Deductive Synthesis

Woosuk Lee

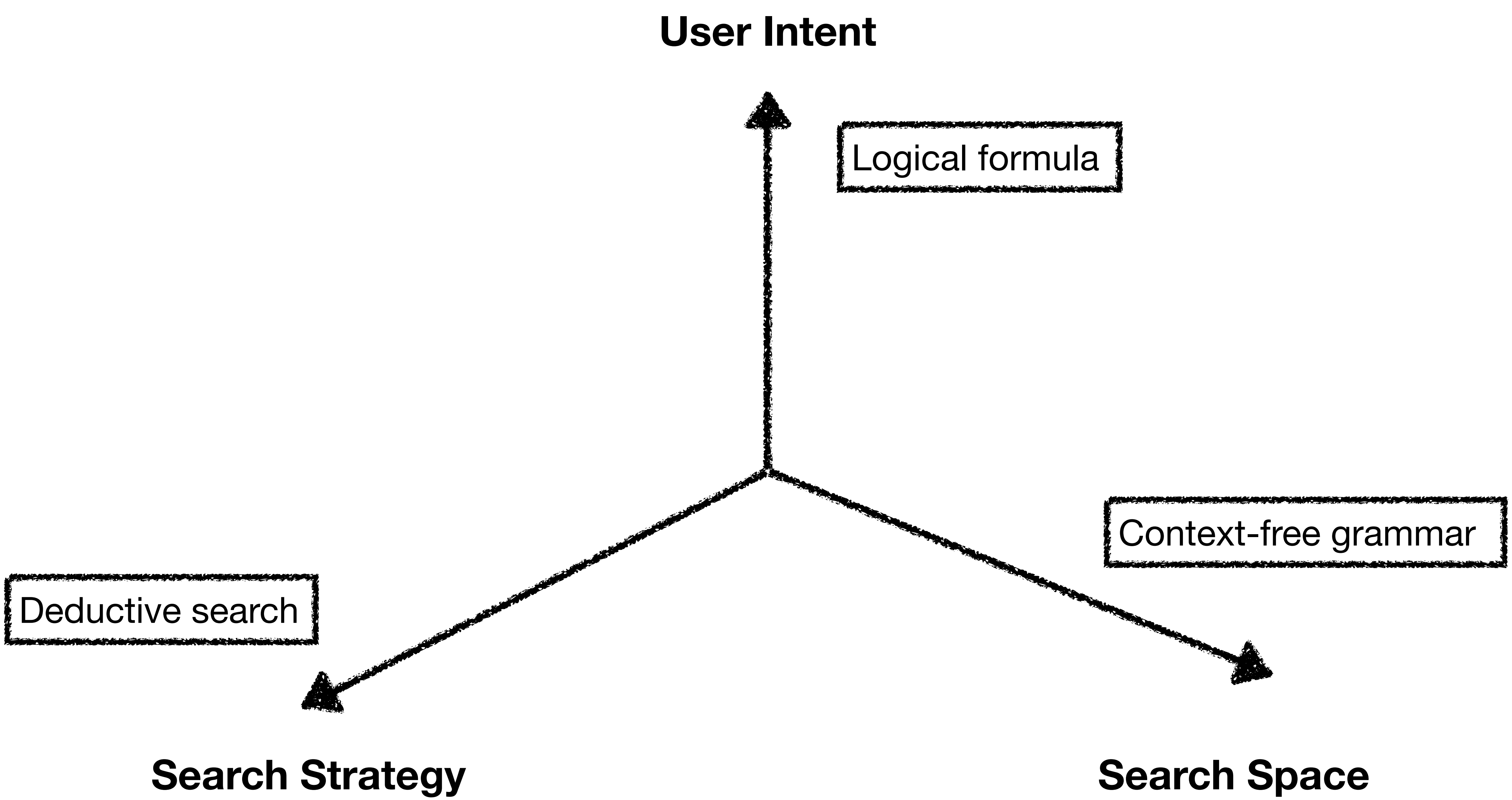
CSE9116 SPRING 2024

Hanyang University

HANYANG UNIVERSITY

Some part is from https://github.com/nadia-polikarpova/cse291-program-synthesis/blob/master/lectures/Lecture11.pdf

