Reductions for Synthesis Procedures

Swen Jacobs

TU Graz, Austria

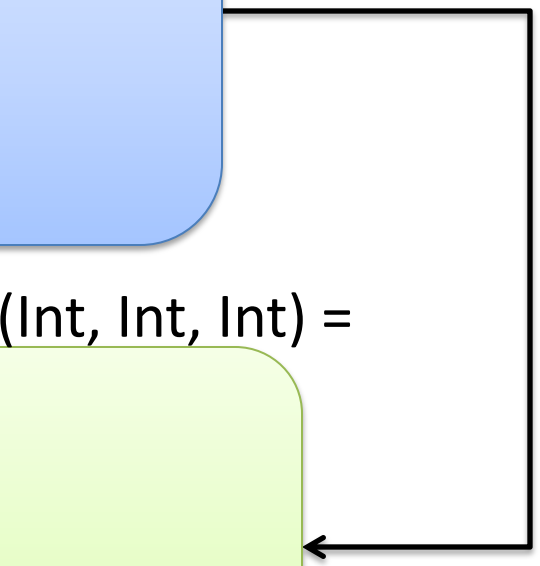
Viktor Kuncak, Philippe Suter

EPFL, Switzerland

(based on the paper of the same name, to be presented at VMCAI 2013)

Compiling Specifications

```
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 / 3600
  val t2 = totalSeconds + ((-3600) * t1)
  val t3 = min(t2 / 60, 59)
  val t4 = totalSeconds + ((-3600) * t1) + (-60 * t3)
  (t1, t3, t4)
```

Transforming a relation into a function.

Decision & Synthesis Procedures

For a well-defined class of formulas:

Decision procedure (model-generating)

a theorem prover that always succeeds

- **Input**: a formula
- **Output**: a model of the formula

$5a + 7x = 31$

$\downarrow$

$a \mapsto 2$

$x \mapsto 3$

Synthesis procedure

a synthesizer that always succeeds

- **Input**: a formula, with *input* and *output* variables
- **Output**: a program to compute *output* values from *input* values

Inputs: { a } outputs: { x }

$5a + 7x = 31$

$\downarrow$

$x \mapsto (31 - 5a) / 7$

Synthesis Procedures

- So far, “monolithic” approach; procedures work as black boxes.
- Synthesis for sets is a one-off encoding into integer linear arithmetic.
- Original motivation for current work: general technique to combine synthesis procedures for different theories (*à la* Nelson-Oppen).

Transforming Relations

$$[\![\, \overline{a} \,\langle\, \varphi_1 \,\rangle\, \overline{x} \,]\!] \vdash [\![\, \overline{a} \,\langle\, \varphi_2 \,\rangle\, \overline{x} \,]\!]$$

Input variables

Synthesis predicate

Output variables

$$\forall\, \overline{a}, \overline{x} : \varphi_2 \Rightarrow \varphi_1$$

Refinement

$$\forall\, \overline{a} : (\exists\, \overline{x} : \varphi_1) \Rightarrow (\exists\, \overline{x} : \varphi_2)$$

Domain preservation

Transforming Relations

$$[\![\, \overline{a} \,\langle\, \varphi \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle$$

Input variables

Output variables

Precondition

Program terms

Represents the relation: $P \Rightarrow (\overline{x} = \overline{T})$

$$\forall\, \overline{a} : P \Rightarrow \varphi[\overline{x} \mapsto \overline{T}]$$

Refinement

$$\forall\, \overline{a} : (\exists\, \overline{x} : \varphi) \Rightarrow P$$

Domain preservation

Transforming Relations

Equivalence

$$\frac{[\![\, \overline{a} \langle\, \varphi_1 \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle \qquad \varphi_1 \Leftrightarrow \varphi_2}{[\![\, \overline{a} \langle\, \varphi_2 \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle}$$

Case-Split

$$\frac{[\![\, \overline{a} \langle\, \varphi_1 \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P_1 \mid \overline{T}_1 \,\rangle \qquad [\![\, \overline{a} \langle\, \varphi_2 \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P_2 \mid \overline{T}_2 \,\rangle}{[\![\, \overline{a} \langle\, \varphi_1 \vee \varphi_2 \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P_1 \vee P_2 \mid \textbf{if}(P_1)\ \overline{T}_1\ \textbf{else}\ \overline{T}_2 \,\rangle}$$

One-Point

$$\frac{[\![\, \overline{a} \langle\, \varphi[x_0 \mapsto t] \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle \qquad x_0 \notin \mathrm{vars}(t)}{[\![\, \overline{a} \langle\, x_0 = t \wedge \varphi \,\rangle\, x_0; \overline{x} \,]\!] \vdash \langle\, P \mid \textbf{let}\ \overline{x} := \overline{T}\ \textbf{in}\ t; \overline{x} \,\rangle}$$

