Minlog – A Tool for Program Extraction Supporting Algebras and Coalgebras

Ulrich Berger[1], Kenji Miyamoto[2*], Helmut Schwichtenberg[2], and Monika Seisenberger[1]

[1] Swansea University, Wales
[2] Ludwig Maximilian University, Munich

* Supported by the Marie Curie Initial Training Network in Mathematical Logic – MALOA – From MAthematical LOgic to Applications, PITN-GA-2009-238381

30.08.2011
CALCO-tools 2011

Contents of this talk

- Introduction
  - Proof Assistant Minlog [Min]
  - Theory of Computable Functionals (TCF in short) [SW11]
- Demo of Program Extraction Case Studies on Minlog
  - Parser
    - Input: a string of parentheses
    - Output: True and the parse tree if the input is balanced
      False and the empty parse tree if the input is not balanced
  - Translator
    - Input: a rational number
    - Output: a real number representation of the input

Proof Assistant Minlog

- Implementation of TCF
- Program extraction supporting (co)induction
- Written in Scheme Language (R5RS)
- User's work in Minlog is in Scheme as well

Example of a Minlog Proof

```
(load "~/minlog/init.scm")

(add-pvar-name "A" "B" (make-arity))

(set-goal "A -> B -> A")
(assume "HypA" "HypB")
(use "HypA")
(save "theorem")
```

Theory of Computable Functionals (TCF)

- First order minimal natural deduction
  - Classical Logic as an Fragment of Minimal Logic
- Goedel's T with extensions
- Semantics
  - Scott-Ershov model of partial continuous functionals
  - Free algebras as base types
  - Algebras are domains of Scott's information systems
- Program Extraction
  - Kreisel's modified realizability interpretation
  - A-Translation and Dialectica Interpretation available for classical proofs

Examples of Free Algebras

1. *Par* (Parentheses)

$$L^{Par}, R^{Par}$$

2. $\mathbb{N}$ (Natural Numbers)

$$0^{\mathbb{N}}, S^{\mathbb{N}\to\mathbb{N}}$$

3. $\mathbb{L}(\rho)$ (List of type ρ)

$$Nil_{\rho}{}^{\mathbb{L}(\rho)}, Cons_{\rho}{}^{\rho\to\mathbb{L}(\rho)\to\mathbb{L}(\rho)}$$

4. $\mathbb{I}$ (Interval [-1,1])

$$I^{\mathbb{I}}, C_{-1}{}^{\mathbb{I}\to\mathbb{I}}, C_0{}^{\mathbb{I}\to\mathbb{I}}, C_1{}^{\mathbb{I}\to\mathbb{I}} \text{ (Whole Interval, Left, Middle, Right)}$$

5. $\mathbb{O}$ (Ordinal, non-finitary)

$$Zero^{\mathbb{O}}, Succ^{\mathbb{O}\to\mathbb{O}}, Sup^{(\mathbb{O}\to\mathbb{O})\to\mathbb{O}}$$

Totality and Cototality

Total ideals of a base type are in a finite constructor expression.

- True, False
- 0, S(S(S0))
- Nil, L::R:

Cototal ideals of a base type are total or in a non-wellfounded constructor expression.

- True, False
- 0, S(S(S0)), S(S(S(S(S(S(S(...
- Nil, L::R:, L::R::L::R::L::R::...

f of a higher type $\sigma \to \delta$ is total if: For any total x^σ, fx is total.

Case Study on Parser

- Prove $\forall x(Sx \vee \neg Sx)$
  - x is a list of parentheses
  - Sx says that x is balanced, predicate S inductively defined
- Extract a program from proofs
- Experiments

Extracted Parser in Goedel's T

```
[x0]
  Test 0 x0@
  (Rec list par=>algState=>algS=>algS)
    x0
    ([st1,b2][if st1 b2 ([b3,st4]CInitS)])
    ([par1,x2,f3,st4,b5]
      [if par1
        (f3(CApState b5 st4)CInitS)
        [if st4 CInitS
                ([b6,st7]f3 st7(CApS b6(CParS b5)))]])
   CInitState
   CInitS
```

Experiments

- Input $L :: L :: R :: R :$

```
(pp (nt (mk-term-in-app-form parser-term
                             (pt "L::L::R::R:"))))
```