Dimension



User Intent
Logical formula
Context-free grammar
Deductive search
Search Strategy
Search Space

The Synthesis Problem

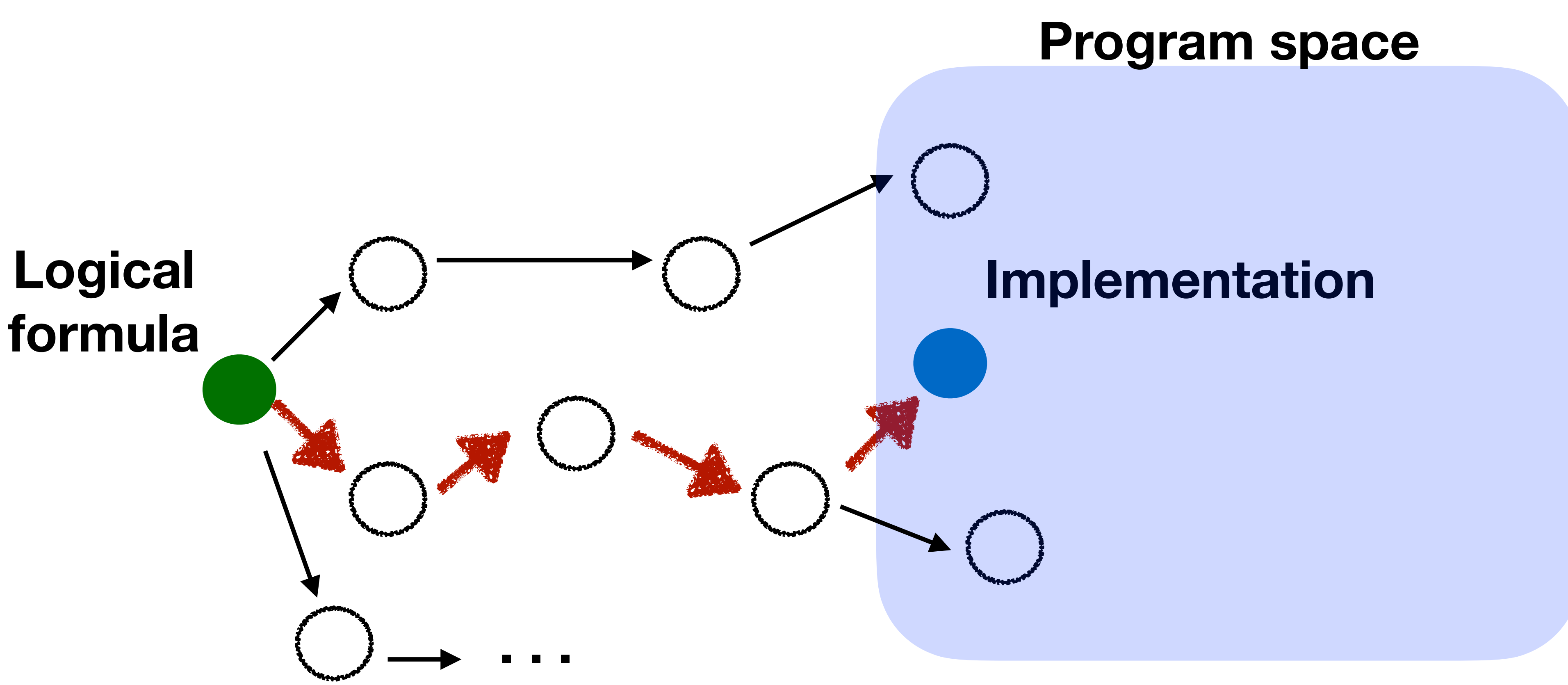
$$\boxed{\underbrace{\exists \mathtt{f}}_{\text{There exists a function } \mathtt{f}}.\ \underbrace{\forall \mathbf{x}.\ \mathtt{P}(\mathtt{f}, \mathbf{x})}_{\text{such that for all } \mathbf{x}\text{, property } \mathtt{P} \text{ holds}}}$$