Validating Inference Rules

$$\frac{[\![\, \overline{a}\, \langle\, \varphi_1 \,\rangle\, \overline{x}\,]\!] \vdash \langle\, P_1 \mid \overline{T}_1 \,\rangle \qquad [\![\, \overline{a}\, \langle\, \varphi_2 \,\rangle\, \overline{x}\,]\!] \vdash \langle\, P_2 \mid \overline{T}_2 \,\rangle}{[\![\, \overline{a}\, \langle\, \varphi_1 \vee \varphi_2 \,\rangle\, \overline{x}\,]\!] \vdash \langle\, P_1 \vee P_2 \mid \textbf{if}(P_1)\ \overline{T}_1\ \textbf{else}\ \overline{T}_2 \,\rangle}$$

Refinement $\forall\, \overline{a} : P \Rightarrow \varphi[\overline{x} \mapsto \overline{T}]$

$$P_1 \vee P_2 \Rightarrow (\varphi_1 \vee \varphi_2)[\overline{x} \mapsto (\textbf{if}(P_1)\ \overline{T}_1\ \textbf{else}\ \overline{T}_2)]$$

Assuming P_1

$$(\varphi_1 \vee \varphi_2)[\overline{x} \mapsto \overline{T}_1] \longrightarrow \varphi_1[\overline{x} \mapsto \overline{T}_1] \vee \varphi_2[\overline{x} \mapsto \overline{T}_1]$$

Domain preservation $\forall\, \overline{a} : (\exists\, \overline{x} : \varphi) \Rightarrow P$

$$(\exists\, \overline{x} : \varphi_1 \vee \varphi_2) \Rightarrow P_1 \vee P_2$$

$$(\exists\, \overline{x} : \varphi_1) \vee (\exists\, \overline{x} : \varphi_2) \Rightarrow P_1 \vee P_2$$

Advantages of Rule-based Synthesis

- Simplifies correctness arguments.
- Simple to reason about interleavings of synthesis steps for different theories.
- Partial synthesis:
  - Synthesis predicates partially in unsupported logics.
  - Program size / efficiency tradeoffs.

Relation to Quantifier Elimination

- A problem of the form:

$$[\![\, \overline{a} \,\langle\, \varphi \,\rangle\, \overline{x} \,]\!]$$

- Corresponds to constructively solving the **quantifier elimination** problem:

$$\exists\, \overline{x} : \varphi(\, \overline{a}, \overline{x}\,)$$

- In the solution, P corresponds to the result of Q.E. and $\overline{T}$ are *witness terms.*

$$\langle\, P \mid \overline{T} \,\rangle$$

Synthesis Procedures: Applicability

- In principle, any model-generating decision procedure can be adapted; at the limit, the generated program *is* the decision procedure.

- Synthesis procedures derived from Q.E.:
  - Integer linear (Presburger) arithmetic
  - Sets with cardinalities
  - Algebraic data types (lists, trees, etc.)

Synthesis for Linear Integer Arithmetic

$$\frac{[\![\, a \,\langle\, 7t \le a \wedge 5a \le 12t \,\rangle\, t \,]\!] \vdash \langle\, \lceil 5a/12 \rceil \le \lfloor 3a/7 \rfloor \mid \lceil 5a/12 \rceil \,\rangle}{[\![\, a \,\langle\, 5x + 7y = a \wedge 0 \le x \wedge x \le y \,\rangle\, x,y \,]\!] \vdash \langle\, \lceil 5a/12 \rceil \le \lfloor 3a/7 \rfloor \mid \textbf{let } t = \lceil 5a/12 \rceil \textbf{ in } (-7t+3a,\ 5t-2a) \,\rangle}$$

$5x + 7y = a$

One-dimensional solution space.

$x = -7t + 3a$

$y = 5t - 2a$

is a solution for any t.

$7t \le a \wedge 5a \le 12t$

t is bound on both sides, and admits a solution whenever

$\lceil 5a/12 \rceil \le \lfloor 3a/7 \rfloor$

Synthesis for Term Algebras

List ::= Nil | Cons(h : Int, t : List)

Unification-1
$$\frac{[\![\, \overline{a} \,\langle\, t_1 = t_3 \wedge t_2 = t_4 \wedge \varphi \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle}{[\![\, \overline{a} \,\langle\, \mathrm{Cons}(t_1, t_2) = \mathrm{Cons}(t_3, t_4) \wedge \varphi \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle}$$

Unification-2
$$[\![\, \overline{a} \,\langle\, \mathrm{Cons}(t_1, t_2) = \mathrm{Nil} \wedge \varphi \,\rangle\, \overline{x} \,]\!] \vdash \langle\, \textbf{false} \mid \_ \,\rangle$$

Unification-3 ...

