Synthesis of Domain Specific Encoders for Bit-Vector Solvers

Jeevana Priya Inala
with
Rohit Singh, Armando Solar-Lezama

To appear at SAT'16

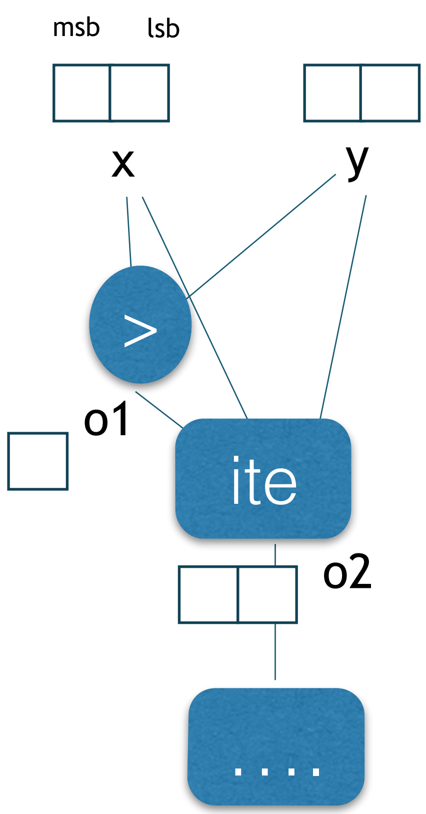
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