- Enumerative approach
    - Enumerate all possible f until a desired one is found.
- **Deductive approach**
    - Reaptedly apply pre-defined transformation rules into the given spec until a solution is found.

Comparison to Compilers

- Compiler optimization: apply transformations in a predefined order

- Deductive synthesis: search is needed to apply transformations in order to arrive at a desired implementation



Logical Specification ⇨ Program

- Waldinger and Manna 1979

- Example: synthesizing a function $lesall(x, l)$ that determines whether x is less than all elements in a list l with the following given spec:

$$lesall(x, l) := \text{compute } x < all(l)$$

where x : number, l : list of numbers

Z. Manna, R. Waldinger, Synthesis: Dreams => Programs

Transformation Rules

- Empty lists: for any predicate P

transformed to

$P(all(l)) \implies true$ if l is an empty list

- Conditional formation:

$$
\begin{aligned}
S\ where\ Q \implies\ & if\ (P)\ S\ where\ (P \wedge Q) \\
& else\ S\ where\ (\neg P \wedge Q)
\end{aligned}
$$

Transformation Rules

- Non-empty lists:

$$P(all(l)) \implies P(head(l)) \wedge P(all(tail(l)))$$

if l is a non-empty list.

- Recursive calls:

given $f(x) := compute\ P(x)\ where\ Q$,

$P(t) \implies f(t)$ if Q is satisfied and $f(t)$ terminates

Example

$lesall(x, l) :=$ compute $x < all(l)$
where x : number, l : list of numbers

$\longrightarrow$ (conditional formation)

$$lesall(x, l) := if\ (empty(l))\ compute\ x < all(l)\ where\ (empty(l) \wedge Q)$$
$$else\ compute\ x < all(l)\ where\ (\neg empty(l) \wedge Q)$$

(Q : x is a number and l is a list of numbers)

$\longrightarrow$ (non-empty lists)

$$lesall(x, l) := if\ (empty(l))\ compute\ x < all(l)\ where\ (empty(l) \wedge Q)$$
$$else\ x < head(l) \wedge x < all(tail(l))\ where\ (\neg empty(l) \wedge Q)$$

Example

$$lesall(x,l) := if\ (empty(l))\ compute\ x < all(l)\ where\ (empty(l) \wedge Q)$$
$$else\ x < head(l) \wedge \underline{x < all(tail(l))}\ where\ (\neg empty(l) \wedge Q)$$

$\longrightarrow$ (recursive call)

$$lesall(x,l) := if\ (empty(l))\ compute\ x < all(l)\ where\ (empty(l) \wedge Q)$$
$$else\ x < head(l) \wedge lesall(x, tail(l))\ where\ (\neg empty(l) \wedge Q)$$

$\longrightarrow$ (empty list)

$$lesall(x,l) := if\ (empty(l))\ true\ where\ (empty(l) \wedge Q)$$
$$else\ x < head(l) \wedge lesall(x, tail(l))\ where\ (\neg empty(l) \wedge Q)$$

Solution!

