Complete Functional Synthesis

Definition (Synthesis Procedure)
A synthesis procedure takes as input formula $F(x, a)$ and outputs a witness term $Ψ$ such that $F(Ψ,a)$ is a valid result of quantifier elimination.

That is, it outputs

1. a precondition formula $pre(a)$
2. list of terms $Ψ$

such that the following holds:

$$∃x. F(x, a) \iff pre(a) \iff F[x := Ψ]$$

Note: $pre(a)$ is the “best” possible precondition under which a solution exists – domain of the relation.
choose((x, y) ⇒ 5 * x + 7 * y == a && x ≤ y)

Corresponding quantifier elimination problem:

∃ x ∃ y . 5x + 7y = a ∧ x ≤ y

Use extended Euclid’s algorithm to find particular solution to 5x + 7y = a:

(5, 7 are mutually prime, else we get divisibility pre.)

Express general solution of equations for x, y using a new variable z:

x = -7z + 3a
y = 5z - 2a

Rewrite inequations x ≤ y in terms of z:

5a ≤ 12z

Obtain synthesized program:

val z = ceil(5*a/12)
val x = -7*z + 3*a
val y = 5*z + -2*a

For a = 31:

z = ceil(5*31/12) = 13
x = -7*13 + 3*31 = 2
y = 5*13 - 2*31 = 3
Compilation in Comfusy

```python
def secondsToTime(totalSeconds: Int) : (Int, Int, Int) =
    choose((h: Int, m: Int, s: Int) ⇒ (h * 3600 + m * 60 + s == totalSeconds && h ≥ 0 && m ≥ 0 && m < 60 && s ≥ 0 && s < 60 ))
```

```python
def secondsToTime(totalSeconds: Int) : (Int, Int, Int) =
    val t1 =
    val t2 =
    val t3 =
    val t4 =
    (t1, t3, t4)
```

Mikaël Mayer  Ruzica Piskac  Philippe Suter,  PLDI’10
Synthesis Procedure on Example

- process every equality: take an equality $E_i$, compute a parametric description of the solution set and insert those values in the rest of formula

$$
\begin{pmatrix}
  h \\
  m \\
  s
\end{pmatrix} = \lambda
\begin{pmatrix}
  1 \\
  0 \\
  -3600
\end{pmatrix} + \mu
\begin{pmatrix}
  0 \\
  1 \\
  -60
\end{pmatrix} +
\begin{pmatrix}
  0 \\
  0 \\
  totalSeconds
\end{pmatrix}
\big| \lambda, \mu \in \mathbb{Z}
$$

CODE:
<Further code will come here>

```java
val h = lambda
val m = mu
val s = (-3600) * lambda + (-60) * mu + totalSeconds
```
Synthesis Procedure by Example

• process every equality: take an equality $E_i$, compute a parametric description of the solution set and insert those values in the rest of formula

$$\begin{pmatrix}h \\ m \\ s\end{pmatrix} = \lambda \begin{pmatrix}1 \\ 0 \\ -3600\end{pmatrix} + \mu \begin{pmatrix}0 \\ 1 \\ -60\end{pmatrix} + \begin{pmatrix}0 \\ 0 \\ totalSeconds\end{pmatrix} \mid \lambda, \mu \in \mathbb{Z}$$

Formula (remain specification):

$0 \leq \lambda, 0 \leq \mu, \mu \leq 59, 0 \leq totalSeconds - 3600\lambda - 60\mu, totalSeconds - 3600\lambda - 60\mu \leq 59$
Processing Inequalities

process output variables one by one

\[ 0 \leq \lambda, \ 0 \leq \mu, \ \mu \leq 59, \ 0 \leq \text{totalSeconds} - 3600\lambda - 60\mu, \]
\[ \text{totalSeconds} - 3600\lambda - 60\mu \leq 59 \]

expressing constraints as bounds on \( \mu \)

\[ 0 \leq \lambda, \ 0 \leq \mu, \ \mu \leq 59, \ \mu \leq \lfloor (\text{totalSeconds} - 3600\lambda)/60 \rfloor, \]
\[ \lceil (\text{totalSeconds} - 3600\lambda - 59)/60 \rceil \leq \mu \]

Code:

```r
val mu = min(59, (totalSeconds - 3600* lambda) div 60)
```
Fourier-Motzkin-Style Elimination

0 ≤ λ, 0 ≤ μ, μ ≤ [(totalSeconds – 3600λ)/60],
[(totalSeconds – 3600λ – 59)/60] ≤ μ

combine each lower and upper bound

0 ≤ λ, 0 ≤ 59, 0 ≤ [(totalSeconds – 3600λ)/60],
[(totalSeconds – 3600λ – 59)/60] ≤ [(totalSeconds – 3600λ)/60],
[(totalSeconds – 3600λ – 59)/60] ≤ 59

basic simplifications

0 ≤ λ, 60λ ≤ [totalSeconds /60],
[(totalSeconds –59)/60] – 59 ≤ 60λ

Code:

val lambda = totalSeconds div 3600

Preconditions: 0 ≤ totalSeconds
def secondsToTime(totalSeconds: Int) : (Int, Int, Int) =

choose((h: Int, m: Int, s: Int) ⇒ (h * 3600 + m * 60 + s == totalSeconds

&& h ≥ 0

&& m ≥ 0 && m < 60

&& s ≥ 0 && s < 60 ))

def secondsToTime(totalSeconds: Int) : (Int, Int, Int) =

val t1 = totalSeconds div 3600
val t2 = totalSeconds -3600 * t1
val t3 = t2 div 60
val t4 = totalSeconds - 3600 * t1 - 60 * t3
(t1, t3, t4)
Handling of Inequalities