CSAIL

High-level constraint to CNF clauses

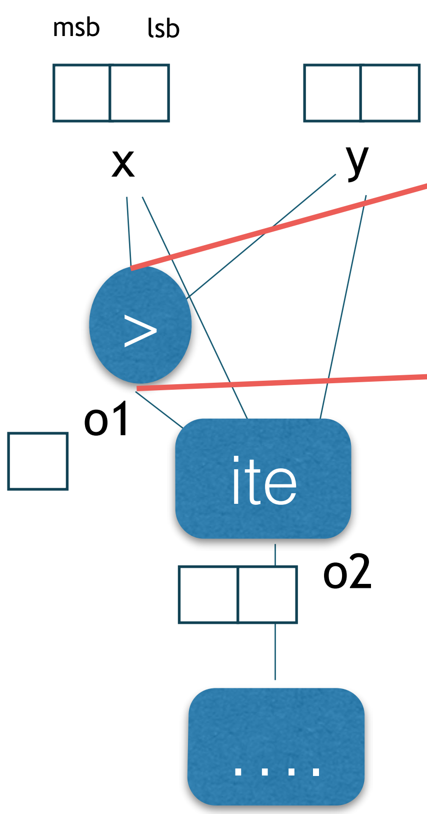
SMT solver
High-level constraint

SAT solver
CNF clauses



High-level constraint to CNF clauses

SMT solver
High-level constraint



SAT solver
CNF clauses

$\overline{y_1} \lor x_1 \lor \overline{t_1}$

$y_1 \lor \overline{x_1} \lor \overline{t_1}$

$\overline{y_1} \lor \overline{x_1} \lor t_1$

$y_1 \lor x_1 \lor t_1$

$\overline{t_2} \lor t_1$

$\overline{t_2} \lor \overline{y_0}$

$\overline{t_2} \lor x_0$

$\overline{t_1} \lor y_0 \lor \overline{x_0} \lor t_2$

$\overline{t_3} \lor \overline{y_1}$

$\overline{t_3} \lor x_1$

$y_1 \lor \overline{x_1} \lor t_3$

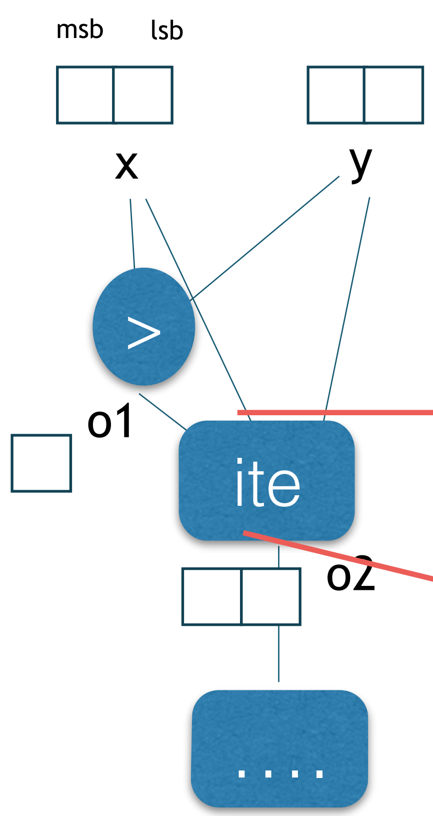
$o_1 \lor \overline{t_2}$

$o_1 \lor \overline{t_3}$

$t_2 \lor t_3 \lor \overline{o_1}$

High-level constraint to CNF clauses

SMT solver
High-level constraint



SAT solver
CNF clauses

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

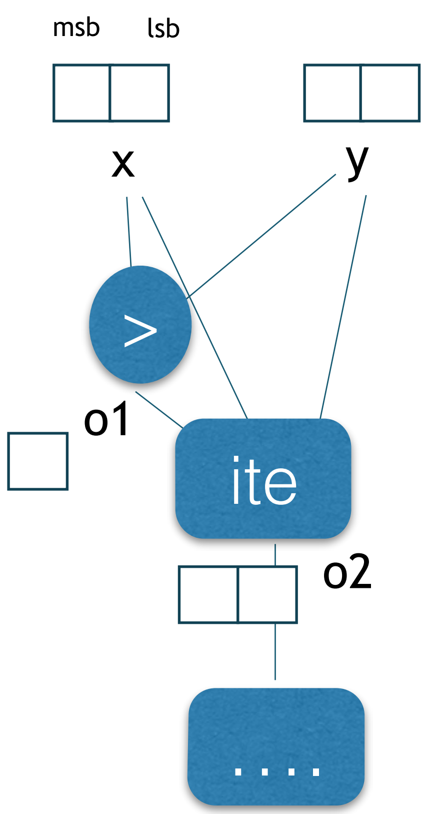
$$\overline{t_1} \vee y_0 \vee \overline{x_0} \vee t_2$$
$$\overline{t_3} \vee \overline{y_1}$$
$$\overline{t_3} \vee x_1$$
$$y_1 \vee \overline{x_1} \vee t_3$$
$$o_1 \vee \overline{t_2}$$
$$o_1 \vee \overline{t_3}$$
$$t_2 \vee t_3 \vee \overline{o_1}$$

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

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

High-level constraint to CNF clauses

SMT solver
High-level constraint



Not 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

What is an optimal encoding?

- Fewer clauses
- Fewer variables
- Maximal propagation

Maximal Propagation

- SAT solvers use unit propagation to infer variable assignments

Maximal Propagation

- SAT solvers use unit propagation to infer variable assignments

Current variables assignment

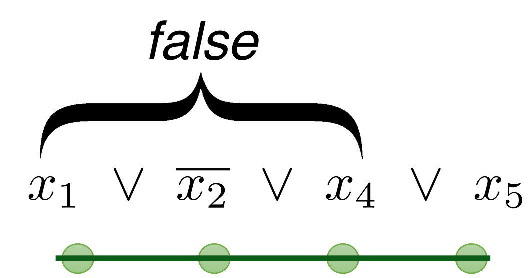
$$true \rightarrow \overline{x_1},\ x_2,\ \overline{x_3},\ \overline{x_4}$$

Maximal Propagation

- SAT solvers use unit propagation to infer variable assignments

Current variables assignment

$$true \rightarrow \overline{x_1},\ x_2,\ \overline{x_3},\ \overline{x_4}$$



Maximal Propagation

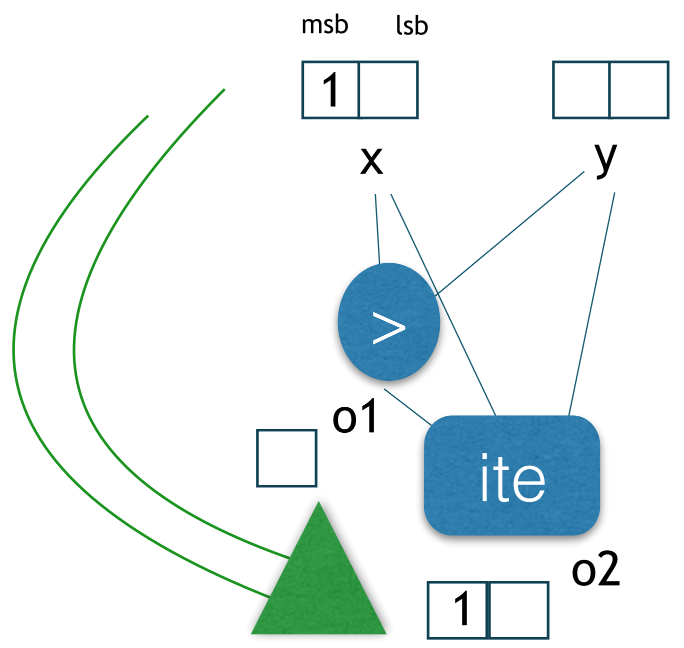
- SAT solvers use unit propagation to infer variable assignments

Current variables assignment

$$true \rightarrow \overline{x_1},\ x_2,\ \overline{x_3},\ \overline{x_4},\ x_5$$

