Program Synthesis as Dependency Quantified Formula Modulo Theory

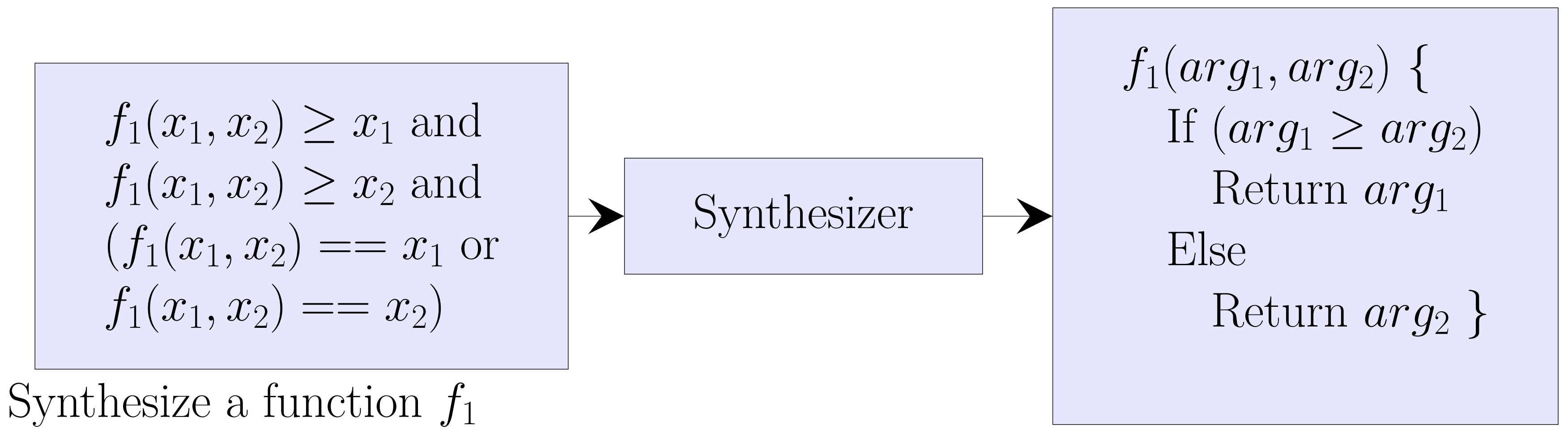
Priyanka Golia[1,2] Subhajit Roy[1] Kuldeep S. Meel[2]

[1]Indian Institute of Technology Kanpur, India [2]National University of Singapore, Singapore

Program Synthesis

Input: A specification as logic formula, underlying theory ($\mathbb{T}$), and a set of typed function symbols to synthesize.

Objective: Synthesize the function that provably satisfies the specifications.


$f_1(x_1, x_2) \geq x_1$ and
$f_1(x_1, x_2) \geq x_2$ and
$(f_1(x_1, x_2) == x_1$ or
$f_1(x_1, x_2) == x_2)$

Synthesizer

```
f_1(arg_1, arg_2) {
  If (arg_1 ≥ arg_2)
    Return arg_1
  Else
    Return arg_2 }
```

Synthesize a function f_1
that satisfies the specification.

Program Synthesis: Diverse Approaches

- On top of SAT/SMT solver (Reynolds et al. 2015,2016,2019)
- Using grammar for Syntax Guided Synthesis (SyGuS) (Alur et al. 2013)
- Enumeration based CEGIS synthesizers (Alur et al. 2013,2017, Udupa et al. 2013)
- Using syntactic templates (Solar-Lezama et al. 2005,2008)

Dependency Quantified Formula ($DQF(\mathbb{T})$)

- Given a quantified formula ϕ in theory $\mathbb{T}$ with universal ($\forall$) and existential ($\exists$) quantifiers.

$$\phi := \forall x_1, \ldots, x_n \, \exists^{H_1} y_1 \ldots \exists^{H_m} y_m \, \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

- Y variables have explicit dependencies. Each $H_i \subseteq \{x_1, \ldots, x_n\}$.
- A $DQF(\mathbb{T})$ formula is True, if there exists function a vector $\boldsymbol{g} : \langle g_1(H_1), \ldots, g_m(H_m) \rangle$ such that $\varphi(x_1, \ldots, x_n, g_1(H_1), \ldots, g_m(H_m))$ is a tautology.
- When $\mathbb{T}$=Boolean, DQF($\mathbb{T}$) formula is Dependency Quantified Boolean Formulas (DQBF).

