modules/Marstrand 86

Marstrand Workshop on
Specification and Derivation
of Programs
June 10-12, 1986
Marstrand, Sweden

DEPENDENT TYPES AND MODULAR STRUCTURE

DAVID MACQUEEN
AT&T BELL LABORATORIES

GOAL: USE TYPES TO EXPRESS THE LARGE-SCALE STRUCTURE OF PROGRAMS

MAIN POINTS:

* Mitchell-Plotkin existential types won't do.
* <u>Simple</u> <u>types</u> and <u>interpreted</u> <u>types</u>
* Expressing type abstraction
  - opaque vs transparent structures
* Interaction of <u>dependency</u> and <u>abstraction</u>
* A stratified dependent type system

EXISTENTIAL TYPES

Start with 2^{nd}-order λ-calculus

$\exists t.\sigma(t)$ (a signature)

typical element:

$\langle \tau, e \rangle$

data algebra
package
structure

"interpretation" of τ

representation type

existential introduction rule:

$$\frac{e : \sigma(\tau)}{\langle \tau, e \rangle : \exists t.\sigma(t)}$$

Example: Complex Numbers

$$\text{COMPLEX} = \exists \text{complex}. \underbrace{\langle i : \text{complex}, \; \text{one} : \text{complex}, \; \text{plus} : \text{complex}^2 \to \text{complex} \rangle}_{\sigma_{\text{COMPLEX}}(\text{complex})}$$

$$\text{Cart} : \text{COMPLEX} =$$

$$\langle \text{real} \times \text{real}, \; \sigma_{\text{COMPLEX}}(\text{real} \times \text{real}) \begin{cases} \langle i = (0.0, 1.0), \\ \text{one} = (1.0, 0.0), \\ \text{plus} = \lambda (r_1, i_1), (r_2, i_2). (r_1 + r_2, i_1 + i_2) \rangle \end{cases} \rangle$$

$$\text{Polar} : \text{COMPLEX} =$$

$$\langle \text{real} \times \text{real}, \; \sigma_{\text{COMPLEX}}(\text{real} \times \text{real}) \begin{cases} \langle i = (1.0, \pi/2.0), \\ \text{one} = (1.0, 0.0), \\ \text{plus} = \lambda (m_1, \theta_1), (m_2, \theta_2). \ \ldots \rangle \end{cases} \rangle$$

USE OF EXISTENTIAL STRUCTURES

$$\underline{\text{abstype}}\ t\ \underline{\text{with}}\ x = \underbrace{e_1}_{\text{implementing structure}}\ \underline{\text{in}}\ \underbrace{e_2}_{\text{client}}$$

Existential elimination rule:

$$\frac{\mathcal{A} \vdash e_1 : \exists t.\sigma(t) \qquad \mathcal{A}[\sigma(t)/x] \vdash e_2 : \rho}{\mathcal{A} \vdash (\underline{\text{abstype}}\ t\ \underline{\text{with}}\ x = e_1\ \underline{\text{in}}\ e_2) : \rho}$$

assuming: $\underline{t \text{ not free in } \rho}$ (opaqueness)
nor any $\mathcal{A}(y)$ *for* $y \neq x$ *free in* e_2

SEMANTIC VIEW

$$[\![\exists t.\sigma(t)]\!] = \bigcup_{A \in \mathcal{Id}} [\![\sigma(A)]\!].$$

$$v \in [\![\exists t.\sigma(t)]\!] \Rightarrow v \in [\![\sigma(A)]\!] \quad \text{some } A \in \mathcal{Id}$$

but A is not uniquely determined.

PROBLEMS WITH OPAQUE EXISTENTIAL STRUCTURES

$$\text{ORDER} = \exists s.\ s \times s \rightarrow \text{bool}$$

$$\text{IntOrd} = \langle \text{int}, < \rangle_{\text{ORDER}} \quad \text{(a "useless" structure)}$$

$$\text{Lexord} = \lambda O : \text{ORDER}.$$
$$\underline{\text{abstype}}\ s\ \underline{\text{with}}\ lt = O\ \underline{\text{in}}$$
$$\langle s\ \text{list}, \lambda l_1, l_2 : s\ \text{list}.\ \ldots \rangle_{\text{ORDER}}$$

$$\text{Lexord} : \text{ORDER} \longrightarrow \text{ORDER} \quad \text{(a "useless" mapping)}$$

$$\text{POINT} = \exists\text{point}.\langle \text{mkpoint}: \text{int} \times \text{int} \to \text{point},$$
$$\text{x\_coord}: \text{point} \to \text{int},$$
$$\text{y-coord}: \text{point} \to \text{int},$$
$$\text{trans}: \text{point} \times \text{int} \times \text{int} \to \text{point}\rangle$$