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

Find an encoding that maximizes what we can learn through unit propagations



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

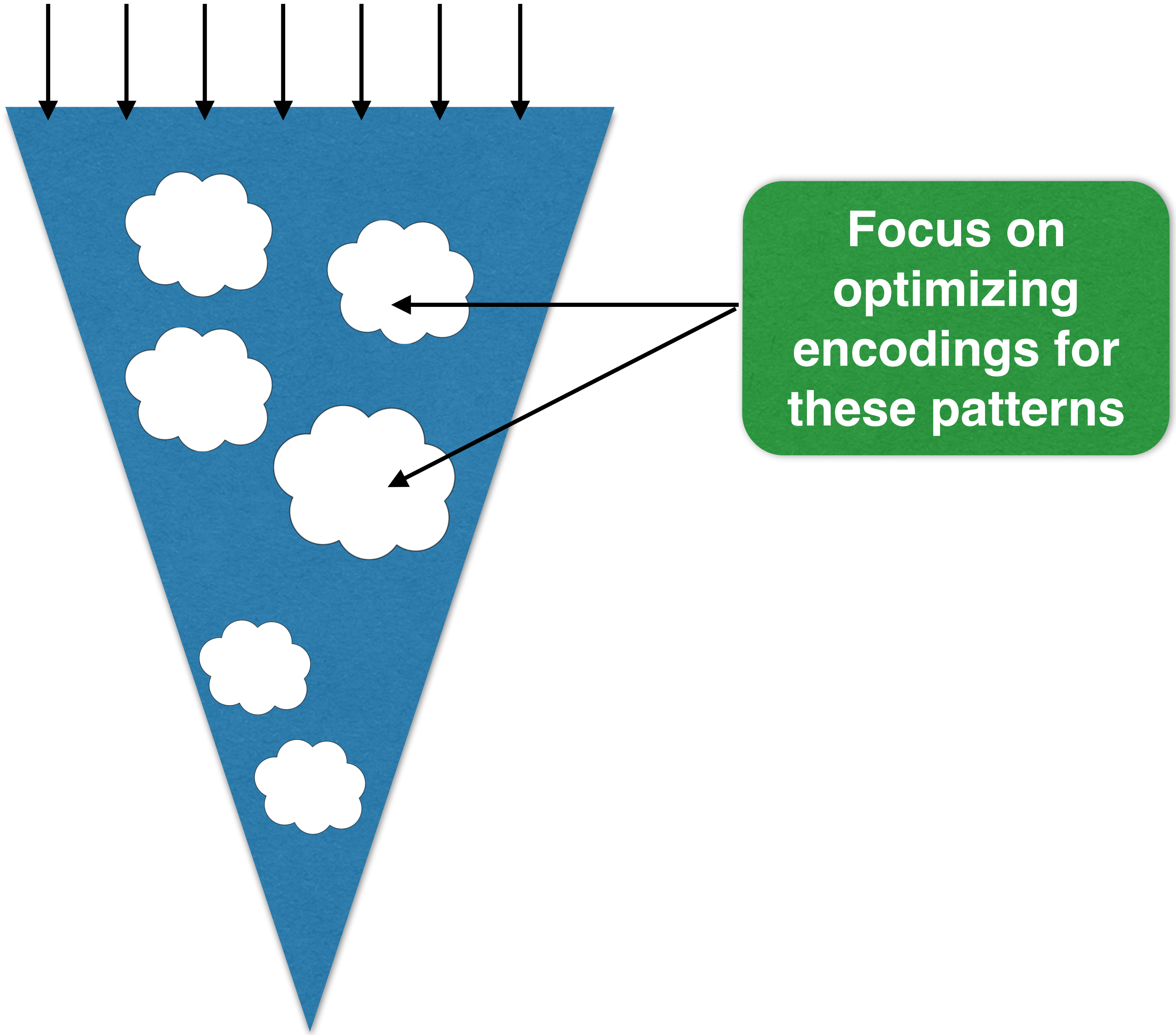
$$\overline{t_1} \vee y_0 \vee \overline{x_0} \vee t_2$$
$$\overline{t_3} \vee \overline{y_1}$$
$$\overline{t_3} \vee x_1$$
$$y_1 \vee \overline{x_1} \vee t_3$$
$$o_1 \vee \overline{t_2}$$
$$o_1 \vee \overline{t_3}$$
$$t_2 \vee t_3 \vee \overline{o_1}$$

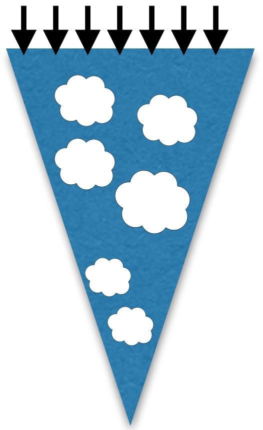
$$\overline{o_1} \vee x_1 \vee \overline{o2_1}$$
$$\overline{o_1} \vee \overline{x_1} \vee o2_1$$
$$o_1 \vee y_1 \vee \overline{o2_1}$$
$$o_1 \vee \overline{y_1} \vee o2_1$$
$$x_1 \vee y_1 \vee \overline{o2_1}$$
$$\overline{x_1} \vee \overline{y_1} \vee o2_1$$