DQBF Solving: Diverse Approaches

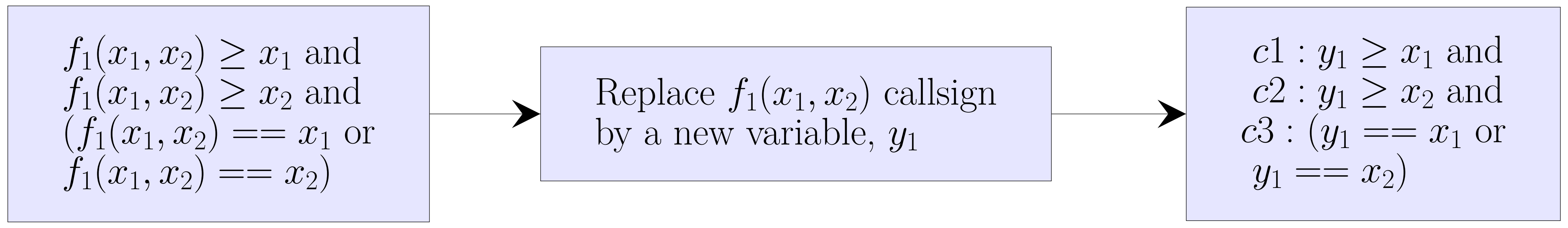
- Lifting CDCL for DQBF (Frohlich et al., 2012)
- Variable expansion based solvers (Bubeck et al. 2006, Gitina et al. 2013, 2015, Sic 2020)
- Clausal abstraction based (Tentrup et al., 2019)
- Definition extraction based (Reichl et al., 2021)

Our Contributions

- Established a connection between $\mathbb{T}$-constrained synthesis and DQF($\mathbb{T}$)— reduction of program synthesis to DQF($\mathbb{T}$).
- Reduction of DQF(BV) to DQBF — allows us to simply plug-in the state of the art DQBF solvers for BV-constrained synthesis.

Central Idea

- Introduce function callsign many new variables, and replace function callsign by the corresponding variable in the specification.
- Construct explicit dependencies for the introduced variables as per the function callsign.


$f_1(x_1, x_2) \geq x_1$ and
$f_1(x_1, x_2) \geq x_2$ and
$(f_1(x_1, x_2) == x_1$ or
$f_1(x_1, x_2) == x_2)$

Replace $f_1(x_1, x_2)$ callsign
by a new variable, y_1

$c1 : y_1 \geq x_1$ and
$c2 : y_1 \geq x_2$ and
$c3 : (y_1 == x_1$ or
$y_1 == x_2)$

Program Synthesis Instance:
Synthesize a function f_1
that satisfies the specification.

$DQF(\mathbb{T})$ instance:
$\forall \, x_1, x_2 \, \exists^{H_1:\{x_1,x_2\}} \, y_1$
$c1 \wedge c_2 \wedge c_3$

- Ask DQF($\mathbb{T}$) solver to synthesize function for y_1 in terms of x_1 and x_2.
- When $\mathbb{T}$ = Bit Vector: DQF($\mathbb{T}$) can be reduced to DQBF with the help of bitblasting.

Experimental Results

- $\mathbb{T}$ = Bit Vector. Benchmarks: 645 instances from SyGuS competition. Timeout: 900 seconds
- The number of instances solved by virtual best SyGuS and DQBF solver.

	SyGuS Solver	DQBF Solver
Total: 645	513	610

Tools Used in the Evaluation.

SyGuS Solvers	DQBF Solvers
CVC4,ESolver EUSolver,DryadSynth Stochpp	CADET, DCAQE Manthan, DepQBF DQBDD

Key Takeaways

- We can use state-of-the-art DQBF solvers for Bit Vector constrained synthesis.
- DQBF solvers were able to solve more than 100 instances that the SyGuS solves could not solve.

https://github.com/meelgroup/dequs