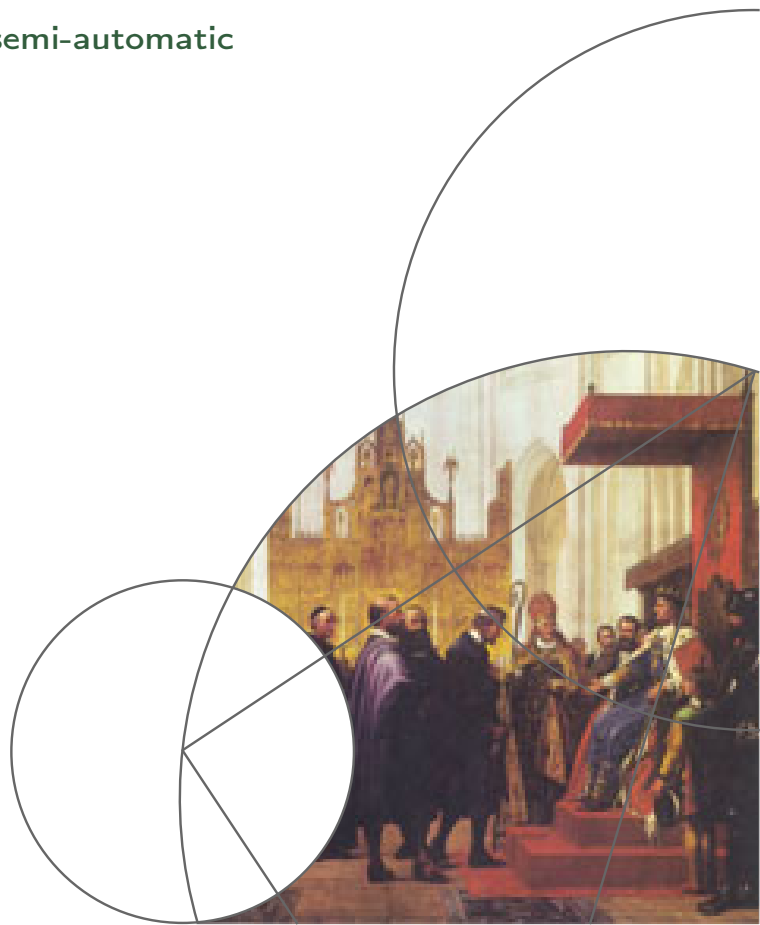
Turtledove

Tool assisted programming in SML, with special emphasis on semi-automatic rewriting to predefined standard forms.

Jesper Reenberg
Department of Computer Science



Program

- Oversigt over relateret arbejde
  - Live demo: (Meget) simpel omdøbning af funktionsnavne
- Afledning af inferensregel
- Konklusion

Oversigt – Relateret arbejde

Af stører (nutidige) projekter findes:

- HaRe (Haskell) – Nærmeste „Konkurrent“
- Værktøjer
  - Programatica (brugt i HaRe)
  - Strafunski (brugt i HaRe)
  - Stratego

Sammenligning – HaRe 1/3

HaRe:

- Kommentar og layout bevarelse
  - Haskell kan bruge layout regler
- Mulighed for brugerdefineret refaktorering og program transformationer.

Turtledove:

- Ingen layout bevarelse
  - Ikke strengt nødvendigt
  - Bruger en simpel pretty printer
- Forberedt for brugerdefinerede omskrivnings regler.

Sammenligning – HaRe 2/3

Refaktoreringer i HaRe (Ikke nødvendigvis begrænset til):

- Structural
  - Rename
  - Add/Remove argument to fun-definition
- Modul
  - Clean import lists
  - Move definition between modules
  - Add/Remove entity to module export list
- Data-Oriented
  - Concrete to abstract data type
    (hide the value constructors from the user)
  - Create discriminator functions
    (isLeaf, isNode, ...)
  - Create constructor functions
    (mkLeaf, mkNode ...)

Sammenligning – HaRe 3/3

Alle nævnte refaktoreringer fra HaRe kan implementeres som plug-ins i Turtledove.

Udviklede „Apps“ er eksempler på simple plug-ins.

Eksempel/Demo: Simpel omdøbning af funktions navne.

Programatica

> *„A Haskell front-end, with functionality similar to what you find in a compiler front-end for Haskell, implemented in Haskell.“*
>
> *[Programatica]*

Features:

- Lexer
- Parser
- Type checker
- Pretty printer
- etc.

