TÉCNICO
LISBOA

Program Synthesis

An overview of logic-based approaches

Ricardo Brancas

Presentation Outline

1. What is it?

2. Logic Background

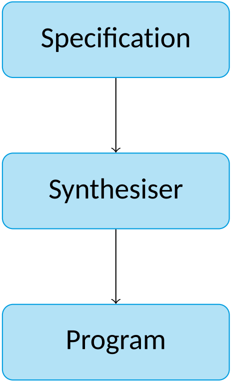
3. Anatomy of a Synthesiser

Program Synthesis

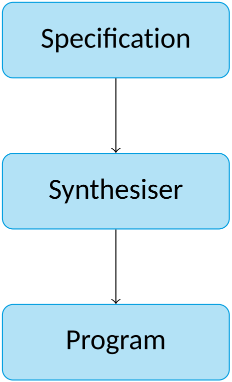
Specification

Program

Program Synthesis



Program Synthesis



Input/Output Examples
First-order Logic Specification
Hoare Logic Specification
Separation Logic Specification
...

Proof of Correctness

How can we check if a program is correct (wrt. to the specification)?

Proof of Correctness

How can we check if a program is correct (wrt. to the specification)?

We can use a logic formulation.

Presentation Outline

1. What is it?
2. Logic Background
3. Anatomy of a Synthesiser

First-order Logic

First-order Logic (FOL)

Consists of expressions, quantified over variables, containing predicates and functions.

Example

$\forall x. P(x)$

$\exists y \forall x. Q(y, x) \land P(f(y))$

First-order Logic

First-order Logic (FOL)

Consists of expressions, quantified over variables, containing predicates and functions.

Example

$\forall x.P(x)$

$\exists y \forall x.Q(y, x) \wedge P(f(y))$

First order logic is undecidable (in general, it is impossible to check if a given formula is correct). How can we solve this?

Satisfiability Modulo Theories

Satisfiability Modulo Theory (SMT)

A decision problem on decidable subsets of first order logic.
Such subsets are called theories.

T-satisfiability

Given a theory T, we say that a formula Φ is T-satisfiable iff there is some model M of T, such that Φ holds in M.

A model can be seen as a mapping from variables to constants/functions.

Theories

Theory

A subset of first-order logic that is decidable. Theories can be combined to create more expressive something.

Example: Equality with Uninterpreted Functions

$(f(b) = d) \wedge (f(a) + f(b) = d) \wedge (a = d)$

Theories

Theory

A subset of first-order logic that is decidable. Theories can be combined to create more expressive something.

Example: Equality with Uninterpreted Functions

$$(f(b) = d) \wedge (f(a) + f(b) = d) \wedge (a = d)$$

Example: Linear Arithmetic

$3x + 2y < 3$

$x + y + z = 45$

Other Useful Theories for PL

- Bit vectors
- Arrays
- Pointer logic
- Quantified Theories

Proof of Correctness (II)

How can we check if a program is correct (wrt. to the specification)?

We can use a logic formulation.

Simple Hoare Triple

```
{ X = 3 }
Y := X - 2
X := X - 1
{ X = 2 ∧ Y = 1}
```

SMT Formula

$$X_0 = 3$$
$$\land \; Y_0 = (X_0 - 2)$$
$$\land \; X_1 = (X_0 - 1)$$
$$\land \; X_1 = 2 \land Y_0 = 1$$

Proof of Correctness (II)

How can we check if a program is correct (wrt. to the specification)?

We can use a logic formulation.

Simple Hoare Triple

```
{ X = 3 }
Y := X - 2
X := X - 1
{ X = 2 ∧ Y = 1}
```

SMT Formula

$$
\begin{aligned}
& X_0 = 3 \\
\wedge\ & Y_0 = (X_0 - 2) \\
\wedge\ & X_1 = (X_0 - 1) \\
\wedge\ & X_1 = 2 \wedge Y_0 = 1
\end{aligned}
$$

We say that a program is correct, if the corresponding SMT formula is satisfiable.

Presentation Outline

1. What is it?
2. Logic Background
3. Anatomy of a Synthesiser

Generating Programs

Now we can check if a given program is correct. But how do we generate them?

Generating Programs

Now we can check if a given program is correct. But how do we generate them?

Enumeration

We can use the grammar of the language to generate (enumerate) all possible programs.

Generating Programs

Now we can check if a given program is correct. But how do we generate them?

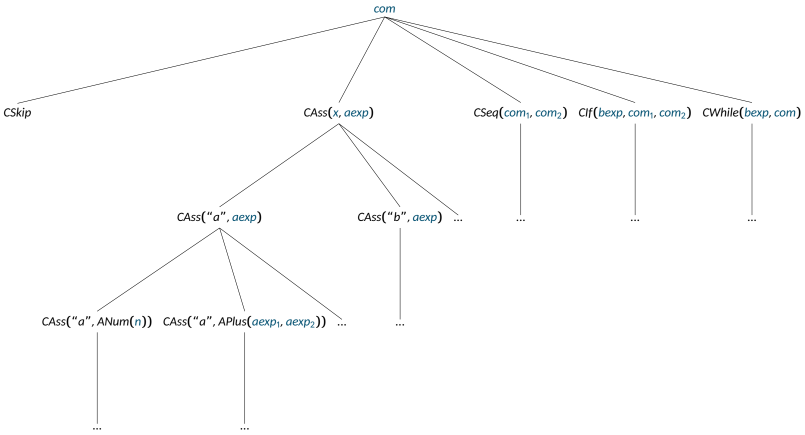
Enumeration

We can use the grammar of the language to generate (enumerate) all possible programs.

Problem

The space of possible programs is exponentially large. It is impossible to just check all programs.

Generating Programs (Imp language)



Deduction Engine

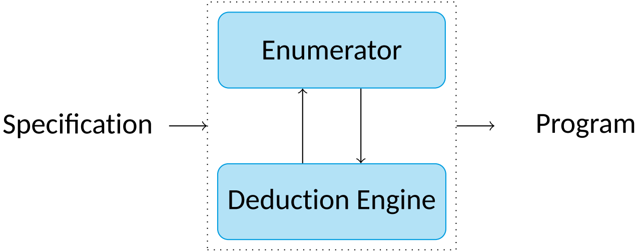
The job of the deduction engine is:

- To prune the search when incorrect programs are generated.

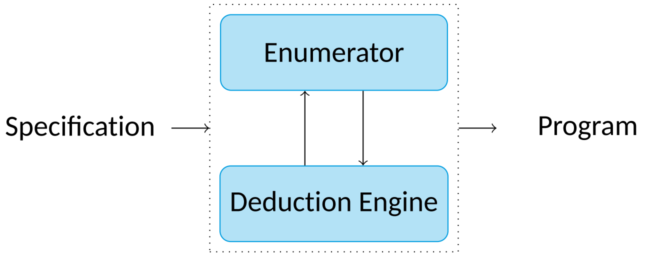
Example

```
TODO
```

The simplest synthesiser



The simplest synthesiser



Problem

The enumerator is dumb. It can get stuck generating programs that will never satisfy the specification.
This is the central problem in program synthesis.