Properties of Deductive Synthesis

- Correct by construction
  - Transformations are semantics-preserving
  - No need to verify a solution candidate
  - But some checks may be needed along the way
- Usually domain specific
  - due to pre-defined transformation rules

Modern Deductive Synthesis for SyGuS

- Two deductive synthesizers for CLIA (conditional linear integer arithmetic)
  - **DryadSynth**: Huang et al., Reconciling Enumerative and Deductive Program Synthesis, PLDI 2020
  - **CVC4**: Reynolds et al., Counterexample-Guided Quantifier Instantiation for Synthesis in SMT, CAV 2015

Modern Deductive Synthesis for SyGuS

- Two deductive synthesizers for CLIA (conditional linear integer arithmetic)
  - **DryadSynth**: Huang et al., Reconciling Enumerative and Deductive Program Synthesis, PLDI 2020
  - **CVC4**: Reynolds et al., Counterexample-Guided Quantifier Instantiation for Synthesis in SMT, CAV 2015

Deductive Synthesis for CLIA

- Example (max3): synthesize the function $f(x, y, z) : \text{int}$
  - Syntactic: $S \rightarrow 0 \mid 1 \mid x \mid y \mid z \mid \text{max2}(S, S)$
    where $\text{max2}(x, y) \triangleq \text{ite}(x \geq y, x, y)$
  - Semantic:
    $f(x, y, z) \geq x \land f(x, y, z) \geq y \land f(x, y, z) \land \geq z$

    $\land\ (f(x, y, z) = x \lor f(x, y, z) = y \lor f(x, y, z) = z)$
  - Solution: $f(x, y, z) = \text{max2}(\text{max2}(x, y), z)$

Transformation Rules

GeMax

$$f(\boldsymbol{e}) \geq e_1 \wedge f(\boldsymbol{e}) \geq e_2 \quad \Longrightarrow \quad f(\boldsymbol{e}) \geq \mathtt{ite}(e_1 \geq e_2, e_1, e_2)$$

LeMin

$$f(\boldsymbol{e}) \leq e_1 \wedge f(\boldsymbol{e}) \leq e_2 \quad \Longrightarrow \quad f(\boldsymbol{e}) \leq \mathtt{ite}(e_1 \geq e_2, e_2, e_1)$$

GeMin

$$f(\boldsymbol{e}) \geq e_1 \vee f(\boldsymbol{e}) \geq e_2 \quad \Longrightarrow \quad f(\boldsymbol{e}) \geq \mathtt{ite}(e_1 \geq e_2, e_2, e_1)$$

LeMax

$$f(\boldsymbol{e}) \leq e_1 \vee f(\boldsymbol{e}) \leq e_2 \quad \Longrightarrow \quad f(\boldsymbol{e}) \leq \mathtt{ite}(e_1 \geq e_2, e_1, e_2)$$

Eq

$$f(\boldsymbol{e}) \geq e_1 \wedge f(\boldsymbol{e}) \leq e_2 \quad \Longrightarrow \quad f(\boldsymbol{e}) = e_1$$

if $\mathcal{T} \models e_1 = e_2$

NotEq

$$f(\boldsymbol{e}) \geq e_1 \vee f(\boldsymbol{e}) \leq e_2 \quad \Longrightarrow \quad f(\boldsymbol{e}) \neq e_1 - 1$$

if $\mathcal{T} \models e_1 = e_2 + 2$

CNF

$$(\Phi \vee \Psi_1) \wedge (\Phi \vee \Psi_2) \quad \Longrightarrow \quad \Phi \vee (\Psi_1 \wedge \Psi_2)$$

if f does not occur in Ψ_1 or Ψ_2

Transformation Rules

InTEq

$$f(\boldsymbol{y}) = e \land \Psi \implies f(\boldsymbol{y}) = e \land \Psi[\lambda \boldsymbol{y}.e/f]$$

IntNeq

$$f(\boldsymbol{y}) \neq e \lor \Psi \implies f(\boldsymbol{y}) \neq e \lor \Psi[\lambda \boldsymbol{y}.e/f]$$