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

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

Composing encodings does not preserve optimality

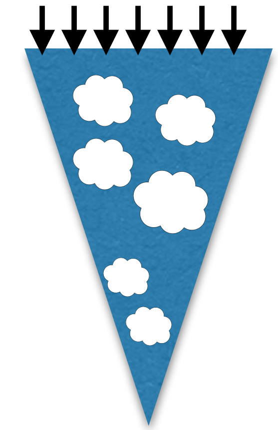

Focus on optimizing encodings for these patterns



What patterns to target?

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

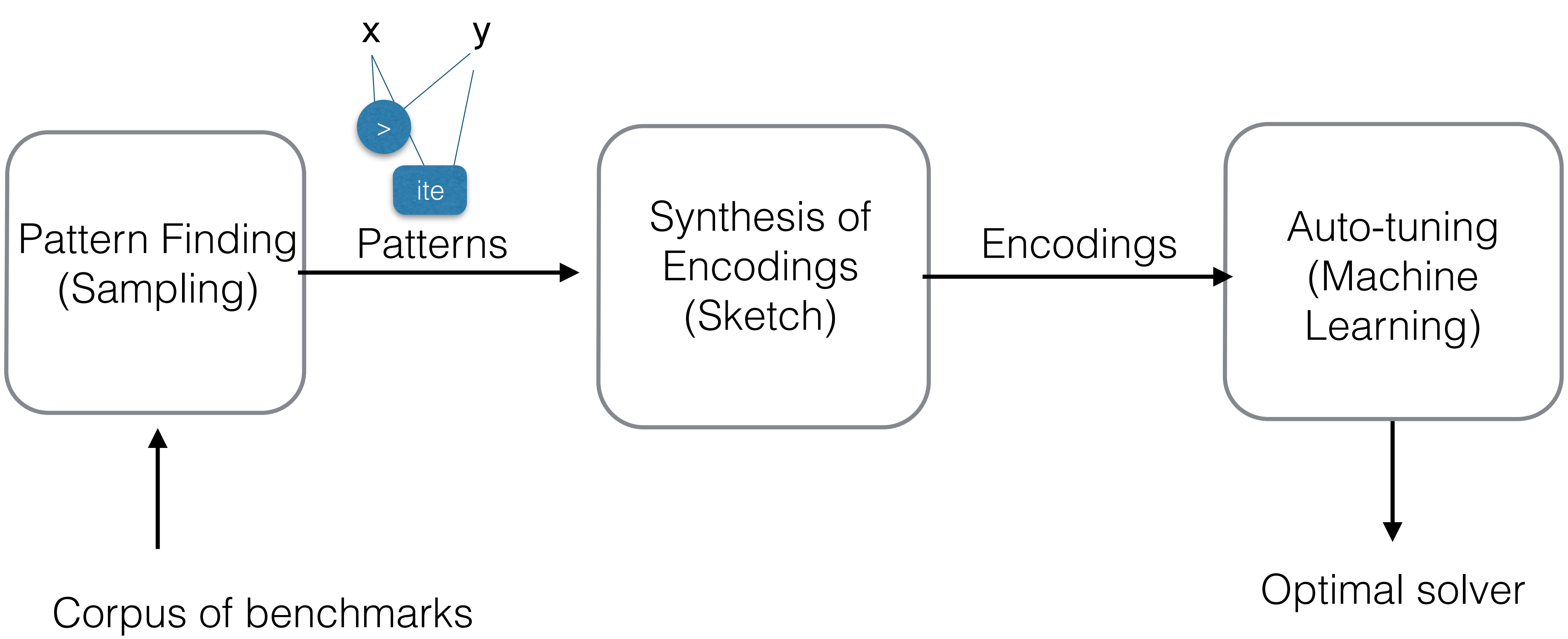
Do these encodings actually improve the performance?

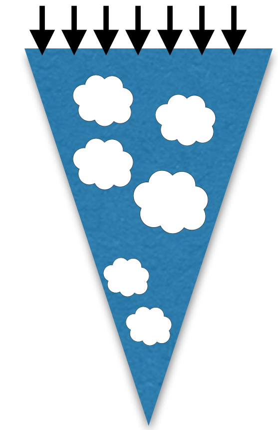


What patterns to target?

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

Do these encodings actually improve the performance?

