = T, e = F$

Unsatisfiable

$a \vee c \vee \overline{d}$	$a \vee T \vee \overline{d}$
$a \vee \overline{c} \vee d$	$a \vee F \vee d$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee \overline{d}$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee d$
$b \vee c \vee \overline{d}$	$T \vee T \vee \overline{d}$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee d$
$d \vee \overline{e}$	$d \vee T$
$\overline{d} \vee e$	$\overline{d} \vee F$

Has unit clauses

Synthesis Friendly Spec

Boolean predicate P $\longrightarrow$ CNF clauses C

1. $\forall \sigma.\ \mathit{satisfiable}(\sigma, P) \implies C(\sigma) \neq \mathit{false}$
2. $\forall \sigma.\ \mathit{maypropagate}(\sigma, P) \implies C(\sigma)$ has a unit clause
3. $\forall \sigma.\ \mathit{unsatisfiable}(\sigma, P) \implies C(\sigma) = \mathit{false}\ (\mathit{or})\ C(\sigma)$ has a unit clause

$d = ite(a, b, c) \wedge e = d$

$b = T, c = T, e = F, d = F$

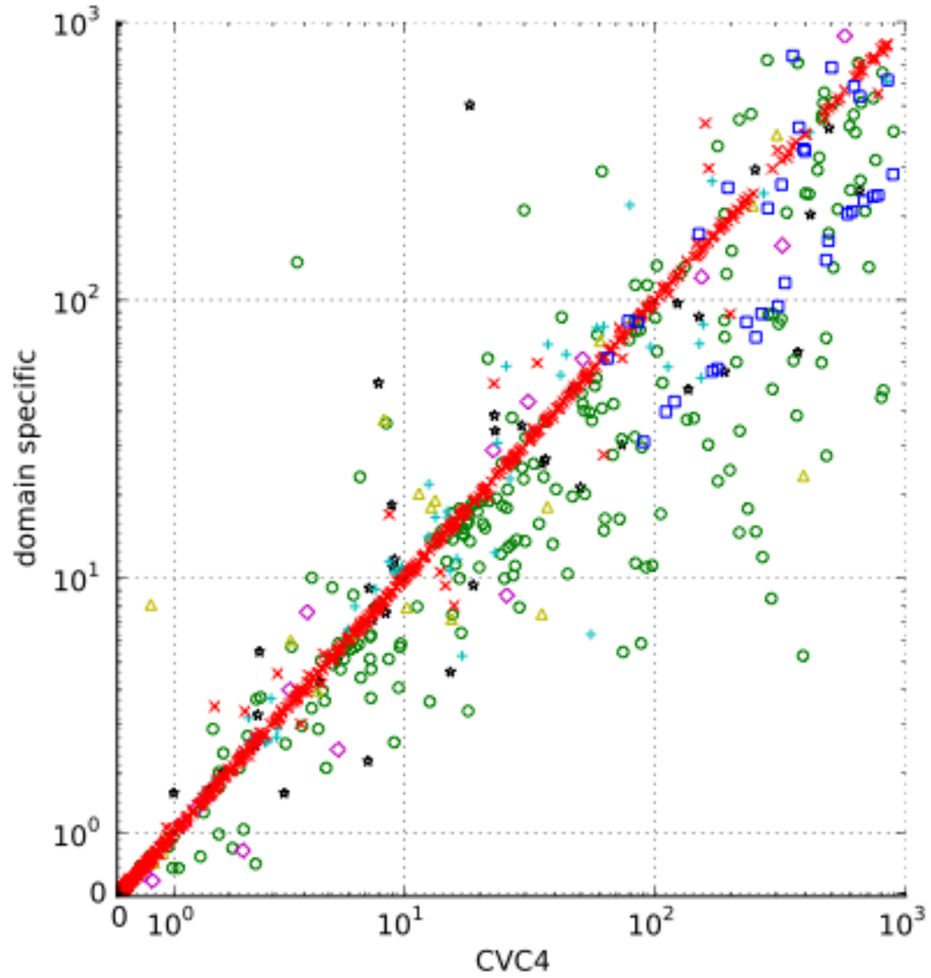
Also unsatisfiable

$a \vee c \vee \overline{d}$	$a \vee T \vee T$
$a \vee \overline{c} \vee d$	$a \vee F \vee F$
$\overline{a} \vee b \vee \overline{d}$	$\overline{a} \vee T \vee T$
$\overline{a} \vee \overline{b} \vee d$	$\overline{a} \vee F \vee F$
$b \vee c \vee \overline{d}$	$T \vee T \vee T$
$\overline{b} \vee \overline{c} \vee d$	$F \vee F \vee F$
$d \vee \overline{e}$	$F \vee T$
$\overline{d} \vee e$	$T \vee F$

Has a false clause

Does this buy you anything?

- 7 domains from bit-vector category in SMT Comp'15



* Excluding the benchmarks used for training

Solve more problems

Benchmark Family	Solved by CVC4 → Our Solver
Log-slicing (79)	*33 → 62*
ASP (365)	*240 → 288*
Mcm (61)	*40 → 43*
Brummayerbiere2 (33)	*28 → 29*
Float (62)	*59 → 60*
Brummayerbiere3 (40)	*23 → 24*
Bruttomesso (676)	*623 → 623*
TOTAL	***1046 → 1129***

Solve more problems

Benchmark Family	Solved by CVC4 → Our Solver
Log-slicing (79)	*33 → 62*
ASP (365)	*240 → 288*
Mcm (61)	*40 → 43*
Brummayerbiere2 (33)	*28 → 29*
Float (62)	*59 → 60*
Brummayerbiere3 (40)	*23 → 24*
Bruttomesso (676)	*623 → 623*
TOTAL	***1046 → 1129***

83 more problems in total

Cross domain performance

Solver → / Domain ↓	log-slicing	asp	mcm	brumma2	float	brumma3	brutto
log-slicing	62	58	36	59	32	35	35
asp	227	288	255	227	236	253	240
mcm	39	38	43	40	39	39	41
brumma2	29	28	28	29	29	29	29
float	57	57	59	57	60	60	59
brumma3	22	22	25	22	23	24	23
brutto	607	606	623	609	623	623	623

Best → Worst

What did it take?

- Around 2000 benchmarks across 7 domains
- Over 200 million nodes in the high level SMT constraints
- Sampling generated ~2000 patterns (size <= 5)
- ~2000 SyGus problems to solve
- Generated ~40k to 160k lines of code per domain (30 lines per encoding)
- 8 hours of auto-tuning per domain
- On total, took 10-20 hours per domain with parallelism of 30

Conclusion

- A framework to generate the code for translating high-level constraints to CNF
- Combined synthesis with pattern finding and auto-tuning to generate domain specific solvers
- Significant performance improvement compared to CVC4

THANK YOU