BoolPos

$$(f(\boldsymbol{y}) \lor \Phi) \land \Psi \implies \Psi[\lambda \boldsymbol{y}.((\neg\Phi) \lor f(\boldsymbol{y}))/f]$$

if f does not occur in Φ

BoolNeg

$$(\neg f(\boldsymbol{y}) \lor \Phi) \land \Psi \implies \Psi[\lambda \boldsymbol{y}.(\Phi \land f(\boldsymbol{y}))/f]$$

if f does not occur in Φ

RemoveVar

$$\Psi \implies \Psi[0/y_i] \quad \text{if } \mathcal{T} \models \Phi \leftrightarrow \Phi[y'_i/y_i]$$

RemoveArg

$$(f, \Phi, \mathcal{G}) \implies (g, \Phi[g(\boldsymbol{e}, \boldsymbol{e'})/f(\boldsymbol{e}, C, \boldsymbol{e'})], \mathcal{G})$$

if the i-th arg of f is always constant C

Match

$$f(\mathbf{y}) = e \implies f(\mathbf{y}) = e'$$

e **and** e' **are semantically equivalent**

Synthesis Process

$$f(x,y,z) \geq x \;\wedge\; f(x,y,z) \geq y \;\wedge\; f(x,y,z) \geq z \;\wedge$$
$$\big(f(x,y,z) = x \vee f(x,y,z) = y \vee f(x,y,z) = z\big) \xRightarrow{\text{CNF}}$$
$$f(x,y,z) \geq x \;\wedge\; f(x,y,z) \geq y \;\wedge\; f(x,y,z) \geq z$$
$$\wedge\; \big(f(x,y,z) \geq x \vee f(x,y,z) \geq y \vee f(x,y,z) \geq z\big)$$
$$\wedge\; \big(f(x,y,z) \leq x \vee f(x,y,z) \leq y \vee f(x,y,z) \leq z\big)$$
$$\wedge\; \ldots \xRightarrow{\textsc{GeMax},\textsc{LeMax},\ldots}$$
$$f(x,y,z) \geq \mathtt{ite}(\mathtt{ite}(x \geq y, x, y) \geq z, \mathtt{ite}(x \geq y, x, y), z)$$
$$\wedge\; f(x,y,z) \leq \mathtt{ite}(\mathtt{ite}(x \geq y, x, y) \geq z, \mathtt{ite}(x \geq y, x, y), z)$$
$$\wedge\; \ldots \xRightarrow{\textsc{Eq},\textsc{IntEq}}$$
$$f(x,y,z) = \mathtt{ite}(\mathtt{ite}(x \geq y, x, y) \geq z, \mathtt{ite}(x \geq y, x, y), z)$$
$$\xRightarrow{\textsc{Match}} f(x,y,z) = \mathit{max2}(\mathit{max2}(x,y),z)$$

Other Ideas in the Paper

- Divide-and-Conquer: synthesize a partial solution satisfying only a sub-part of the given spec and use it to get the entire solution
    - e.g., synthesize max2 function first and use it to synthesize max3
- Synthesizing ite expressions by finding coefficients

Synthesizing ite expressions by finding coefficients

- Example (max2): finding the function $f(x, y)$ s.t.
  $f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y)$

- Express a solution as
  $\underline{\mathrm{ite}(c_1 x + c_2 y + d_1 \geq 0, c_3 x + c_4 y + d_2, c_3 x + c_4 y + d_3)}$

  Let's call it g

- Invoke an SMT solver with
  $\forall x, y \,.\; g \geq x \land g \geq y \land (g = x \lor g = y)$ and obtains
  $\{c_1 = 1, c_2 = -1, d_1 = 0, c_3 = 1, c_4 = 0, d_2 = 0, c_3 = 0, c_4 = 1, d_3 = 0\}$

- If a solution cannot be found, increase the size of a potential solution and try again

Modern Deductive Synthesis for SyGuS

- Two deductive synthesizers for CLIA (conditional linear integer arithmetic)
  - **DryadSynth**: Huang et al., Reconciling Enumerative and Deductive Program Synthesis, PLDI 2020
  - **CVC4**: Reynolds et al., Counterexample-Guided Quantifier Instantiation for Synthesis in SMT, CAV 2015

