

The Syntax of MiniEssence

Alan M. Frisch Matthew Grum Chris Jefferson
Bernadette Martínez Hernández Ian Miguel

February 22, 2005

We are developing a language, called ESSENCE, for abstractly specifying combinatorial problems and combinatorial optimisation problems. We are developing the language incrementally, adding features as we go along. This document specifies the syntax of MiniESSENCE, our initial version of ESSENCE.

Notation:

- An *identifier* is a string whose first element is a letter and the rest are alphanumeric characters, “_” or “’”. Identifier recognition is case sensitive.
- A *number* is any string whose elements are the numeric characters.
- $\{a\}$ stands for for a non-empty list of a 's.
- $\{a\}'$ stands for a non-empty list of a 's separated by commas.
- $[a]$ stands for nil or one occurrences of a .

And additional feature that it is not specified in the grammar is the possibility of adding comments to the specification file. Comment lines are preceded by two consecutive “-”.

1 Model

```
model ::= [ { declaration } ]  
          find { typedIdentifier }'  
          [ objective ]  
          [ such that { expression } ]'  
declaration ::= given { typedIdentifier }' |  
                  letting { constant }'  
objective ::= maximising expression |  
                  minimising expression  
typedIdentifier ::= identifier “:” type  
constant ::= identifier be type |  
                  typedIdentifier be expression
```

2 Types

```

type ::= "(" type ")" | identifier |
        bool | int |
        int "(" rangeId "." rangeId ")" |
        set [setDomain] of type |
        mset [setDomain] of type |
rangeId ::= number | identifier
setDomain ::= "(" size atom ")" | "(" maxsize atom ")"

```

3 Expressions

```

expression ::= "(" expression ")" |
               atom |
               true | false |
               "-" expression | "|" expression "|" | "¬" expression |
               expression biOp expression |
               "{" [{"expression"}] "}" |
               "[" [{"expression"}] "]" |
               quantifier bindingExpression "." expression
atom ::= number | identifier [":" type]
biOp ::= "+" | "-" | "/" | "*" | "^" |
           "∧" | "∨" | "←" | "↔" |
           "=" | "≠" |
           "≤" | "<" | "≥" | ">" |
           "∩" | "∪" |
           setBindOp
setBindOp ::= "⊆" | "⊂" | "∈"
quantifier ::= "∀" | "∃" | "Σ"
bindingExpression ::= typedIdentifier |
                       bindUnit setBindOp identifier
bindUnit ::= identifier [":" type] |
               "{" [{"rangeId"}] "}" |
               "[" [{"rangeId"}] "]"

```

4 Precedence

The operators have the precedence described in the following table. They are organized in a decreasing order of precedence.

Operator	Associativity
"-" (Unary)	Right
"^"	Left
"*", "/"	Left
"+", "-"	Left
">", "<", "≥", "≤"	None
"∩"	Left
"∪"	Left
"."	Right
"¬"	Right
"∧"	Left
"∨"	Left
"←"	Left
"↔"	Left
"=", "≠"	Left