Chronosymbolic Learning

Efficient CHC Solving with Symbolic Reasoning and Inductive Learning

Ray Luo [1,2] Xujie Si [2,3]

[1]School of Computer Science
McGill University

[2]Mila

[3]School of Computer Science
University of Toronto

Table of Contents

Table of Contents

Background: SMT (satisfiability modulo theories) Solving

Constraints like this can be solved by *SMT solvers*:

$$\forall a, b, c, \;\; a > 0 \;\wedge\; b \leq a \;\wedge\; c = 0 \;\rightarrow a + b + c \geq 0$$

- It is **SAT** iff for all assignments of (a, b, c), the constraint holds
- It is **UNSAT** iff for some assignments, the constraint does not hold
- Logical implication: $p \rightarrow q \triangleq \neg p \vee q$

One step further: Predicate

How about this?

$$\forall a, b, c,\ \ a > 0 \ \wedge\ \ b \leq a \ \wedge\ \ c = 0 \ \rightarrow p(a, b, c)$$

- $p(x, y, z)$: an unknown predicate that maps variables to True/False
- Example: $p(x, y, z) = x < y$
- It is **SAT** iff there exist an *interpretation* of p that makes the constraint holds for all assignments of (a, b, c)
- It is **UNSAT** iff there is no such *interpretation* of p that makes the constraint holds for all assignments of (a, b, c)
- A logical implication with unknown predicate is called a **Constrained Horn Clause (CHC)**

Constrained Horn Clause System

How about a *system* of such constraints?

$$\mathcal{C}_0 : \forall a, b, c,\ \ a > 0 \ \wedge\ b \leq a \ \wedge\ c = 0 \rightarrow p(a, b, c)$$
$$\mathcal{C}_1 : \forall a, b, c, c_1,\ \ c_1 = 1 + c \wedge p(a, b, c) \rightarrow q(a, b, c_1)$$
$$\mathcal{C}_2 : \forall a, b, c,\ \ b < a \cdot c \ \wedge\ q(a, b, c) \rightarrow \perp$$

- It is **SAT** iff there exist an *interpretation* of all predicates (p, q) that makes all constraints SAT
- It is **UNSAT** iff there is no such *interpretation* that makes all constraints SAT

CHCs and Formal Verification

$$x = 1 \wedge y = 0 \rightarrow p(x, y) \quad (1)$$

$$p(x, y) \wedge x' = x + y \wedge y' = y + 1 \rightarrow p(x', y') \quad (2)$$

$$p(x, y) \wedge x' = x + y \wedge y' = y + 1 \rightarrow x' >= y' \quad (3)$$

$$x = 1 \wedge y = 0 \rightarrow x >= y \quad (4)$$

```
main(){
 int x,y;
 x=1; y=0;
 while(*){
  x=x+y;
  y++;}
 assert(x>=y);}
```

$\longrightarrow$ Some nondeterministic expression of x, y

CHC is the *universal format* for formal verification!

- Can represent *any programming languages* and *any correctness specifications*
- Satisfiability of CHCs $\Leftrightarrow$ Correctness of a program
- A program $\Rightarrow$ CHCs can be automated

Table of Contents

Symbolic Reasoning-based CHC Solving

Symbolic reasoning-based CHC solvers solve CHCs by an organized way of (heuristic-guided) SMT solving

- Efficient: no need to collect data, many years of heuristic tuning
- Often focusing more on reasoning local information

Solving CHCs by data-driven approaches

Collecting data points for variables from "executing the program"

- Directly learns interpretations by induction learning
- Teacher and Learner paradigm, "Guess-and-check"
- Can take global information into account, but some cheap prior knowledge in CHC systems is ignored; Guess-and-check is slow

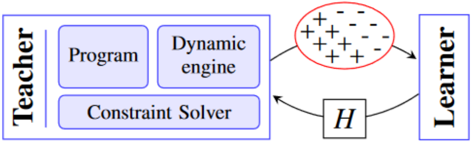
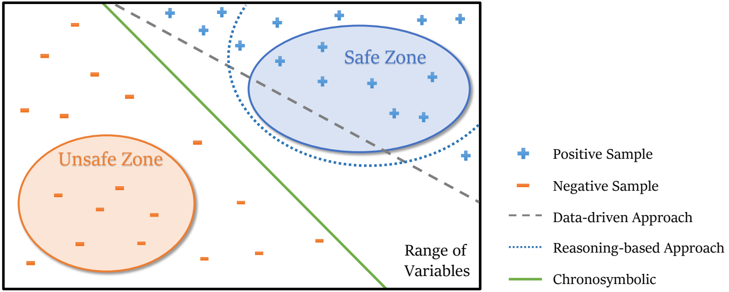
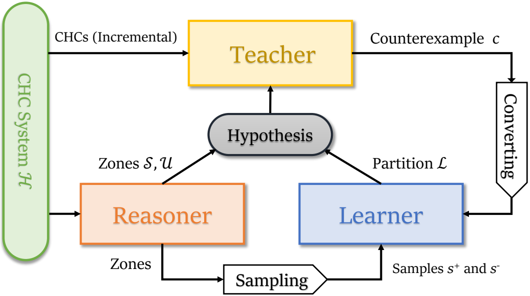


Table of Contents

Can we rethink these two distinct methods into one unified framework?

- Partial solutions of reasoning are represented by *zones*
- Data *samples* are classified by binary classifier
- Making *Chronosymbolic* hypothesis $\tilde{\mathcal{I}}_{slu}[p_i] = \mathcal{S}_{p_i} \vee (\mathcal{L}_{p_i} \wedge \neg\, \mathcal{U}_{p_i})$





Architecture of *Chronosymbolic Learning*