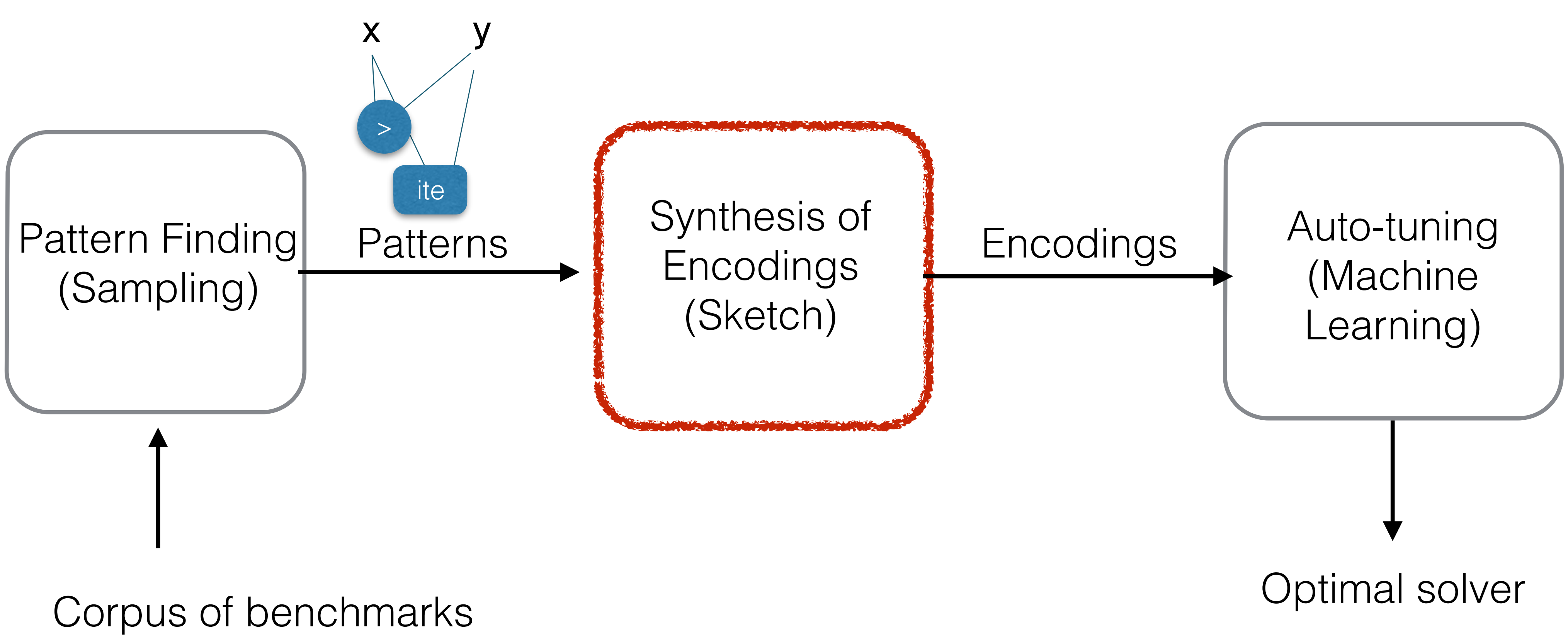




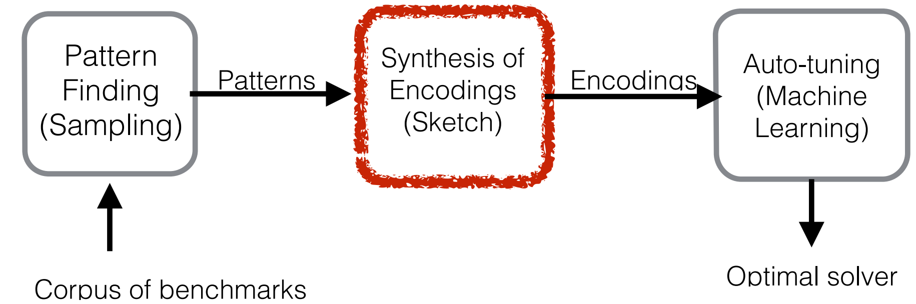
What patterns to target?

