A Practical Overview of Deductive Program Synthesis

Alexander D. Weinert

RWTH Aachen University

Spring School on Program Synthesis
Papenburg 2012

Another Point of View

Traditional Programming

Specify *how* to do something

Program Synthesis

Specify *what* to do

Our task

Define methods for

- Specification
- Transformation of specification to program

Specification

Prerequisite

- Only consider side-effect-free programs
  - No GUI, no persistence, ...

Criteria

User-oriented

- Easy to write
- Oriented on well-known languages

Tool-oriented

- Unambiguous

$\Rightarrow$ Logic

Specifications in Logic

Natural formulation

Find a program `prog(x)` with output `y`, such that `post(x,y)` holds.
I assure that `pre(x)` holds.

Mathematical notation

`prog(x)` $\Leftarrow$ Find `y`, such that `post(x,y)`,
where `pre(x)`

Background theories

Integers, Strings, Sets, . . .

Expressions in logic

- Basic propositions: true, false, `a|b`, `char(x)`, $a \in X$, ...
- Basic functions: `gcd(x,y)`, `head(a)`, `union(X,Y)`, ...
- Equality: `x=y`, `last(a)=head(b)`, `union(X,Y)=union(A,B)`, ...
- Boolean connectives: $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \underset{\uparrow}{\rightarrow} \varphi_2$

  "if, then"
- Quantifiers: $\underset{\uparrow}{\exists} y.\varphi(y)$, $\underset{\uparrow}{\forall} y.\varphi(y)$,

  "there exists" "for all"

Valid formula

$$\texttt{pre(X)} := \forall n.\mathrm{Int}(n) \rightarrow n \in \texttt{X}$$

Specification of a square root program

Square root

$$\texttt{sqrt(x,}\epsilon\texttt{)} \quad \Leftarrow \quad \text{Find y, such that } y^2 \leq x \wedge x < (y+\epsilon)^2, \text{ where } \epsilon > 0$$

$$\texttt{post(x,}\epsilon\texttt{,y)} := y^2 \leq x \wedge x < (y+\epsilon)^2$$
$$\texttt{pre(x,}\epsilon\texttt{)} := \epsilon > 0$$

Front and last element of lists

Front and last element

$$\langle \mathit{front}(s), \mathit{last}(s)\rangle \quad \Leftarrow \quad \begin{array}{l}\text{Find } \langle y, z\rangle\text{, such that}\\ \texttt{char}(z) \wedge s = y \cdot z,\\ \text{where } \neg(s = \epsilon)\end{array}$$

$$\texttt{post(s,y,z)} := (\texttt{char}(z) \wedge s = y \cdot z)$$
$$\texttt{pre(s)} := s \neq \epsilon$$

The method

General idea

We want to show: There exists a program fulfilling the specification

Proof has to be "sufficiently constructive"

Reminder

$$\texttt{prog(x)} \quad \Leftarrow \quad \text{Find } \texttt{y}\text{, such that } \texttt{post(x,y)}\text{, where } \texttt{pre(x)}$$

To Show

If `pre(x)` holds, then `post(x,y)` holds for some `y`.

Tableau notation

Assertions	Goals	Outputs
pre(x)		
	post(x,y)	y

Output some y that fulfills the postcondition

General tableaux

Assertions	Goals	Outputs
$A_{c,1}(x)$		$t_{c,1}(x)$
$A_{c,2}(x)$		$t_{c,2}(x)$
$\vdots$		$\vdots$
$A_{c,m}(x)$		$t_{c,m}(x)$
	$G_{c,1}(x)$	$t_{c,m+1}(x)$
	$G_{c,2}(x)$	$t_{c,m+2}(x)$
	$\vdots$	$\vdots$
	$G_{c,n}(x)$	$t_{c,m+n}(x)$

Associated Sentence

If for all x : $A_{c,1}(x)$ and $A_{c,2}(x)$ and $\ldots A_{c,m}(x)$

then there exists some x : $G_{c,1}(x)$ or $G_{c,2}(x)$ or $\ldots G_{c,n}(x)$

Steps of Deduction

Assertions	Goals	Outputs
`pre(x)`		
	`post(x,y)`	`y`

$\rightsquigarrow$ $\Leftarrow$ Deduction Rules

Assertions	Goals	Outputs
	`true`	...