Ikke brugbar fordi:

- Haskell parser
- Havde SML/MLB parser

Strafunski

„Strafunski is a Haskell-centred software bundle for implementing language processing components — most notably program analyses and transformation“
[Lämmel 03]

Features:

- Generisk travasering og tranformation af AST's
  - Via forskellige strategier

Ikke brugbar fordi:

- Alt arbejde skal udføres i Haskell
- (Umiddelbart) begrænset til omskrivninger

Stratego/XT

Stratego

- Sprog til program transformationer

Xt

- Toolset
- Parser, pretty print
- Transformator komponenter

Ikke brugbar fordi:

- Syntaks skal defineres så der kan genereres en parser.
- Vi havde allerede en parser.
- (Umiddelbart) begrænset til omskrivninger

Eksempel: Foldr

Hvis vi kikker på den inferens regel der „matcher“ en funktions clausul og regel clausul: $\boxed{\sigma \vdash clause : sclause}$.

Regel: Definition 21

$$\begin{array}{ll} \mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}] \Rightarrow \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}])) \\ \mathcal{D} \qquad\qquad \Rightarrow \mathbb{E}(\mathcal{D}) \\ \qquad \text{where samedom}(\mathcal{C}, \mathcal{D}) \end{array}$$

$$\Downarrow$$

$$\mathcal{C}[\texttt{xs}] \Rightarrow \texttt{foldr (fn (x, a) =>}\ \mathbb{D}(\mathcal{C}[\texttt{x}],\ \texttt{a}))\ (\mathbb{E}(\mathcal{C}[\texttt{xs}]))\ \texttt{xs}$$

Norm fun: Eksempel 19, uden curry, forkortet navn

```
fun cmplist (y :: ys, x) = y (cmplist (ys, x))
  | cmplist (y, x) = x
```

Dog kun første clausul, da anden clausul er ikke spændene.

Eksempel: Foldr 1/4

Lad $\sigma(C) = (\diamond_1,\ \overline{a})$:

$$\boxed{\sigma \vdash clause : sclause}$$

$$\frac{\overline{\sigma(spat) = mpat} \qquad \overline{pat : \langle mpat,\ \theta \rangle} \qquad \overline{\sigma,\ \theta \vdash sexp : exp}}{\sigma \vdash \texttt{(y :: ys, x) => y (cmplist (ys, x))} : \mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}] \Rightarrow \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}]))}$$

Eksempel: Foldr 2/4

Lad $\sigma(C) = (\diamond_1,\ \overline{a})$:

$$\boxed{\sigma(spat) = mpat}$$

$$\frac{}{\sigma(\mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}]) = (\diamond_1,\ \overline{a})[\overline{x}\ \texttt{::}\ \texttt{xs}/\diamond_1]}\ \sigma(C) = (\diamond_1,\ \overline{a})$$

$$\frac{\dfrac{}{\sigma(\mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}]) = (\overline{x}\ \ \texttt{::}\ \texttt{xs},\ \overline{a})} \qquad \dfrac{}{pat : \langle mpat,\ \theta\rangle} \qquad \dfrac{}{\sigma,\ \theta \vdash sexp : exp}}{\sigma \vdash \texttt{(y :: ys, x) => y (cmplist (ys, x))} : \mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}] \Rightarrow \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}]))}$$

Eksempel: Foldr 3/4

Lad $\sigma(C) = (\diamond_1,\ \overline{a})$:

$$\boxed{pat : \langle mpat,\ \theta \rangle}$$

$$\frac{\dfrac{}{\texttt{y :: ys} : \langle \overline{x}\ \texttt{::}\ \texttt{xs},\ \{\ \ \overline{x} \mapsto \texttt{y}\ \ \}\rangle} \qquad \dfrac{}{\texttt{x} : \langle \overline{a},\ \{\overline{a} \mapsto \texttt{x}\}\rangle}}{\texttt{(y :: ys, x)} : \left\langle (\overline{x}\ \texttt{::}\ \texttt{xs},\ \overline{a}),\ \theta = \left\{ \begin{array}{l} \overline{x} \mapsto \texttt{y} \\ \overline{a} \mapsto \texttt{x} \end{array} \right\} \right\rangle}$$

Bemærk $\texttt{xs} \mapsto \texttt{ys}$

$$\frac{\dfrac{\dfrac{}{\sigma(\mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}]) = (\overline{x}\ \ \texttt{::}\ \texttt{xs},\ \overline{a})} \qquad \dfrac{\vdots}{\texttt{(y :: ys, x)} : \langle (\overline{x}\ \texttt{::}\ \texttt{xs},\ \overline{a}),\ \theta\rangle}}{\sigma,\ \theta \vdash sexp : exp}}{\sigma \vdash \texttt{(y :: ys, x) => y (cmplist (ys, x))} : \mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}] \Rightarrow \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}]))}$$