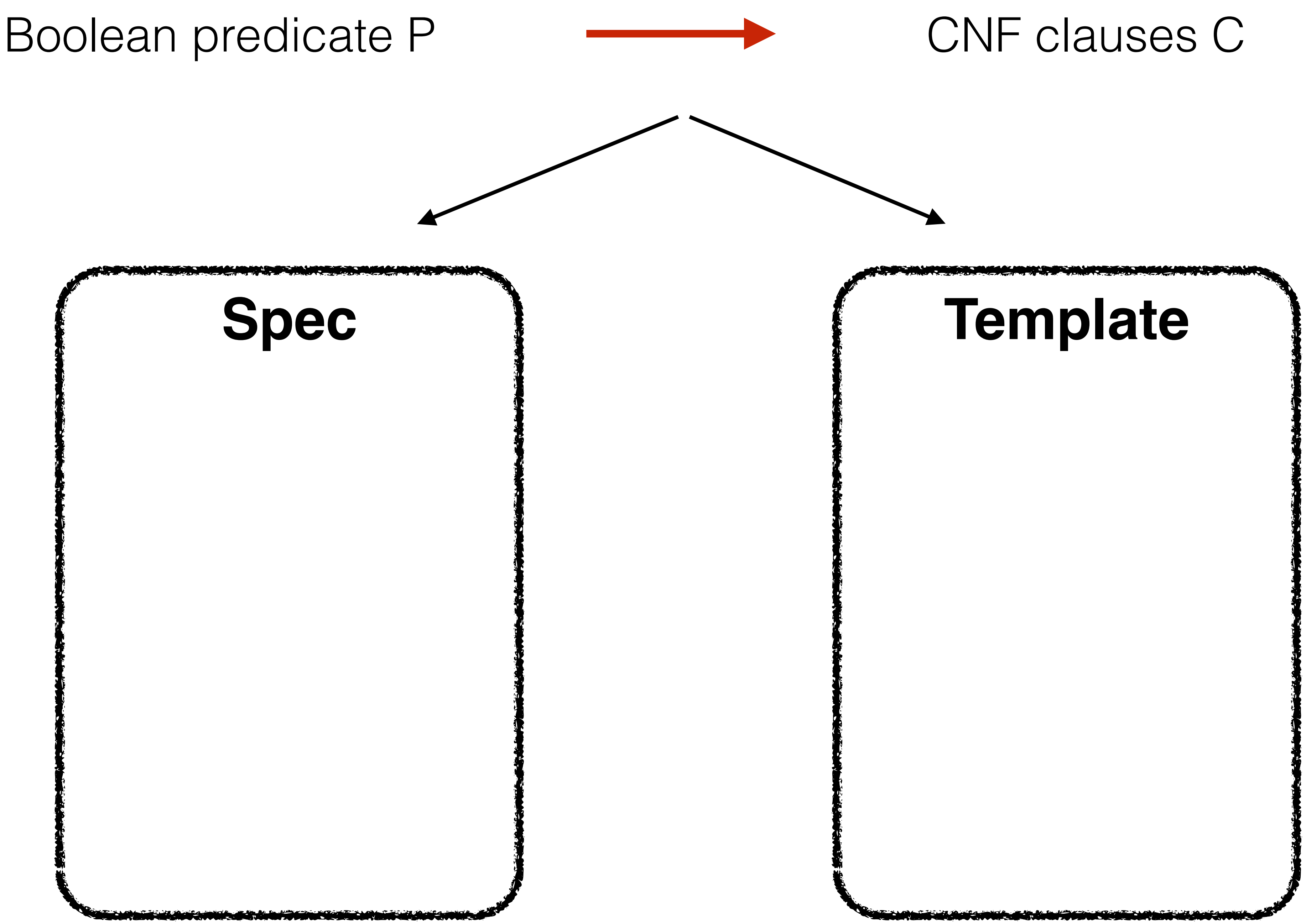
How do we come up with "optimal" encoding for a pattern?

Do these encodings actually improve the performance?