$\leftarrow$ generated program

or

Assertions	Goals	Outputs
`false`		...

$\leftarrow$ generated program

General Form

Prerequisite

Assertions	Goals	Outputs
$A_{c,1}(x)$		$t_{c,1}(x)$
$\vdots$		$\vdots$
$A_{c,m}(x)$		$t_{c,m}(x)$
	$G_{c,1}(x)$	$t_{c,m+1}(x)$
	$\vdots$	$\vdots$
	$G_{c,n}(x)$	$t_{c,m+n}(x)$

Deduction

Assertions	Goals	Outputs
$A'_{c,1}(x)$		$t'_{c,1}(x)$
$\vdots$		$\vdots$
$A'_{c,m'}(x)$		$t'_{c,m'}(x)$
	$G'_{c,1}(x)$	$t'_{c,m'+1}(x)$
	$\vdots$	$\vdots$
	$G'_{c,n'}(x)$	$t'_{c,m'+n'}(x)$

Duality

Duality

Assertions	Goals	Outputs
$\neg\varphi$		t

$$\rightsquigarrow$$

Assertions	Goals	Outputs
	φ	t

$$\begin{aligned} & \cdots \land \forall x.\neg\varphi(x) \\ \Leftrightarrow \quad & \cdots \land \neg\exists x.\varphi(x) \\ \Leftrightarrow \quad & \cdots \lor \exists x.\varphi(x) \end{aligned}$$

Duality: Example

Assertions	Goals	Outputs
$s \neq \epsilon$		

$$\rightsquigarrow$$

Assertions	Goals	Outputs
$s \neq \epsilon$		
	$s = \epsilon$	

Splitting Rules

AND-Splitting

Assertions	Goals	Outputs
$\varphi \wedge \psi$		t

$\rightsquigarrow$

Assertions	Goals	Outputs
φ		t
ψ		t

OR-Splitting

Assertions	Goals	Outputs
	$\varphi \vee \psi$	t

$\rightsquigarrow$

Assertions	Goals	Outputs
	φ	t
	ψ	t

Recall:
If for all x all assertions hold,
then there exists some x, such that some goals hold.

Splitting Rules

IF-Splitting

Assertions	Goals	Outputs
	If φ, then ψ	t

$\rightsquigarrow$

Assertions	Goals	Outputs
	$\neg\varphi \vee \psi$	t

$\rightsquigarrow$

Assertions	Goals	Outputs
φ		t
	ψ	t

Splitting: Example

Assertions	Goals	Outputs
	if $s \neq \epsilon$, then $s = t_1 \cdot t_2$	

$\rightsquigarrow$

	Assertions	Goals	Outputs	
		if $s \neq \epsilon$, then $s = t_1 \cdot t_2$		
Disprove this $\rightarrow$	$s \neq \epsilon$			or show
		$s = t_1 \cdot t_2$		$\leftarrow$ this

Simplification

Let $\varphi \equiv \psi$

Assertions	Goals	Outputs
φ		t

$\rightsquigarrow$

Assertions	Goals	Outputs
ψ		t

Example

Assertions	Goals	Outputs
$\neg(a \wedge b)$		a

$\rightsquigarrow$

Assertions	Goals	Outputs
$\neg a \vee \neg b$		a

Equality

Assertions	Goals	Outputs
$\phi(\tau = \sigma)$		s
$\psi(\tau)$		t

$$\rightsquigarrow$$

Assertions	Goals	Outputs
$\phi(\texttt{false})$ $\vee$ $\psi(\sigma)$		if$(\tau = \sigma)$ then t else s