Dual
$$\frac{[\![\, \overline{a} \,\langle\, t_1 = a.h \wedge t_2 = a.t \wedge \varphi \,\rangle\, \overline{x} \,]\!] \vdash \langle\, P \mid \overline{T} \,\rangle}{[\![\, \overline{a} \,\langle\, a = \mathrm{Cons}(t_1, t_2) \wedge \varphi \,\rangle\, \overline{x} \,]\!] \vdash \langle\, \mathrm{isCons}(a) \wedge P \mid \overline{T} \,\rangle}$$

Synthesis for Term Algebras

$[\![\ a\ \langle\ x \neq a\ \rangle\ x,y\ ]\!] \vdash$
$\langle$ **true** | (a+1, Nil) $\rangle$

$[\![\ b\ \langle\ b \neq y\ \rangle\ x,y\ ]\!] \vdash$
$\langle$ **true** | (0, Δ(b)) $\rangle$

$[\![\ a,b\ \langle\ \text{Cons}(x,b) \neq \text{Cons}(a,y)\ \rangle\ x,y\ ]\!] \vdash$
$\langle$ **true** | **if**(**true**) (a+1, Nil) **else** (0 , Δ(b)) $\rangle$

Disequalities between constructors are trivially true or decomposable.

$$x \neq a \vee b \neq y$$

Proceed with case-split.

Given a conjunction of disequalities:

$$x \neq T_1 \wedge \ldots \wedge x \neq T_n$$

We can always solve for *x*. We call *Δ* a computable function s.t.

$$\Delta(T_1,\ldots,T_n) \neq T_i$$

(Alternative Case-Splitting)

Case-Split
$$\frac{[\![\, \bar{a} \langle\, \varphi_1 \,\rangle \bar{x} \,]\!] \vdash \langle\, P_1 \mid \bar{T}_1 \,\rangle \qquad [\![\, \bar{a} \langle\, \varphi_2 \,\rangle \bar{x} \,]\!] \vdash \langle\, P_2 \mid \bar{T}_2 \,\rangle}{[\![\, \bar{a} \langle\, \varphi_1 \vee \varphi_2 \,\rangle \bar{x} \,]\!] \vdash \langle\, P_1 \vee P_2 \mid \textbf{if}(P_1)\, \bar{T}_1 \,\textbf{else}\, \bar{T}_2 \,\rangle}$$

$$\frac{[\![\, \bar{a} \langle\, \varphi_1 \,\rangle \bar{x} \,]\!] \vdash \langle\, P_1 \mid \bar{T}_1 \,\rangle \qquad [\![\, \bar{a} \langle\, \neg P_1 \wedge \varphi_2 \,\rangle \bar{x} \,]\!] \vdash \langle\, P_2 \mid \bar{T}_2 \,\rangle}{[\![\, \bar{a} \langle\, \varphi_1 \vee \varphi_2 \,\rangle \bar{x} \,]\!] \vdash \langle\, P_1 \vee P_2 \mid \textbf{if}(P_1)\, \bar{T}_1 \,\textbf{else}\, \bar{T}_2 \,\rangle}$$

May simplify the problem, but requires closing a branch first.

Synthesis for Arrays

- How do you handle $[\![a \langle \text{a[x]} = \text{x} \rangle x]\!]$?
- In general, need to generate a search over indices.
- Overall strategy:
  - Project φ on the index variables. Use as a filter for the search, or use an *enumerating* synthesis procedure.
  - Within the loop, the problem is purely in the element theory.

Implementation(s) Status

- *Comfusy*
  - Synthesis for (conjunctions in) integer linear arithmetic and sets with cardinality constraints.
  - Implemented as a compiler plugin for Scala.

Mayer, Kuncak, Piskac, Suter, "*Complete Functional Synthesis*", PLDI 2010.

Spielmann, Kuncak "*Synthesis for Unbounded Bit-Vector Arithmetic",* IJCAR 2012.

- Ongoing work:
  - System based on inference rules.
  - Mixture of *definite* rules and *heuristics* to handle problems that need more than straight-line code.

Jacobs, Kuncak, Suter, "*Reductions for Synthesis Procedures*", VMCAI 2013.

Reductions for Synthesis Procedures

- Synthesis: rewrite a relation into a function.
  - Synthesis procedures: algorithms that always succeed for a class of problems.

- Rule-based framework for designing synthesis procedures.
  - Allows for interleavings of theory steps.
  - Sets of rules for term algebras, linear arithmetic, arrays.

Thank you.