Iman PoernomoSchool of Con

Overview

-
-
- Proofs-as-imperative-as

Constructive program synthesis

- Proofs

The general idea

The Curry-hism

Example: ND rules as type inference rules

Introduction rule for universal quantification corresponds to a type inference rules for lambda abstraction with dependent product type

$$\frac{\Delta \vdash A[y/x]}{\Delta \vdash \forall x : s \bullet A}\ (\forall\text{-I})$$

$$\Delta \vdash \lambda x : s.p^{\forall x : s \bullet A}$$

Example: Encoding a proof as a lambda term

Normalization

- Proof normalization corresponds to lambda reduction
-

- Based directly on the Curry-Howard rwomorphism
- The lambda calculus of the logical type theory is considered as a programming language
- Formulae = types = specifications of required realizing programs
- Normalization = lambda term reduction =

Realizability

Example: proof as realizing

-

- Specification of programs given by a different *modified realizability* relation between formulae

- Proof terms in the LTT are transformed into prograare uft satisfy the proved specification pormulae (as modified reali oers)

Modified realizability

- Our modified realizers are simply typed SML programs – they are correct with respect to a specification, but do *not* carry proofs of correctness with them.
- Formally, a program p is a modified realizer of a formula A when p can be used in place of the Skolem function for the

Skolem form

- The Skolem form of

  dbConnected V – dbConnected

is equivalent to

  f=inl() -> dbConnected &

  f=inr() -> -dbConnected

-

- Modified realizer does not prove this fact – it exists in a separate language

Transforming proofs into programs

- We use an *extraction map* to transform a prooftgram

Extraction

The Curry-Howard protocol

- How should this transformative style of program synthesis be generalized over arbitrary logics and programming paradigms?
- Logics: Linear logic, Modal logics, Hoare logic, Hennessy-Milner proof systems, etc
- Programming paradigms: Imperative,

The protocol

Roles in the protocol

[illegible]	[illegible]	[illegible] semantics.

Relations between roles (1)

[illegible] to programs [illegible] extract from proofs of the [illegible] of the [illegible] such that, given a proof of [illegible] specification, the extracted program [illegible] the specification.

A useful generalizing framework?

- It can be seen that functional/constructive logic tran .1fmative proofs -as-programs con.1fms to this framework
- The usefulness of the protocol as a

Application:

Problem

- Want to be able to adapt programs t
  Hoare logic for synthesis of correct *return values*

Hoare Logic

Hoare logic

Correct synthesis of return values

- The presence of side
- However, side-effectfree return values are also TD 0 0 0 rg

How to adapt constructive results?

- Follow the Curry-Howard protocol
-

Constructive Hoare logic

Core rules

- Program/formulae pairs used in the Hoare logic are treated as types in a corresponding type theory
-
- A typing judgment corresponds to a proof derivation

Core type inference rules

(seq)

(loop)

where A does not contain any state identifiers

(cons)

$\vdash_{LTT(\mathsf{IHL})} \mathsf{cons}(q_1, q_2)$

- A program p is a *return value realizer*

Example

it is true that

Realizability

- An SML program p with return values realizes a specification in the Hoare logic q**s**A when
  - p

Extraction

- We extend the extraction map of ordinary proofs-as-programs to our case, to extract return-value realizers from proofs in the Hoare logic
- The map is from our LTT to our CTT (SML with return values)

Example cases:

- Extraction over Harrop terms is trivial: the program of the
- Extraction over (ite2hs7a e results in conditional statement with alternate return values
-

- Return value specified as required return value realizer

Require a return value x
that is an appropriate response

Pre-condition – PIN has been
entered into machine

ATM example

We use the following axioms that define the domain knowledge about the behaviour of the ATM machine

This axiom says that, for any program, if the card is destroyed, it is appropriate to tell the user about it

This program destroys the card if the PIN is incorrect

[redacted]

Proof-term

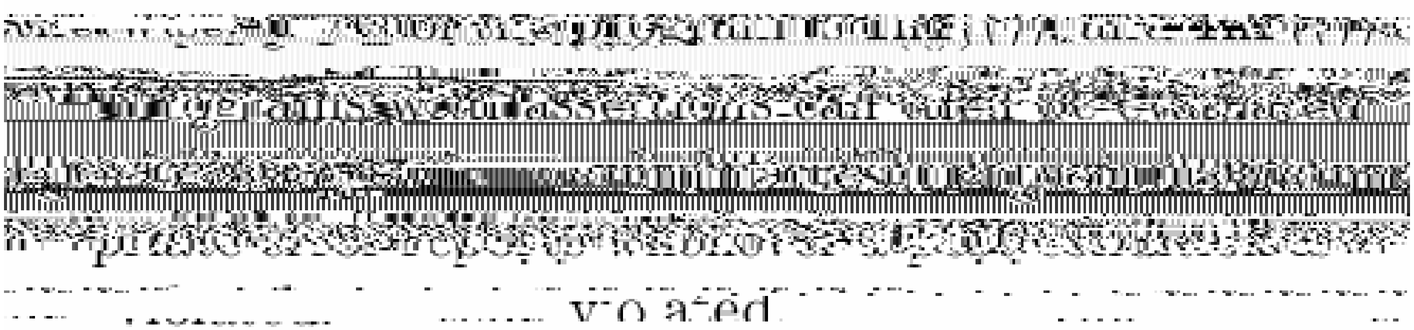
Extraction: Example

by 0 rg ation: Design

- Design-by-contract is a well established method of

Contracts as return values

- Assertions are a special kind of return value
- We can simulate assertions in SML as return values of disjoint union type



contracts

Specification and Synthesis of

Improving programs built with faults

Improving programs built with faults

- Instead of using formulae that assert the correctness of programs, we use disjunctive statements stating that the program may or may not be faulty.
- Because disjunctive statements correspond to post-condition assertion 0kour synthesis enables the automatic construction of a program with

Example

- By our synthesis methods, proofs that involve this theorem can also be transformed into programs that use a version of `connectDB` with
-

- The Curry

Further reading and Questions

- Iman Poernomo, John N. Crossley, *Protocols between Programs and Proofs*, in Kung-Kiu Lau (Ed.), LOPSTR 2000, LNCS 2042, pp. 18-37
-
- Iman Poernomo, John N. Crossley, Martin Wirsing, *Programs, Proofs and Parameterized Specifications*
-