Equality: Example

Assertions	Goals	Outputs
$x = 5 \wedge y = 2$		
$x \cdot y = 10$		

$\rightsquigarrow$

Assertions	Goals	Outputs
$\mathit{false} \wedge y = 2$		
$5 \cdot y = 10$		

Resolution

GG-Resolution

Assertions	Goals	Outputs
	$\varphi(\tau)$	s
	$\psi(\tau)$	t

$$\rightsquigarrow$$

Assertions	Goals	Outputs
	$\varphi(\text{true}) \wedge \psi(\text{false})$	if τ then s else t

Resolution

AA-Resolution

Assertions	Goals	Outputs
$\varphi(\tau)$		s
$\psi(\tau)$		t

$\rightsquigarrow$

Assertions	Goals	Outputs
$\varphi(\text{true}) \vee \psi(\text{false})$		if τ then s else t

Also: AG-Resolution, GA-Resolution

Resolution: Example

Assertions	Goals	Outputs
	Integer(x)	true
	$\neg$ Integer(x)	false

$\rightsquigarrow$

Assertions	Goals	Outputs
	true	if Integer(x), then true else false

Conditional Substitution

$$r \Rightarrow s \text{ if } C$$

Example

$$n|0 \Rightarrow \ \text{true} \ \text{ if } \texttt{integer(n)} \wedge n \neq 0$$

Assertions	Goals	Outputs
$\varphi(r)$		$t(r)$

$\rightsquigarrow$

Assertions	Goals	Outputs
if C then $\varphi(s)$		$t(s)$

Assertions	Goals	Outputs
	$\varphi(r)$	$t(r)$

$\rightsquigarrow$

Assertions	Goals	Outputs
	$C \wedge \varphi(s)$	$t(s)$

Conditional Substitution: Example

Assertions	Goals	Outputs
	$(1 < x \wedge x < 2) \vee (x \mid 0)$	x

Example

$$n \mid 0 \Rightarrow \text{ true } \text{ if } \texttt{integer(n)} \wedge n \neq 0$$

$$\rightsquigarrow$$

Assertions	Goals	Outputs
	$(1 < x \wedge x < 2) \vee (\texttt{integer(n)} \wedge x \neq 0 \wedge \mathit{true})$	x
	$(1 < x \wedge x < 2) \vee (\texttt{integer(n)} \wedge x \neq 0)$	x

Induction $\leftrightarrow$ Recursion

Mathematical Induction

- Start of induction
- Induction Hypothesis
- Induction step

Recursion

- Ground case
- Recursive call
- Use of recursive result

Observation

Induction $\hat{=}$ Recursion

Recursion rule

Assumption

$<$ is a well-founded relation

Assertions	Goals	f(in)
`pre(in)`		
	`post(in,out)`	out

$\rightsquigarrow$

Assertions	Goals	f(in)
If $z < in$, then if `pre(z)`, then `post(z,f(z))`		

Recursion: Example

Assertions	Goals	last(s)
$s \neq \epsilon$		
	$\texttt{char}(t_2) \wedge s = t_1 \cdot t_2$	t_2

$$\rightsquigarrow$$

Assertions	Goals	Outputs
if $s' < s \wedge s' \neq \epsilon$, then $\texttt{char(last(s'))}$ $\wedge s' = t'_1 \cdot \texttt{last(s')}$		

Derivation of Front and Last

Basic proposition: char(x)
Basic functions: Concatenation $(\cdot)$, head(x), tail(x)

$\langle$`front(s),last(s)`$\rangle \quad \Leftarrow \quad$ Find $\langle t_1, t_2 \rangle$, such that char$(t_2) \land s = t_1 \cdot t_2$, where $\neg(s = \epsilon)$

$\rightsquigarrow$

No.	Assertions	Goals	`front(s)`	`last(s)`
1.	$\neg(s = \epsilon)$			
2.		char$(t_2) \land s = t_1 \cdot t_2$	t_1	t_2