• Solve for one by one variable:
  – separate inequalities depending on polarity of $x$:
    • $A_i \leq \alpha_i x$
    • $\beta_j x \leq B_j$
  – define values $a = \max_i [A_i/\alpha_i]$ and $b = \min_j [B_j/\beta_j]$
• if $b$ is defined, return $x = b$ else return $x = a$
• further continue with the conjunction of all formulas $[A_i/\alpha_i] \leq [B_j/\beta_j]$
\[2y - b \leq 3x + a \land 2x - a \leq 4y + b\]

\[2y - b - a \leq 3x \land 2x \leq 4y + a + b\]
\[4y - 2b - 2a \leq 6x \leq 12y + 3a + 3b\]
\[(4y - 2b - 2a) / 6 \leq x \leq (12y + 3a + 3b) / 6\]

two extra variables:
\[(4y - 2b - 2a) / 6 \leq l \land 12y + 3a + 3b = 6 \ast l + k\]
\[\land 0 \leq k \leq 5\]

\[4y - 2b - 2a \leq 6 \ast l \land 12y + 3a + 3b = 6 \ast l + k\]
\[\land 0 \leq k \leq 5\]

**pre:** \[6 \mid 3a + 3b - k + 12y\]

\[4y - 2b - 2a \leq 12y + 3a + 3b - k\]
\[-5b - 5a + k \leq 8y\]

\[\Rightarrow y = [(k - 5a - 5b)/8]\]
val \( (x_1, y_1) = \textbf{choose}(x: \text{Int}, y: \text{Int} \Rightarrow \) \\
\quad 2*y - b =< 3*x + a \&\& 2*x - a =< 4*y + b) \\
\text{val } k\text{Found} = \text{false} \\
\textbf{for } k = 0 \text{ to } 5 \text{ do } \{ \\
\quad \text{val } v_1 = 3 \times a + 3 \times b - k \\
\quad \text{if } (v_1 \mod 6 == 0) \{ \\
\quad \quad \text{val } \alpha = ((k - 5 \times a - 5 \times b)/8).\text{ceiling} \\
\quad \quad \text{val } l = (v_1 / 6) + 2 \times \alpha \\
\quad \quad \text{val } y = \alpha \\
\quad \quad \text{val } k\text{Found} = \text{true} \\
\quad \quad \text{break } \} \} \\
\text{if } (k\text{Found}) \\
\quad \text{val } x = ((4 \times y + a + b)/2).\text{floor} \\
\textbf{else throw} \text{ new Exception("No solution exists")}
NP-Hard Constructs

- Divisibility combined with inequalities:
  - corresponding to big disjunction in q.e. , we will generate a for loop with constant bounds (could be expanded if we wish)

- Disjunctions
  - Synthesis of a formula computes program and exact precondition of when output exists
  - Given disjunctive normal form, use preconditions to generate if-then-else expressions (try one by one)
Synthesis for Disjunctions

\[
\begin{align*}
[\vec{x}, D_1 \lor \ldots \lor D_n] &= \text{let } (\text{pre}_1, \vec{\Psi}_1) = [\vec{x}, D_1] \\
&\quad \ldots \\
&\quad (\text{pre}_n, \vec{\Psi}_n) = [\vec{x}, D_n] \\
&\text{in} \\
&\bigg( \bigg( \bigvee_{i=1}^{n} \text{pre}_i, \begin{cases} 
& \text{if } (\text{pre}_1) \vec{\Psi}_1 \\
& \text{else if } (\text{pre}_2) \vec{\Psi}_2 \\
& \ldots \\
& \text{else if } (\text{pre}_n) \vec{\Psi}_n \\
& \text{else assert(false)}
\end{cases} \bigg) \bigg) 
\end{align*}
\]
choose $x$ such that $F(x,a) \rightarrow x = t(a)$

Result $t(a)$ is expressed in terms of $+$, $C^*$, $/C$, $\%C$, $\text{if}$

Need arithmetic for solving equations

Need conditionals for

- disjunctions in input formula
- divisibility and inequalities (find a witness meeting bounds and divisibility by constants)

$t(a) = \text{if } P_1(a) \ t_1(a) \ \text{elseif} \ldots \ \text{elseif} \ P_n(a) \ t_n(a) \ \text{else error}(	ext{“No solution exists for input”},a)$
Methodology QE $\rightarrow$ Synthesis

- Each quantifier elimination ‘trick’ we found corresponds to a synthesis trick
- Find the corresponding terms
- Key techniques:
  - one point rule immediately gives a term
  - change variables, using a computable function
  - strengthen formula while preserving realizability
  - recursively eliminate variables one-by one
- Example use
  - transform formula into disjunction
  - strengthen each disjunct using equality
  - apply one-point rule
def secondsToTime(totalSeconds: Int) : (Int, Int, Int) = 
  choose((h: Int, m: Int, s: Int) ⇒ ( 
    h * 3600 + m * 60 + s == totalSeconds 
    && h ≥ 0 && h < 24
    && m ≥ 0 && m < 60
    && s ≥ 0 && s < 60
  ))

Warning: Synthesis predicate is not satisfiable for variable assignment: 
  totalSeconds = 86400
def secondsToTime(totalSeconds: Int) : (Int, Int, Int) =
    choose((h: Int, m: Int, s: Int) ⇒ (h * 3600 + m * 60 + s == totalSeconds
    && h ≥ 0
    && m ≥ 0 && m < 60
    && s ≥ 0 && s ≤ 60))

Warning: Synthesis predicate has multiple solutions for variable assignment:
    totalSeconds = 60
Solution 1: h = 0, m = 0, s = 60
Solution 2: h = 0, m = 1, s = 0