$\Longrightarrow$ True@CApS CInitS(CParS(CApS CInitS(CParS CInitS)))

- Input $R :: L :$

```
(pp (nt (mk-term-in-app-form parser-term
                             (pt "R::L:"))))
```

$\Longrightarrow$ False@CInitS

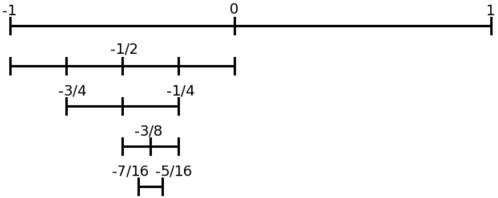
Computational Content from (Co)Inductively Defined Predicates

- Defining Sx to tell that x is balanced
  - $S(Nil)$
  - $\forall x(Sx \to S(LxR))$
  - $\forall xy(Sx \to Sy \to S(xy))$
- Algebra ι_S for parse trees obtained from S
  - $\texttt{CInitS}^{\iota_S}$ from $S(Nil)$
  - $\texttt{CParS}^{\iota_S \to \iota_S}$ from $\forall x(Sx \to S(LxR))$
  - $\texttt{CApS}^{\iota_S \to \iota_S \to \iota_S}$ from $\forall x(Sx \to Sy \to S(xy))$

In the next case study, we obtain the interval algebra from a coinductively defined predicate.

Signed Digit Stream Representation of Real Numbers

- Representing real numbers in SDS [CDG06]
- SDS is a stream (or non-wellfounded list) of signed digits $-1, 0, 1$
  - Example. $-1 :: 0 :: 1 :: 0 :: 1 :: 0 :: 1 :: \ldots$
- Represented as a cototal ideal in TCF
- SDS tells how to compute rational intervals as accurate as required
- A real number represented by $-1 :: 0 :: 1 :: 0 :: 1 :: 0 :: 1 :: \ldots$



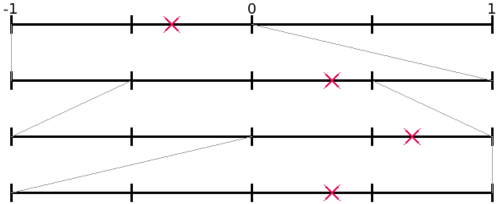
An approximation of $-\frac{1}{3}$.

Idea for the Translator

We construct an SDS from a real number.

- Take an appropriate signed digit for the given $x \in [-1, 1]$
  1. If x is in the left, take -1 and let the next x be $2x + 1$
  2. If x is in the middle, take 0 and let the next x be $2x$
  3. If x is in the right, take 1 and let the next x be $2x - 1$
- Since $x \in [-1, 1]$, we can repeat it as many as required

Example. $-\frac{1}{3}$ in SDS



We obtain an SDS $-1 :: 0 :: 1 :: \ldots$

Case Study on Translator

- Theorem: if rational $a \in [-1, 1]$, a is approximable in SDS.
- Proof by coinduction
- Extracting a program from the proof
- Experiments

We describe the theorem in the following formula:

$$\forall_a (Q\ a \to {}^{co}I\ a)$$

$Q\ a$ holds if $a \in [-1, 1]$. ${}^{co}I$ is defined coinductively.

Coinductively Defined Predicate ${}^{co}I$

A predicate P to say that a is approximable.

- If $P\ a$ holds
  1. a is left and $P(2a+1)$ or
  2. a is middle and $P(2a)$ or
  3. a is right and $P(2a-1)$

Such a predicate can be defined by coinduction.

$$
\begin{aligned}
{}^{co}I\ a \to a = 0 \lor \exists_b\Big(a = \frac{b+1}{2} \land {}^{co}I\ b\Big) \\
\lor \exists_b\Big(a = \frac{b}{2} \land {}^{co}I\ b\Big) \\
\lor \exists_b\Big(a = \frac{b-1}{2} \land {}^{co}I\ b\Big)
\end{aligned}
$$

This formula is also used as a coclosure axiom, written ${}^{co}I^{-}$.