Variables: t_1, t_2
Constants: s

No.	Assertions	Goals	`front(s)`	`last(s)`
1.	$\neg(s = \epsilon)$			
2.		$\mathrm{char}(t_2) \wedge s = t_1 \cdot t_2$	t_1	t_2

Target

- Recursive program
  - Base case: only one character
  - Recursive case: more than one character

No.	Assertions	Goals	`front(s)`	`last(s)`
3.		$\mathrm{char}(t_2) \wedge s = t_2$	ϵ	t_2

Resolution

No.	Assertions	Goals	front(s)	last(s)
1.	$\neg(s = \epsilon)$			
2.		$\mathrm{char}(t_2) \wedge s = t_1 \cdot t_2$	t_1	t_2
3.		$\mathrm{char}(t_2) \wedge s = t_2$	ϵ	t_2
	$x = x$			
		$\neg(x = x)$		

$$\begin{array}{rcl} \mathrm{char}(t_2) \wedge s = t_2 & & \neg(x = x) \\ \downarrow t_2 := s & & \downarrow x := s \\ \mathrm{char}(s) \wedge \boxed{s = s} & & \neg(\boxed{s = s}) \\ \mathrm{char}(s) \wedge \boxed{\mathrm{true}} & \wedge & \neg(\boxed{\mathrm{false}}) \end{array}$$

$$\Rightarrow \mathrm{char}(s)$$

Recursion

No.	Assertions	Goals	`front(s)`	`last(s)`
1.	$\neg(s = \epsilon)$			
2.		$\mathrm{char}(t_2) \wedge s = t_1 \cdot t_2$	t_1	t_2
4.		$\mathrm{char}(s)$	ϵ	s
5.		$\mathrm{char}(u) \wedge \mathrm{char}(t_2)$ $\wedge s = u \cdot t_1 \cdot t_2$	$u \cdot t_1$	t_2

Induction Hypothesis

If $x < s$, then
if $\neg(x = \epsilon)$, then
$\mathrm{char}(\mathtt{last}(x)) \wedge x = \mathtt{front}(x) \cdot \mathtt{last}(x)$

Recursion (cont.)

No.	Assertions	Goals	front(s)	last(s)
5.		$\mathrm{char}(u) \wedge \mathrm{char}(t_2)$ $\wedge s = u \cdot t_1 \cdot t_2$	$u \cdot t_1$	t_2

Induction Hypothesis

If $x < s \wedge \neg(x = \epsilon)$, then $\mathrm{char}(\mathtt{last}(x)) \wedge x = \mathtt{front}(x) \cdot \mathtt{last}(x)$

$$t_1 := \mathtt{front}(x), t_2 := \mathtt{last}(x)$$

No.	Assertions	Goals	front(s)	last(s)
5.		$x < s \wedge \neg(x = \epsilon)$ $\mathrm{char}(u) \wedge \mathrm{char}(\mathtt{last}(x))$ $\wedge s = u \cdot x$	$u \cdot \mathtt{front}(x)$	$\mathtt{last}(x)$

Recursion (simplification)

Induction Hypothesis

If $x < s \wedge \neg(x = \epsilon)$, then $\mathrm{char}(\mathtt{last}(x)) \wedge x = \mathtt{front}(x) \cdot \mathtt{last}(x)$

No.	Assertions	Goals	front(s)	last(s)
5.		$x < s \wedge \neg(x = \epsilon)$ $\mathrm{char}(u) \wedge \mathrm{char}(\mathtt{last}(x))$ $\wedge s = u \cdot x$	$u \cdot \mathtt{front}(x)$	$\mathtt{last}(x)$

No.	Assertions	Goals	front(s)	last(s)
5.		$x < s \wedge \neg(x = \epsilon)$ $\mathrm{char}(u) \wedge s = u \cdot x$	$u \cdot \mathtt{front}(x)$	$\mathtt{last}(x)$

Recursion (simplification cont.)