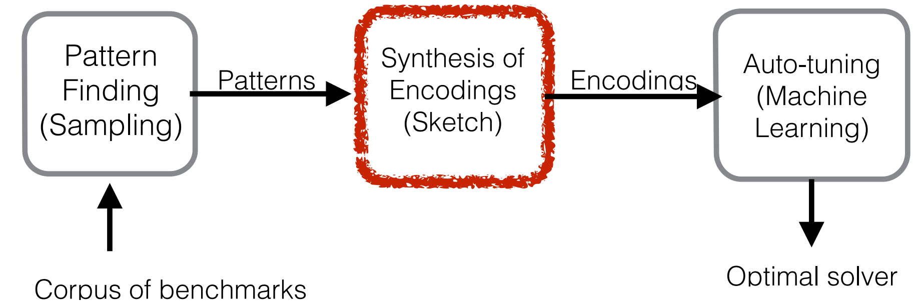


Synthesis as a SyGus problem





Synthesis as a SyGus problem



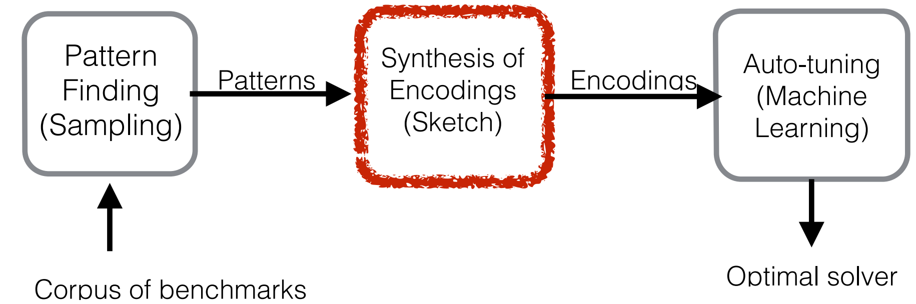
Boolean predicate P → CNF clauses C

Spec

- Involves reasoning about SAT solvers

Template

Synthesis as a SyGus problem



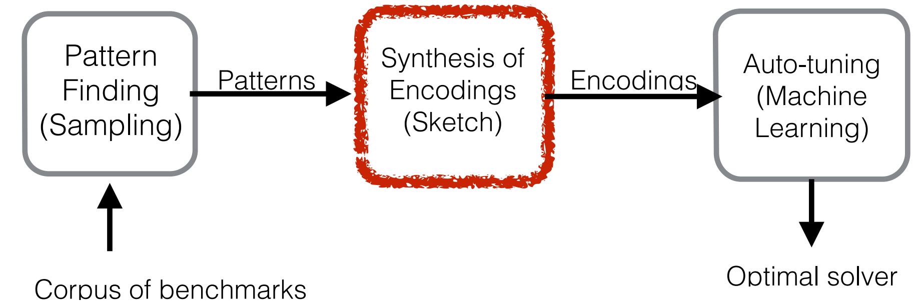
Boolean predicate P → CNF clauses C

Spec

- Involves reasoning about SAT solvers

Template

- Huge search space of encodings



Synthesis as a SyGus problem

Boolean predicate P → CNF clauses C

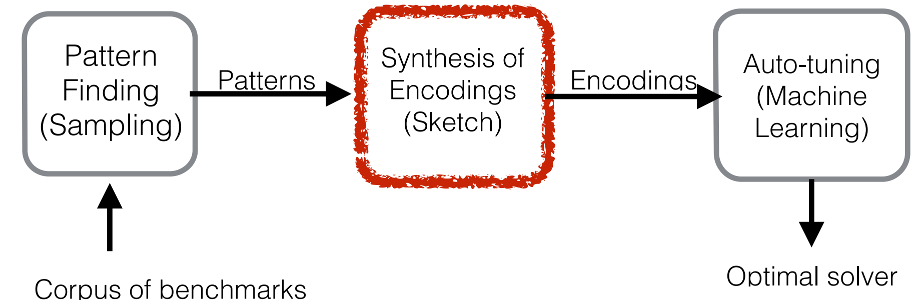
Spec

- Involves reasoning about SAT solvers
- Synthesis friendly formalism for maximal propagation

Template

- Huge search space of encodings

Synthesis as a SyGus problem



Boolean predicate P → CNF clauses C

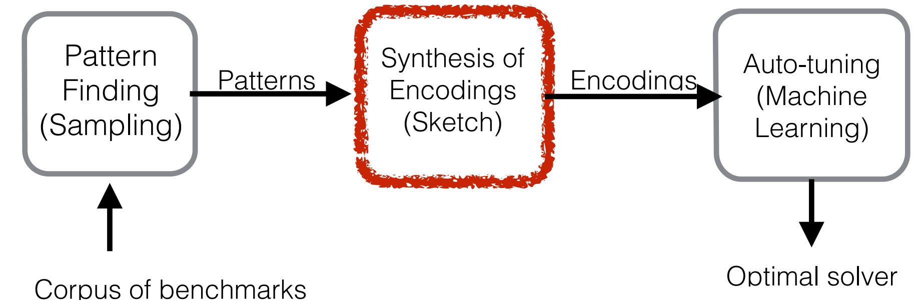
Spec

- Involves reasoning about SAT solvers
- Synthesis friendly formalism for maximal propagation

Template

- Huge search space of encodings
- Leverage structure in encodings to design templates