Coinduction

Coinduction axiom ${}^{co}I^{+}$ is yielded from the definition of ${}^{co}I$.
Set theoretically,

$$X \subseteq \Phi(X) \to X \subseteq \nu\Phi \quad \text{(coinduction)}$$

where Φ a monotone operator, ν the greatest fixed point operator.
In our setting, we give a GFP axiom:

$$\begin{aligned}
\forall a(P\ a \to a = 0 \lor \exists_b(a = \frac{b+1}{2} \land P(b)) \\
\lor \exists_b(a = \frac{b}{2} \land P(b)) \\
\lor \exists_b(a = \frac{b-1}{2} \land P(b))) \\
\to P\ a \to {}^{co}I\ a
\end{aligned}$$

P is an arbitrary predicate.

Proof Sketch

We show $\forall a(Q\ a \to {}^{co}I\ a)$. Assume a. We prove $Q\ a \to {}^{co}I\ a$ by means of the following GFP axiom with substituting Q for P.

$$\begin{aligned}\forall a(Q\ a \to a = 0 \lor \exists_b(a = \frac{b+1}{2} \land Q(b)) \\ \lor \exists_b(a = \frac{b}{2} \land Q(b)) \lor \exists_b(a = \frac{b-1}{2} \land Q(b))) \\ \to Q\ a \to {}^{co}I\ a\end{aligned}$$

What we have to show is the first premise

$$\begin{aligned}\forall a(Q\ a \to a = 0 \lor \exists_b(a = \frac{b+1}{2} \land Q(b)) \\ \lor \exists_b(a = \frac{b}{2} \land Q(b)) \lor \exists_b(a = \frac{b-1}{2} \land Q(b)))\end{aligned}$$

It is done by the case distinction on a

$$a \in [-1, 0] \text{ or } a \in [-\frac{1}{2}, \frac{1}{2}] \text{ or } a \in [0, 1]$$

Coinduction on Minlog

```
input> (set-goal "allnc a^(Q a^ -> CoI a^)")
;?_1:allnc a^(Q a^ -> CoI a^)

input> (assume "a^0")
;ok, we now have the new goal
;?_2:Q a^0 -> CoI a^0 from
;  {a^0}

input> (coind)
;ok, ?_2 can be obtained from
;?_3:allnc a^(
;      Q a^ ->
;      a^ eqd 0 orr
;      exr a^0(a^ eqd(a^0-1)/2 & (CoI a^0 ord Q a^0)) ord
;      exr a^0(a^ eqd a^0/2 & (CoI a^0 ord Q a^0)) ord
;      exr a^0(a^ eqd(a^0+1)/2 & (CoI a^0 ord Q a^0))) from
;  {a^0}  1:Q a^0
```

Program Extraction via Realizability Interpretation

- Decoration of Logical Connectives
  - $\to^c$, $\to^{nc}$, $\forall^c$, $\forall^{nc}$
  - ^{c} stands for computational, ^{nc} for non-computational
  - Logically same, Computationally different
- Modified Realizability Interpretation
  - $t \mathbf{r} (A \to^{\mathrm{c}} B) := \forall_x (x \mathbf{r} A \to tx \mathbf{r} B)$
  - $t \mathbf{r} (A \to^{\mathrm{nc}} B) := \forall_x (x \mathbf{r} A \to t \mathbf{r} B)$
  - $t \mathbf{r} \forall^{\mathrm{c}}_x A := \forall_x (tx \mathbf{r} A)$
  - $t \mathbf{r} \forall^{\mathrm{nc}}_x A := \forall_x (t \mathbf{r} A)$
- Extracted Term
  - $et((\lambda_u M)^{A \to^{\mathrm{c}} B}) := \lambda_{x_u} et(M)$
  - $et((\lambda_u M)^{A \to^{\mathrm{nc}} B}) := et(M)$
  - $et(I_i^+) := C_i$ (constructor)
  - $et(I^-) := \mathcal{R}$ (recursion operator)
  - $et({}^{co}I^-) := \mathcal{D}$ (destructor)
  - $et({}^{co}I^+) := {}^{\mathrm{co}}\mathcal{R}$ (corecursion operator)

(Soundness) Let M be a proof of formula A, $et(M) \mathbf{r} A$ holds.