No.	Assertions	Goals	front(s)	last(s)
5.		$x < s \land \neg(x = \epsilon)$ $\text{char}(u) \land s = u \cdot x$	$u \cdot \texttt{front}(x)$	$\texttt{last}(x)$

Decomposition Lemma

If $\neg(y = \epsilon)$, then $y = \texttt{head}(y) \cdot \texttt{tail}(y)$

$$y := s, u := \texttt{head}(y), x := \texttt{tail}(y)$$

No.	Assertions	Goals	front(s)	last(s)
5.		$\texttt{tail}(s) < s \land$ $\neg(\texttt{tail}(s) = \epsilon) \land$ $\text{char}(\texttt{head}(s)) \land$ $\neg(s = \epsilon)$	$\texttt{head}(s) \cdot$ $\texttt{front}($ $\texttt{tail}(y)$ $)$	$\texttt{last}($ $\texttt{tail}(s)$ $)$

Recursion (simplification cont.)

No.	Assertions	Goals	front(s)	last(s)
5.		$\mathtt{tail}(s) < s \wedge$ $\neg(\mathtt{tail}(s) = \epsilon) \wedge$ $\mathrm{char}(\mathtt{head}(s)) \wedge$ $\neg(s = \epsilon)$	$\mathtt{head}(s) \cdot$ $\mathtt{front}($ $\mathtt{tail}(y))$	$\mathtt{last}($ $\mathtt{tail(s)}$ $)$

Resolution

Domain knowledge: $\neg(s = \epsilon) \rightarrow \mathrm{char}(\mathtt{head}(s))$ and
$\neg(\mathtt{head}(s) = \epsilon) \rightarrow \mathtt{tail}(s) < s$.
Formally: Resolution

No.	Assertions	Goals	front(s)	last(s)
5.		$\neg(\mathtt{tail}(s) = \epsilon) \wedge$ $\neg(s = \epsilon)$	$\mathtt{head}(s) \cdot$ $\mathtt{front}(\mathtt{tail}(y))$	$\mathtt{last}($ $\mathtt{tail(s)})$

Simplification

No.	Assertions	Goals	front(s)	last(s)
5.		$\neg(\mathtt{tail}(s) = \epsilon) \wedge \neg(s = \epsilon)$	$\mathtt{head}(s) \cdot \mathtt{front}(\mathtt{tail}(y))$	last(tail(s))

Trichotomy property

Domain Knowledge:

$$y = \epsilon \vee \text{char}(y) \vee \neg(\mathtt{tail}(y) = \epsilon)$$

No.	Assertions	Goals	front(s)	last(s)
5.		$\neg(s = \epsilon) \wedge \neg(\text{char}(s))$	$\mathtt{head}(s) \cdot \mathtt{front}(\mathtt{tail}(y))$	last(tail(s))

Final steps

No.	Assertions	Goals	`front(s)`	`last(s)`
5.		$\neg(s = \epsilon) \wedge \neg(\text{char}(s))$	$\texttt{head}(s) \cdot \texttt{front}(\texttt{tail}(y))$	$\texttt{last}(\texttt{tail(s)})$
1.	$\neg(s = \epsilon)$			
		$\neg(\text{char}(s))$	$\texttt{head}(s) \cdot \texttt{front}(\texttt{tail}(y))$	$\texttt{last}(\texttt{tail(s)})$
4.		$\text{char}(s)$	ϵ	s
		true	if $\text{char}(s)$, then ϵ, else $\texttt{head}(s) \cdot \texttt{front}(\texttt{tail}(y))$	if $\text{char}(s))$, then s, else $\texttt{last}(\texttt{tail(s)})$

Final remarks

- Very good result in this case
- Efficient choice of rules by humans
- Not clear which axioms have to be used
- Result might not always be understandable
- No specification of performance

Thank you for your attention

Manna, Z. and Waldinger, R. (1980).
A Deductive Approach to Program Synthesis.
IEEE Transactions on Programming Languages and Systems, 2(1):90–121.

Manna, Z. and Waldinger, R. (1992).
Fundamentals of Deductive Program Synthesis.
IEEE Transactions on Software Engineering, 18(8):674–702.