Eksempel: Foldr 4/4

Lad $\sigma(C) = (\diamond_1,\ \overline{a})$:

$$\boxed{\sigma,\ \theta \vdash sexp : exp \equiv}$$

$$\sigma(\mathcal{C}) = (\diamond_1, \overline{a}) : \dagger$$
$$\sigma(\texttt{self}) = (\texttt{cmplist}, 1) : \ddagger$$
$$\sigma(\mathbb{D}) = (pat, exp) = (((\texttt{a},\_),\ \texttt{b}), \texttt{a b}) : \dagger\dagger$$

$$\dfrac{\dfrac{\dfrac{\vdots}{\sigma,\ \theta \vdash (\diamond_1, \overline{a})[\overline{x}/\diamond_1] : (\texttt{y},\ \texttt{x})}}{\sigma,\ \theta \vdash \mathcal{C}[\overline{x}] : (\texttt{y},\ \texttt{x})}\dagger \qquad \dfrac{\dfrac{\dfrac{\vdots}{\sigma,\ \theta \vdash (\diamond_1, \overline{a})[\texttt{xs}/\diamond_1] : (\texttt{ys},\ \texttt{x})}}{\sigma,\ \theta \vdash \mathcal{C}[\texttt{xs}] : (\texttt{ys},\ \texttt{x})}\dagger}{\sigma,\ \theta \vdash \texttt{self}(\mathcal{C}[\texttt{xs}]) : (\texttt{cmplist}\ x_1)[(\texttt{ys},\ \texttt{x})/x_1]}\ddagger}{\sigma,\ \theta \vdash \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}])) : \texttt{a b}[((y, x), \texttt{cmplist (ys, x)})/((\texttt{a},\_),\ \texttt{b})]}\dagger\dagger$$

$$\dfrac{\dfrac{\dfrac{}{\sigma(\mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}]) = (\overline{x}\ \ \texttt{::}\ \texttt{xs},\ \overline{a})} \qquad \dfrac{\dfrac{\vdots}{(\texttt{y :: ys, x}) : \langle(\overline{x}\ \texttt{::}\ \texttt{xs},\ \overline{a}),\ \theta\rangle}}{\vdots}}{\sigma,\ \theta \vdash \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}])) : \texttt{y (cmplist (ys, x))}}}{\sigma \vdash \texttt{(y :: ys, x) => y (cmplist (ys, x))} : \mathcal{C}[\overline{x}\ \texttt{::}\ \texttt{xs}] \Rightarrow \mathbb{D}(\mathcal{C}[\overline{x}],\ \texttt{self}(\mathcal{C}[\texttt{xs}]))}$$

Fejl i normalform

Eksempel: (foo [1,2,3])

```
fun foo (x :: y :: ys) = x + 41 :: foo ( y :: ys)
  | foo _ = nil
```

Smider sidste element væk.
Normaliseres til

```
fun foo (x :: c) = x + 41 :: foo c
  | foo x = nil
```

Smider IKKE sidste element væk.

`y :: ys` generaliseres. Fix: Skal kun generaliseres når alle konstruktørene er representeret (her mangler tilfældet nil).

Første eksempel er normalform i sig selv. Dette er fint eksempel på hvorfor map reglerne kræver en konkret variabel `xs` og ikke $\overline{xs}$ da $\overline{xs}$ ville matche `y :: ys` og ovenstående ville blive omskrevet til en map.

Konklusion

- Eksisterende miljøer/frameworks kan ikke tilbyde samme funktionalitet
- Der eksisterer ikke nogle værktøjer til SML
  - Dokumentations væktøjer undtaget.
- Simple væktøjer (se implementerede apps) gør det mere „behageligt“ og „effektivt“ at kode SML.
- Turtledove er platforms uafhængigt.

Bibliografi

R. Lämmel & J. Visser.
A Strafunski Application Letter.
In V. Dahl & P. Wadler, editeurs, Proc. of Practical Aspects of Declarative Programming (PADL'03), volume 2562 of *LNCS*, pages 357–375. Springer-Verlag, January 2003.

Programatica.
Features of the programatica haskell tools [online].
Available from: `http://ogi.altocumulus.org/~hallgren/Programatica/tools/features.html` [cited 21 March 2011].