§10.13–11.3: FracExceptions example, Backtracking

17 Nov 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals in folder***

Review of last time (10.12)

- Exceptions: another level of error handling
  - Raise/handle (a.k.a. throw/catch)
  - EXCEPT clause:
    - Do nothing, RETURN, RETRY
  - Built-in exceptions: M2EXCEPTION:
    - M2Exceptions, IsM2Exception(), M2Exception()
  - Standard library exceptions: e.g., IOChan:
    - ChanExceptions, IsChanException(),ChanException()
  - User defined exceptions: how to raise/handle
  - Exceptions and termination

Fractions example (10.13)

- Making our own ADT as a library module:
  - DEFINITION MODULE Fractions;

```
TYPE
    Fraction = ARRAY [1..2] OF INTEGER;
    FracExceptions =
        (zeroDenominator, noInverse, zeroDivide);
PROCEDURE Add, Sub, Mul, Div, etc...
```

  - IMPLEMENTATION MODULE Fractions;

```
VAR
    fracExSource : ExceptionSource;
PROCEDURE Add, Sub, Mul, Div, etc...
    RAISE (fracExSource, ORD(noInverse) "..."); .....
```

Encapsulation

- We want to restrict users of our Fractions ADT to access the ADT only via our procedures
  - Means we need to provide enough procedures for our ADT to be useful
- Exceptions raised by our procedures can be handled by modules that use our ADT

FracExceptions

- Define enumeration type:

```
TYPE FracExceptions =
        (zeroDenominator, noInverse, zeroDivide);
```

- Library procedures can raise exceptions:

```
RAISE (fracExSource, ORD (zeroDivide),
    "Cannot divide by zero");
```

- Allocate exception source in library initialization:

```
BEGIN (* initialize *)
   AllocateSource (fracExSource);
```

- Handle exceptions in termination clause:

```
FINALLY
    IF IsFracException ()
```

Backtracking: recursion appl.

- Knight's tour classic chess problem:
  - Find a sequence of legal knight moves that touches every square of the board once
    - Input: size of board, starting position
    - Output: sequence of board coordinates (x,y)
- Algorithm:
  - Find possible moves from current position
    - Omit squares we've already touched
  - For each move, take the move and recurse
  - If no possible moves, return (backtrack)

TODO items

- HW due tomorrow: 10.15 #53
- Quiz ch10 tomorrow
- Reading: through §11.3 for tomorrow
- Lab #9 next week: 10.15 #(44 / 49)
- Midterm ch8–10 Wed 23Nov (next week)