Refutation-based Synthesis

- Given

$$\boxed{\underbrace{\exists \texttt{f}}_{\text{There exists a function } \texttt{f}}.\underbrace{\forall \mathbf{x}.\texttt{P}(\texttt{f},\mathbf{x})}_{\text{such that for all } \mathbf{x}\text{, property } \texttt{P} \text{ holds}}}$$

- Negate it

$$\neg\exists f.\ \forall x.\ P(f,x) \Longleftrightarrow \underline{\forall f.\ \exists x.\ \neg P(f,x)}$$

- If an SMT solver determines it unsatisfiable,

- We know $\exists f.\ \forall x.\ P(f,x)$ is satisfiable, i.e., there exists a solution.

Max2 Example

- Finding the function $f(x, y)$

- Syntactic: $S \rightarrow 0 \mid 1 \mid x \mid y \mid \mathrm{ite}(B, S, S)$
  $$B \rightarrow S \leq S \mid S \geq S$$

- Semantic:
  $$f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$$

Max2 Example

- Finding the function $f(x, y)$
- Syntactic: $S \rightarrow 0 \mid 1 \mid x \mid y \mid \mathrm{ite}(B, S, S)$

$$B \rightarrow S \leq S \mid S \geq S$$

- Semantic:

$$f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$$

Single invocation property: all occurrences of f are of a particular form, e.g., $f(x, y)$

Refuting the Spec

$$\neg \exists f \,.\, \forall x, y \,.\, f(x,y) \geq x \wedge f(x,y) \geq y \wedge (f(x,y) = x \vee f(x,y) = y)$$

Push negation inwards

$$\forall f \,.\, \exists x, y \,.\, f(x,y) < x \vee f(x,y) < y \vee (f(x,y) \neq x \wedge f(x,y) \neq y)$$

Replace $f(x, y)$ with a new variable g

$$\forall g \,.\, \exists x, y \,.\, g < x \vee g < y \vee (g \neq x \wedge g \neq y)$$

Introduce new vars a, b for the existential quantifier
(a.k.a skolemization)

$$\forall g \,.\, g < a \vee g < b \vee (g \neq a \wedge g \neq b)$$

(Universal) Quantifier Instantiation

- Find a set S such that

$$\forall g \in S\,.\; g < a \vee g < b \vee (g \neq a \wedge g \neq b)$$

is unsatisfiable.

- $S = \{a, b\}$ is such a set.

$g \mapsto a$

$$(a < a \vee a < b \vee (a \neq a \wedge a \neq b)) \wedge$$
$$(b < a \vee b < b \vee (b \neq a \wedge b \neq b))$$
$$\iff \; a < b \wedge b < a$$

$g \mapsto b$

Solution Construction

- A solution can be constructed from the set S

$$\mathrm{ite}(\underbrace{a < a \lor a < b \lor (a \neq a \land a \neq b)}_{g \mapsto a}, a, b)$$

$\Downarrow$ Simplify

$$\mathrm{ite}(a < b, a, b)$$

$\Downarrow$ Negate the condition and replace a, b with x, y

$$f(x, y) := \mathrm{ite}(x \geq y, x, y)$$

Why is That a Solution?

- Given: $\exists f \,.\, \forall \mathbf{x} \,.\, P(f(\mathbf{x}), \mathbf{x})$ where
  - P satisfies the single invocation property
  - $\mathbf{x}$: input variables
- Found: $S = \{t_1, t_2, \cdots, t_n\}$
  - i.e., $\neg P(t_1, \mathbf{k}) \wedge \neg P(t_2, \mathbf{k}) \wedge \cdots \wedge \neg P(t_n, \mathbf{k})$ is always false
  - $\mathbf{k}$: skolemized variables

Why is That a Solution?