Unfolding Corecursion Operator

- From our GFP axiom the following corecursion operator extracted

$$
\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{I}} &: (\tau \to \mathbb{U} + \tau + \tau + \tau) \to \tau \to \mathbb{I} \\
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{I}} MN &\mapsto [\lambda\_\mathbf{I}, \lambda_x(\mathbf{C}_{-1}({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{I}} Mx)), \\
&\qquad \lambda_x(\mathbf{C}_{0}({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{I}} Mx)), \lambda_x(\mathbf{C}_{1}({}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbb{I}} Mx))](MN)
\end{aligned}
$$

- Function $M^{\tau \to \mathbb{U}+\tau+\tau+\tau}$ determines which constructor should be output.
  1. If $(MN)^{\mathbb{U}+\tau+\tau+\tau}$ is the injection of $\mathbb{U}$, ${}^{\mathrm{co}}\mathcal{R}_{\mathbb{I}} MN \mapsto \mathbf{I}$
  2. If $(MN)^{\mathbb{U}+\tau+\tau+\tau}$ is the injection of some τ, ${}^{\mathrm{co}}\mathcal{R}_{\mathbb{I}} MN \mapsto \mathbf{C}_d({}^{\mathrm{co}}\mathcal{R}_{\mathbb{I}} MN')$ for the corresponding d

Extracted Translator

```
[algQ0]
 (CoRec algQ=>intv)algQ0
 ([algQ1]
   [if algQ1
     ([a2]
      [if (a2-(IntN 1#3))
        ([k3,p4]
         [if k3
           ([p5]
            [if (a2-(1#3))
              ([k6,p7] ...... )))))))))
```

Unfolding Corecursion Operator to Normalize

```
input> (pp
        (nt
         (undelay-delayed-corec
          (make-term-in-app-form translator
                                 (pt "CGenQ(IntN 1#3)"))
          5)))
;CIntN
;(CIntZ
; (CIntP
;  (CIntZ
;   (CIntP
;    ((CoRec algQ=>intv)(CGenQ(1#3))
;     ([algQ0]
;       [if algQ0 .......... ]))))))
```

Output is $-1 :: 0 :: 1 :: 0 :: 1 :: ....$, which we already saw.

Conclusion

- TCF and its implementation Minlog
  - Coinductive reasoning
  - Program extraction
- Two Case Studies on Program Extraction
  - Parsing Balanced Parentheses
  - Translating a rational number into a real number representation

Related Work

- Other Systems
  - Coq has a different program extraction [Coq][Let03]
  - Isabelle has a program extraction after Minlog [Isa]
  - Agda has an experimental program extraction [Agd][Chu11]
- Our Case Study
  - Cauchy Reals
  - Extracted Flip Function on $\mathbb{I}$, $f : x \mapsto -x$
  - Extracted Average Function on $\mathbb{I}$, $f : (x, y) \mapsto \frac{x+y}{2}$

Future Work

- Extracting Uniformly Continuous functions of $\mathbb{I}^n \rightarrow \mathbb{I}$ [BS10]
- Improving exact real arithmetic [BH08]

References

[Agd] Agda. http://wiki.portal.chalmers.se/agda/.

[BH08] U. Berger and T. Hou.
Coinduction for exact real number computation.
Theory of Computing Systems, 43:394–409, 2008.

[BS10] U. Berger and M. Seisenberger.
Proofs, programs, processes.
Programs, Proofs, Processes, CiE 2010, LNCS 6158, pp. 39–48, 2010.

[CDG06] A. Ciaffaglione and P. Di Gianantonio.
A certified, corecursive implementation of exact real numbers.
Theor. Comp. Sci., 351:39–51, 2006.

[Chu11] C. M. Chuang. *Extraction of Programs for Exact Real Number Computation Using Agda*.
PhD thesis, Swansea University, Wales, 2011.

References (cont.)

[Coq] The Coq Proof Assistant. `http://coq.inria.fr/`.

[Isa] Isabelle. `http://isabelle.in.tum.de/`.

[Let] P. Letouzey. A New Extraction for Coq. *Types for Proofs and Programs*, TYPES 2002, LNCS 2646, 2003.

[Min] The Minlog System. `http://www.minlog-system.de`.

[SW11] H. Schwichtenberg and S. S. Wainer. *Proofs and Computations*. Perspectives in Logic. Assoc. Symb. Logic and Cambridge Univ. Press, to appear, 2011.