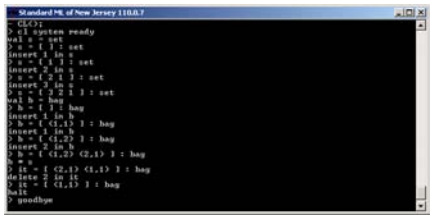
Exercise Session Informatik III

4. CL Interpreter

Demo



globis

Exercises

- Implement functions to perform union, intersection and minus over sets and bags
- Add exception handling to the program
- Implement a function to delete a binding in the store

globis

Overview

["a", "b", "+", "c", "d"]

Lexical Analyser

[TokIDENT("ab"), TokOP(Union), TokIDENT("cd")]

Parser

OPexpr(Union, IDENTexpr("ab"), IDENTexpr("cd")

Evaluator

val it = union(SetColl("ab", set1), SetColl("cd", set2))

globis

Operations

- What is already done?
  - Bindings are retrieved from the store
  - Recursive expressions are handled correctly
  - Conversions between bindings are performed
- What needs to be done?
  - For each type and each operator a function must be implemented that performs the operation

globis

But how?

- Search for the section **"Implement these Functions"** in the code
- A dummy implementation is given for all functions that are needed to evaluate expressions
- Implement these functions meaningful
- **Note:** You will need to define additional support functions to implement the operations

globis

Example

Instead of

```
fun setUnion lst1 lst2 = lst1 @ lst2
```

implement

```
fun setUnion set nil = set
|   setUnion set (hd::tl) =
       if setContains hd set then setUnion set tl
                             else hd::(setUnion set tl)
```

Support Functions

- As metioned there will be some "helper" functions needed
- Typically these will be used to perform one of the following tasks
  - check if an element is in a set
  - check if an element is in a bag
  - count the occurences of an element in a bag
  - remove all occurences of an element in a bag

Exception Handling

- There is **already some exception handling** in the interpreter function
- Exception should be handled at an **high level** in the program
- Hence the interpreter function is a **good spot** to insert exception handling
- Exceptions that are raised inside the interpreter are declared at the **beginning of the program**

Example

```
fun interpreter(S) =
   ( case evaluate (parser()) S
      handle
         NotImplemented =>
            ( print("> ERROR: sorry, not implemented\n");
              (false, S))
      |  Overflow =>
            ( print("> ERROR: number too big\n");
              (false, S))
      |  NoSuchIdent(s) =>
            ( print("> ERROR: ident "); print(s);
              print(" unknown\n"); (false, S))
      of
         (true,S)   => print("> goodbye\n\n")
      |  (false,S') => interpreter(S')
   )
```

Clearing of Bindings

- The store is organized as a list of bindings

  `SetColl("a",[1,3])::BagColl("b",[(1,2),(2,5)])::  ...`
- To clear a binding from the store we traverse this list an remove the collection that matches to the given name

```
fun clear ident nil = nil
|   cleat ident ...
```

- **Note:** Compare with function **`setDelete`** to see how a given element can be removed from a list

But where?

```
fun evaluate (HALTexpr) S = (true,S)
|   evaluate (DECLexpr(ident,bulk)) S =
       if check ident S then
          ( print("> "); print(ident); print(" = [ ]\n");
            (false, (create ident bulk)::S) )
       else raise IllegalIdent(ident)
|   evaluate (INSERTexpr(num, ident)) S =
       (false, printIdent ident (insert num ident S))
|   evaluate (DELETEexpr(num, ident)) S =
       (false, printIdent ident (delete num ident S))
|   evaluate (CLEARexpr(ident)) S =
       ( print("> clearing "); print(ident); print("\n");
         (false, clear ident S) )
|   evaluate (IDENTexpr(ident)) S =
       (false, printIdent ident S)
|   evaluate (OPexpr(operation, exp1, exp2)) S =
       ( printBulk(evaluateOp (OPexpr(operation, exp1,
         exp2)) S); (false, S) )
|   evaluate (_) S = raise NotImplemented
```

That's all folks!

Nächste Woche
- Abgabe des CL Interpreters
- Goodbye SML
- Einführung in Prolog

globis