- Theorem. The following is the solution.

$$
\begin{aligned}
f(\mathbf{x}) := \ & \mathrm{ite}(P(t_1, \mathbf{k}), t_1, \\
& (\mathrm{ite}(P(t_2, \mathbf{k}), t_2, \\
& \ldots \\
& (\ldots(\mathrm{ite}(P(t_{n-1}, \mathbf{k}), t_{n-1}, \\
& \qquad\qquad\qquad\qquad t_n)\ldots)[\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

Replace all occurrences of $\mathbf{k}$ with $\mathbf{x}$

Why is That a Solution?

- Theorem. The following is the solution.

$$
\begin{aligned}
f(\mathbf{x}) := \;& \underline{\mathrm{ite}(P(t_1, \mathbf{k}), t_1,} \\
& (\mathrm{ite}(P(t_2, \mathbf{k}), t_2, \\
& \dots \\
& (\dots(\mathrm{ite}(P(t_{n-1}, \mathbf{k}), t_{n-1}, \\
& \qquad\qquad\qquad\qquad t_n)\dots)[\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

If P holds for t_1, return t_1:
For a given input $\mathbf{x}$, $f(\mathbf{x}) = t_1$ makes the spec satisfied. Thus, f outputs t_1.

Why is That a Solution?

- Theorem. The following is the solution.

$$
\begin{aligned}
f(\mathbf{x}) := \ & \text{ite}(P(t_1, \mathbf{k}), t_1, \\
& \underline{(\text{ite}(P(t_2, \mathbf{k}), t_2,} \\
& \dots \\
& (\dots(\text{ite}(P(t_{n-1}, \mathbf{k}), t_{n-1}, \\
& \qquad\qquad\qquad\qquad t_n)\dots)[\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

If P holds for t_2, return t_2

Why is That a Solution?

- Theorem. The following is the solution.

$$
\begin{aligned}
f(\mathbf{x}) := {} & \mathrm{ite}(P(t_1, \mathbf{k}), t_1, \\
& (\mathrm{ite}(P(t_2, \mathbf{k}), t_2, \\
& \ldots \\
& (\ldots \underline{(\mathrm{ite}(P(t_{n-1}, \mathbf{k}), t_{n-1},} \\
& \qquad\qquad\qquad\qquad t_n)\ldots)[\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

If P holds for t_{n-1}, return t_{n-1}

Why is That a Solution?

- Theorem. The following is the solution.

$$
\begin{aligned}
f(\mathbf{x}) := \ & \text{ite}(P(t_1, \mathbf{k}), t_1, \\
& (\text{ite}(P(t_2, \mathbf{k}), t_2, \\
& \dots \\
& (\dots(\text{ite}(P(t_{n-1}, \mathbf{k}), t_{n-1}, \\
& \qquad\qquad\qquad\qquad\qquad \underline{t_n})\dots)[\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

$\neg P(t_1, \mathbf{k}) \wedge \neg P(t_2, \mathbf{k}) \wedge \cdots \wedge \neg P(t_n, \mathbf{k})$ is false

$\neg P(t_1, \mathbf{k}) \wedge \cdots \wedge \neg P(t_{n-1}, \mathbf{k})$ **is true**

$\therefore P(t_n, \mathbf{k})$

If P holds for t_n, return t_n

Other Details

- How to find such a set S?
  - Counterexample-guided quantifier instantiation (CEGQI)
- Limitation: single invocation property
  - However, if all occurrences of f in a spec are of a form either $f(x, y)$ or $f(y, x)$ and f satisfies the commutativity, such a spec can be handled.
- How to construct a solution complying with a syntactic restriction?
  - First, find a solution while ignoring the syntactic restriction and then transform it into equivalent one that meets the restriction.

Summary

- Deductive synthesis = applying transformations
- Efficient for specific domains (e.g., for a specific theory like CLIA or particular form of logical spec)
- Synergistically combined with inductive enumerative search nowadays