CS538 Lecture 16: ML Exceptions

April 1997

James R. Larus
Computer Sciences Department
University of Wisconsin–Madison
larus@cs.wisc.edu

Outline:

- Predefined exceptions
- User-defined exceptions

Built-in Exceptions

- ML predefines many exceptions

```
 1.0 / 0.0;
uncaught exception Div

hd(nil);
uncaught exception Hd
```

- Essential to detect and report errors in safe language
- Method to handle errors

```
1.0 / 0.0 handle Div => 0.0;
val it = 0.0 : real

hd(nil) handle Hd => nil;
val it = [] : 'a list
```

- Mechanism extensible for program errors

SML Exceptions

- System Level
  - Io: I/O error
  - Bind: Pattern failed to match expression in binding
  - Match: Pattern failed to match expression in application
- Arithmetic
  - Div: Divide by zero
  - Overflow: Value too large
  - Sqrt, Exp, Ln: Domain error

SML Exceptions, cont'd

- Character & String
  - Chr: **`chr`** function given value out of range 0..255
  - Ord: **`ord`** function given empty string
  - Substring: **`substring`** function given range outside of string
  - Range: **`bytearray`** function given bad argument
- List & Array
  - Hd, Tl: **`hd`** or **`tl`** function applied to empty list
  - Nth, NthTail: **`nth`**, **`nthtail`** applied to short lists
  - Subscript: Array subscript out of bounds

Exception Handlers

- Catch exceptions raised in evaluating expression

```
<expression> handle <match>
```

  - Use pattern matching to specify exceptions

```
<match> → <pattern> => <expression> ...
```

- If no exception, pattern not matched
  - If no match, exception propagates
- Expression types must be compatible

```
 1.0 / 0.0 handle Div => "Div by zero!";
std_in:4.1-4.38 Error: expression and handler don't agree
(tycon mismatch)
  body:           real
  handler range:  string
  in expression:
    / (1.0,0.0)
    handle
        Div => "Div by zero!"
```


Modified: April 9, 1997

User-Defined Exceptions

- Program can define new exceptions

```
exception <identifier>
exception <identifier> of <type>
```

  - Creates constructor for exception

```
exception BadValue of int;
exception BadValue of int

- raise BadValue(1);
uncaught exception BadValue
```

- Program can raise exceptions

```
raise <identifier>
raise <identifier>( <expression> )
```

  - Type is exn, which is compatible with all types

```
if 1 > 2 then 4.0 else raise Div;
uncaught exception Div
```


Modified: April 9, 1997

Exception Scope

- Scope of exception identifier is lexical
  - Handler is bound dynamically
  - Handler must be within lexical scope of exception declaration

```
fun comb2(n,m) =
  let
    exception OutOfRange of int*int
  in
    if n <= 0 then raise OutOfRange(n,m)
    else if m<0 orelse m>n then raise OutOfRange(n,m)
    else if m=0 orelse m=n then 1
    else comb2(n-1,m)+comb2(n-1,m-1)
  end;
```

Exception scope is limited to this block

Cannot catch exception

```
fun comb(n,m) = comb2(n,m) handle
  OutOfRange(0,0) => 1
| OutOFRange(n,m) => (print(“out of range”); 0);
```