The Syntax of Essence

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

May 12, 2006

We are developing the language Essence to formulate abstract specifications of combinatorial (optimisation) problems. The development of the language is incremental, we add features as we go along. This document specifies the syntax of Essence Version 1 (EV1), 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 as.
- $\{a\}$**'** stands for a non-empty list of as separated by commas.
- $\{a\}$***** stands for a non-empty list of as separated by the symbol '*'.
- $[a]$ stands for nil or one occurrences of a.

An additional feature that it is not specified in the grammar is that comment lines are preceded "$".

1 Model

model	::=	[{*declaration*}] [*objective*] [`such that` {*expression*}']
declaration	::=	`given` {*parameter*}' | `where` {*expression*}' | `letting` {*constant*}' | `find` {*variable*}'
objective	::=	`maximising` *expression* | `minimising` *expression*
domainIdentifiers	::=	{*identifier*}' ":" *domain*
constant	::=	*identifier* `be domain` *domain* | *identifier* [":" *domain*] `be` *expression* | *identifier* `be new type` *type*
parameter	::=	*domainIdentifiers* |`enum` "(...)"
variable	::=	*domainIdentifiers*

2 Domains

domain	::=	"(" *domain* ")" | `bool` | `int` ["(" {*rangeAtom*}' ")"] | *identifier* ["(" {*rangeAtom*}' ")"] | `set` [*sizeSet*] `of` *domain* | `mset` [*sizeSet*] `of` *domain* | `matrix indexed by` "[" {*domain*}' "]" `of` *domain* | *domain* "→" ["(" [*kindFunc*] [*classFunc*] ")"] *domain* | `rel` [*sizeSet*] *domainRel* | `partition` [*sizeSet*] *domain* | `rpartition` [*sizeSet*] *domain*
rangeAtom	::=	*expression* | *expression* ".." *expression* | "..." *expression* | *expression* "..."
sizeSet	::=	"(" `size` *expression* ")" | "(" `maxsize` *expression* ")"
kindFunc	::=	`partial` | `total`
classFunc	::=	`injective` | `surjective` | `bijective`
domainRel	::=	*domain* [[*multiplicity*] "*" {[*multiplicity*] *domain* [*multiplicity*]}*]
multiplicity	::=	"(" *expression* ")"

3 Types

type	::=	`of size` *expression* | `enum` "{" {*identifier*}' "}"

4 Expressions

expression	::=	"(" *expression* ")" |
		atomicExpression |
		nonAtomicExpression |
		groupOpExpression |
		unitOpExpression |
		binaryOpExpression |
		functionOpExpression |
		quantifierOpExpression
atomExpression	::=	*number* | `true` | `false` | *identifier* |
		"{" "}" ":" *domain* |
		"#" "#" ":" *domain*
nonAtomicExpression	::=	*expression* "[" {*expression*}' "]" |
		expression "(" {*projExpression*}' ")"
projExpression	::=	*expression* | "_"
groupOpExpression	::=	"{" {*expression*}' "}" |
		"#" {*expression*}' "#" |
		"<" {*expression*}' ">" |
		"[" {*expression*}' "]" [":" *domain*]
unitOpExpression	::=	"-" *expression* |
		"|" *expression* "|" |
		"¬" *expression*
binaryOpExpression	::=	*expression biOp expression*
biOp	::=	"+" | "−" | "/" | "∗" | "^" |
		"∧" | "∨" | "⇒" | "⇔" |
		"=" | "≠" |
		"≤" | "<" | "≥" | ">" |
		"∩" | "∪" | *subsetOp* | *memberOp*
subsetOp	::=	"⊂" | "⊆"
memberOp	::=	"∈" | "∉"
functionOpExpression	::=	`allDiff` "(" *expression* ")" | `inv` "(" *expression* ")" |
		`min` "(" *expression* ")" | `max` "(" *expression* ")"
		`dom` "(" *expression* ")" | `ran` "(" *expression* ")" |
		`freq` "(" *expression* "," *expression* ")" |
		`image` "(" *expression* "," *expression* ")" |
		`atleast` "(" *expression* "," *expression* "," *expression* ")" |
		`atmost` "(" *expression* "," *expression* "," *expression* ")"
quantifierOpExpression	::=	*quantifier bindingExpression* "." *expression*
quantifier	::=	"∀" | "∃" | "Σ"
bindingExpression	::=	{*identifier*}' ":" *domain* |
		bindUnit memberOp expression
		identifierList subsetOp expression
bindUnit	::=	*identifierList* | *tupleList*
identifierList	::=	{*identifier*}' [":" *domain*]
tupleList	::=	{"<"{*identifier*}'">"}' [":" *domain*]