Various

Contents

A short survey of chapters 16, 13, 14 and 17:

- Underspecification and Non-determinism 2
- Case Expressions and Patterns 5
- Let Expressions 11
- Overloading 14

Otherwise self-study.

Underspecification

versus

Non-determinism

Under Specification

Under specified means: has more than 1 model.

Example:

value x : **Int**
axiom $x = 1 \vee x = 2 \vee x = 3$

Non-deterministic Expressions

$0 \sqcap 1 \sqcap 2$
let x : **Nat** • x < 3 **in** x **end**

$(0 \sqcap 1 \sqcap 2 \equiv 0 \sqcap 1 \sqcap 2) \equiv$ **true**

$(0 \sqcap 1 \sqcap 2 = 0 \sqcap 1 \sqcap 2) \equiv ($**true** $\sqcap$ **false**$)$

Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Case Expressions and Patterns

Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Case Expressions

Similar to ML's case expressions.

Example:

value
  rev : $\mathbf{Int}^* \to \mathbf{Int}^*$
  rev(l) $\equiv$
    case l **of**
      $\langle\rangle \to \langle\rangle$,
      $\langle i\rangle \mathbin{^\frown} l1 \to rev(l1) \mathbin{^\frown} \langle i\rangle$
    end

General form:

case value_expr **of**
  $\mathrm{pattern}_1 \to \mathrm{value\_expr}_1$,
  $\vdots$
  $\mathrm{pattern}_n \to \mathrm{value\_expr}_n$
end

Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Patterns

- literal patterns
- wildcard patterns
- product patterns
- list patterns
- name patterns
- **record patterns**
- equality patterns

matching value against pattern:

- successful or not
- if successful: may $\Rightarrow$ definitions

Technical University of Denmark
Informatics and Mathematical Modelling
Computer Science and Technology

Pitfalls of case expressions

- no successful pattern matching
- more than one successful pattern matching (implies non determinism) (can only happen for record patterns)

Advices to avoid this:

- ensure exhaustive cases (e.g. by using the _ case)
- never make case of a variant type value, if the variant type has records without destructors

Record Patterns

type
  List == empty | add(head : Colour, tail : List)
value
  invert_list : List → List
  invert_list(l) ≡
    case l **of**
      empty → empty,
      add(c,l1) → add(invert(c),invert_list(l1))
    end

v matches

id(inner_pattern$_1$,...,inner_pattern$_n$)

iff $\exists$ (v_1,...,v_n) • id(v_1,...,v_n) = v
and v_i matches inner_pattern$_i$

Exercise

Evaluate invert_list(add(white, add(black, empty)))

Record Patterns

type
  Set == empty | add(Colour, Set)
axiom
  [unordered]
    ∀ e1,e2 : Colour, s : Set •
      add(e1,add(e2,s)) ≡ add(e2,add(e1,s)),
  [no_duplicates]
    ∀ e : Colour, s : Set • add(e,add(e,s)) ≡ add(e,s)
value
  choose : Set $\xrightarrow{\sim}$ Colour
  choose(s) ≡
    case s **of**
      add(c,s1) → c
    end
  pre s ≠ empty

Exercise

Evaluate choose(add(white, add(black, empty)))

Let Expressions

Let Expressions

- explicit let expressions, e.g.
  let x = 2 **in** x + 5 **end**
- implicit let expressions, e.g.
  let x : **Nat** • x < 3 **in** x **end**

Overloading

Overloading

Survey

- *Overloading* means that the same name has multiple declarations.
- Overloading of Built-in Operators:

  $+ : \mathbf{Int} \times \mathbf{Int} \to \mathbf{Int},$
  $+ : \mathbf{Real} \times \mathbf{Real} \to \mathbf{Real}$

- Overloading of User-defined Identifiers:

  value
  sel_flight : Landing → Flight,
  sel_flight : Take_Off → Flight

- User-defined Infix Operators: + : Table × Table → Table
- User-defined Prefix Operators: **elems** : Tree → Elem-**set**
- Type checking in presence of overloading:
  - Compatibility of Value Definitions
  - Overload Resolution
- New kinds of value expressions:
  - Type Disambiguation: value_expr : type_expr
  - Turning User-defined Operators into Expressions

Compatibility of Value Definitions

Value definitions must be *compatible*:

- they must introduce distinct names, or
- the maximal types of the names must be distinguishable

Example

value
max : **Nat** × **Nat** → **Nat**,
max : **Int** × **Int** → **Int**

are not compatible

max(7,3)

Overload Resolution

Requirement:

For each applied occurence of a name (identifier or operator) overloading must be resolvable, i.e. there must be one and only one declaration of that name that makes the specification type correct.

Type Disambiguation

value
empty : **Int-set**,
is_empty : **Int-set** → **Bool**
empty : $\mathbf{Int}^*$,
is_empty : $\mathbf{Int}^*$ → **Bool**

Overloading is resolvable in:

$\text{empty} \cup \{1\} = \{1\}$
$\text{empty} \frown \langle 1 \rangle = \langle 1 \rangle$

Overloading is not resolvable in:

axiom
is_empty(empty)

Overloading is resolvable in:

axiom
is_empty(empty : **Int-set**),
is_empty(empty : $\mathbf{Int}^*$)

General form of type disambiguation expr:

value_expr : type_expr