Synthesis as a SyGus problem

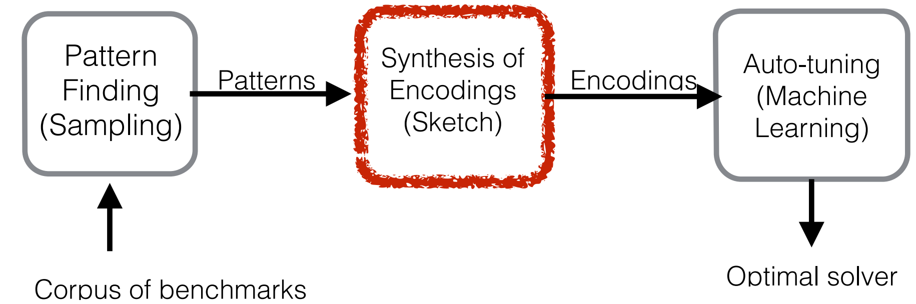


Boolean predicate P → CNF clauses C

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

→

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



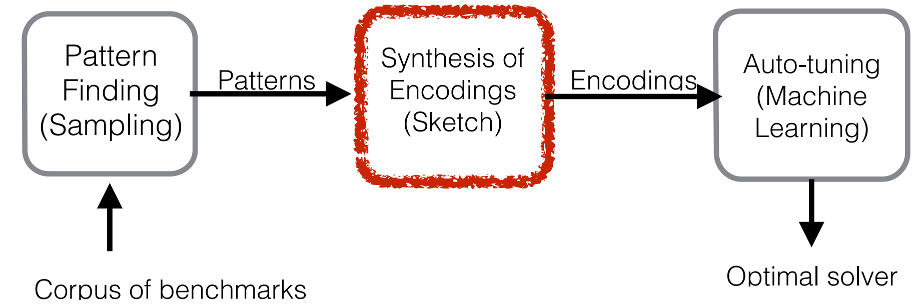
Synthesis as a SyGus problem

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$ →

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

Synthesis as a SyGus problem

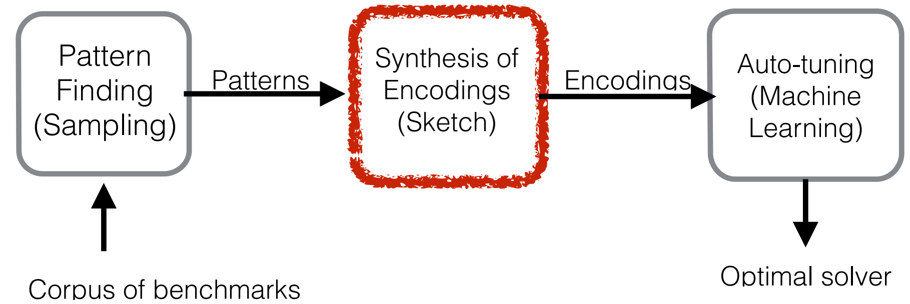


Boolean predicate P → CNF clauses C

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

→

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



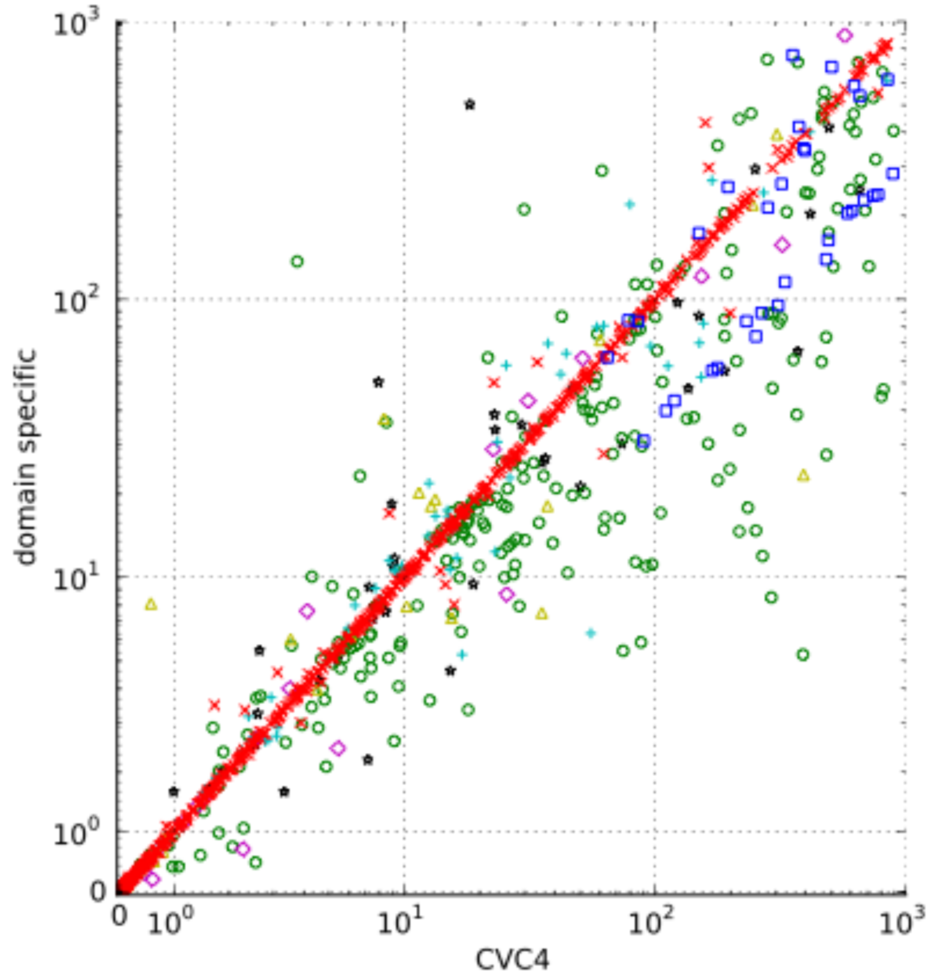
Synthesis as a SyGus problem

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$ →

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

Does this buy you anything?



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