CircleWRT(P:POINT) =

abstype p with point_ops = P in

$$\langle p, \langle p \times \text{int}, \text{"circle\_ops"}\rangle_{\text{CIRCLE}'(p)}\rangle_{\text{CIRCLE}}$$

RectWRT(P:POINT) = ...

$$\left.\begin{array}{l} C = \text{CircleWRT(Point)} \\ R = \text{RectWRT(Point)} \end{array}\right\} \text{No interaction!}$$

abstype p with point_ops = Point in
  let C = CircleWRT*[p](point_ops)
  and R = RectWRT*[p](point_ops)
  in abstype c with circle_ops = C
   in abstype r with rect_ops = R in
     ...

MAKING COMPONENTS ACCESSIBLE

$(\exists s.\ s \times s \rightarrow \text{bool}) \Rightarrow$ **sig**

    type s
    val lt : $s \times s \rightarrow$ bool

end

```
structure IntOrd =
    struct
        type s = int
        val lt = (<) : int x int -> bool
    end
```

IntOrd : ORDER

$\left.\begin{array}{l}\text{IntOrd.s} = \text{int} \\ \\ \text{IntOrd.lt} : \text{int} \times \text{int} \rightarrow \text{bool}\end{array}\right\}$ transparency

IntOrd.lt (5+1, hd [7,3,4])

DEPENDENCE: Σ-CLOSURE

```
signature POINT =
   sig
     type point
     val  mkpoint : int x int -> point
     val  x_coord : point -> int
     ...

structure Point : POINT = ...

signature CIRCLE =
  sig

    type circle
    val  mkcircle :  point x int -> circle
    val  center : circle ->   point
    ...
  end

structure Circle : CIRCLE =
  struct
    structure P = Point
    type circle = P.point x int
    val  mkcircle (p, n) = (p, n)
    ...
```

DEPENDENCE: Σ-CLOSURE

```
signature POINT =
   sig
     type point
     val  mkpoint : int x int -> point
     val  x_coord : point -> int
     ...

structure Point : POINT = ...

signature CIRCLE =
  sig
    structure P: POINT
    type circle
    val  mkcircle : P.point x int -> circle
    val  center : circle -> P.point
    ...
  end
```

$\equiv \quad \Sigma P{:}\text{POINT}.\ \Sigma \text{circle}{:}\text{TYPE}.\ [\ \cdots\ ]$

```
structure Circle : CIRCLE =
  struct
    structure P = Point
    type circle = P.point x int
    val  mkcircle (p, n) = (p, n)
    ...
```

ABSTRACTION

```
functor MkCircle (P': POINT) : CIRCLE =
    struct
        structure P = P'
        type circle = P.point x int
        ...
    end
```

MkCircle : $\Pi P' : \text{POINT}.\ \text{CIRCLE}\{P = P'\}$

The parameter P′ is <u>opaque</u>, therefore <u>abstract</u>. {within the body of the functor}

SHARING

```
functor MkFigure (C : CIRCLE, R : RECT
                         sharing C.P = R.P) : FIGURE
    = struct ...
```

MkFigure : $\Pi C : \text{CIRCLE} \times R : \text{RECT}\{C.P = R.P\}.\ \text{FIGURE}$

STRATIFIED SOL

$$\sigma ::= \mathrm{int} \mid \mathrm{bool} \mid \sigma * \sigma \mid \sigma \to \sigma \mid \qquad \text{small } (\mathcal{U}_1)$$
$$\exists t.\sigma(t) \mid \forall t.\sigma(t) \qquad \text{large } (\mathcal{U}_2)$$

(t ranging over small types only, $t : \mathcal{U}_1$)

Restrictions:

type application: $e[\sigma]$ - σ must be small

existential pairing: $\langle \sigma, e \rangle$ - σ must be small

Structures are small:

$s \in \exists t.\sigma(t)$
$\exists t.\sigma(t) \in \mathcal{U}_2$ (quantification over $\mathcal{U}_1$)
$s \in \mathcal{U}_1$ (particular type: $s = \langle \tau, e \rangle$)

? Functors are large

$F \in \forall t.\sigma(t) \in \mathcal{U}_2$ (quantification over $\mathcal{U}_1$)
$F \in \mathcal{U}_1$ (abstraction over $\mathcal{U}_1$)

Level analogies

structure	~	object	~	set
signature	~	category	~	class (of sets)
"functor"	~	functor	~	class (operation over sets)