CSE4312F12 Project Solution
ROI

Damien Gruel (cse23089@cse.yorku.ca)
Ludovic Lavalette (cse23088@cse.yorku.ca)

November 19, 2012

Note

- A customer elicitation session was held during class on Tuesday November 6, 2012. If you were not there sure to catch up with a fellow student who was there.
- This template is handed out *caveat emptor*. There may be errors and wrong information. It is ultimately your responsibility to elicit the correct requirements from the customer and to ensure that you satisfy the customer goals and specify correct output from the input.
- Your are required to correct any errors or ambiguities in this template and use this template to produce your final requirements document.

Revisions

Date	Revision	Description
10 October 2012	1.0	Initial customer elicitation
15 November 2012	2.0	Initial Student solution
1 December 2012	3.0	Final Student solution

Contents

List of Figures

List of Tables

1 Context Diagram

The following diagram is the context diagram for the ROI system.
The only monitored variable is the CSV file (provided by the user), which contains the evaluation dates (*start* and *end*).

The format of the output is the following (whole input = everything between the earliest date and the latest date in the sequence of tuple data):

Whole input: yyyy-mm-dd to yyyy-mm-dd
TWR: ?? %
ROI: ?? %
Benchmark: ?? %

Evaluation Period: yyyy-mm-dd to yyyy-mm-dd
TWR: ?? %
ROI: ?? %
Benchmark: ?? %

The controlled variables are also a warning (if a calculation is not possible) and an error (if the CSV file is not valid).

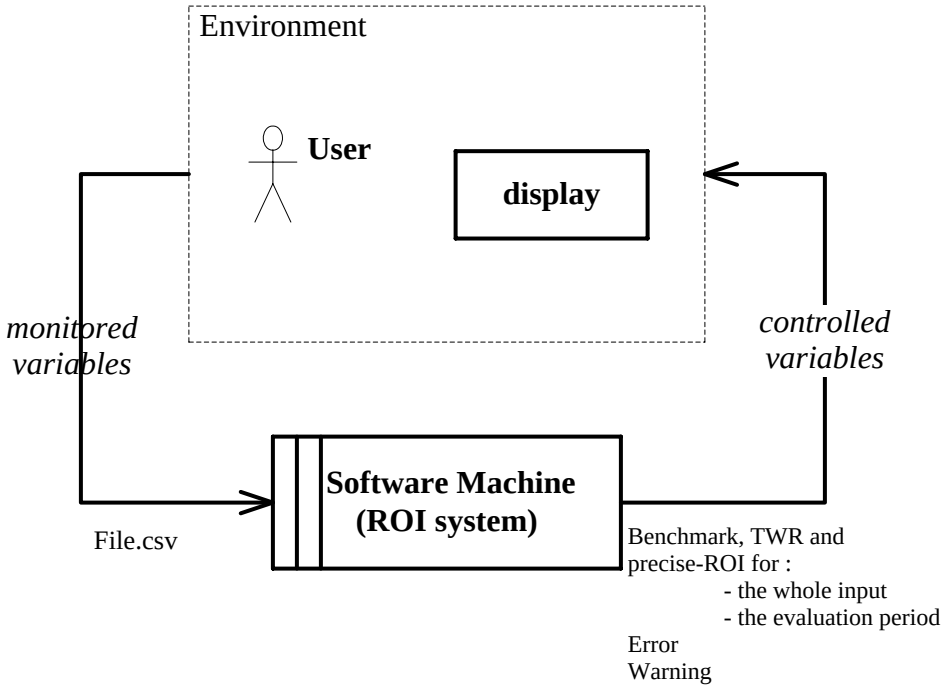

Figure 1: Context diagram for the ROI system

2 Dictionary

Agent fees: Money that the customer pays to the investment advisor to run the account.

Benchmark: Standard used as a point of reference for evaluating performance.

Cash Flow: Revenue or expense stream that changes a cash account over a given period.

CSV: Comma Separated Value file format used to store tabular data in which numbers and text are stored in plain-text form that can be easily written and read in a text editor.

Customer: The user of the software system.

Evaluation Period: a start and end date (provided by the user) for the portfolio history over which the return on investment is calculated.

GIPS: Global Investment Performance Standards

Investment broker: Runs the portfolio on behalf of the customer and supplies portfolio accounts.

Portfolio statement: List of all investments and current value.

Portfolio History: the historical data of investment performance over time that the customer stores about their investments as gleaned from their monthly or yearly investment accounts. Usually stored by customers in a CSV file (see Figure 1).

ROI: Return On Investment: Performance measure used to evaluate the efficiency of an investment.

TWR: Time Weighted Return: Measure of the compound rate of growth in a portfolio.

Tuple data: *date*, *market value*, *cash flow*, *agent fees* and *benchmark*.

3 E/R-descriptions

3.1 E-descriptions

ID	Description	Comment
E1	Customers create and store a portfolio history, i.e. the historical data of their investment performance as determined from portfolio statements.	
E2	Customers store their portfolio history as a CSV text file. CSV files may be prepared on editors of any operating system and encoded as ANSI or UTF-8.	
E3	Every portfolio history has a name.	
E4	Optionally, every portfolio history has a description, account number, email, address, and phone number fields.	
E5	A portfolio history records investment performance in a non-empty sequence of tuple data, each tuple having the fields: date, market value, cash flow, agent fees and benchmark.	See *tr* of TWR_ROI_CALCULATION (Fig. 7)
E6	When there is a customer contribution, the cash flow is a positive number. For a withdrawal, the number is negative.	
E7	Agent fees can be internal (deducted from within the portfolio) or external (additional amounts paid by the customer to the investment broker). The portfolio history reflects only external agent fees, always reported as a non-negative amount.	
E8	Optionally, every portfolio has an evaluation period that is between the start and end date of the historical performance data.	See Invariant 1 of TWR_ROI_CALCULATION (Fig. 7)

3.2 R-descriptions

ID	Description	Comment
R1	All return on investment calculations shall follow the GIPS standard.	See twr, roi, benchmark (Fig. 7)

Evaluation period		
R2.1	The evaluation period is in range.	See Invariant 1 of TWR_ROI_CALCULATION (Fig. 7)
R2.2	If no evaluation period is provided, then the start date is the earliest date and the end date the latest date in the sequence of tuple data.	See Start_Valid and End_Valid in Function Table
R2.3	If the evaluation dates are not valid, then the following error message shall be displayed to the user: "Invalid_Evaluation_Period"	See Function Table

CSV file		
R3.1	Every data tuple (row in the CSV file) has a date and a non-negative market value.	See Invariant 2 of TWR_ROI_CALCULATION (Fig. 7)
R3.2	Dates in the tuples are unique and ordered.	See Invariant 3 of TWR_ROI_CALCULATION (Fig. 7)
R3.3	No withdrawal in the tuple data can be greater than the market value.	See Invariant 4 of TWR_ROI_CALCULATION (Fig. 7)
R3.4	An account cannot grow from zero market value and cash flow.	See Invariant 5 of TWR_ROI_CALCULATION (Fig. 7)
R3.5	For each tuple, the market value plus cash-flow plus agent-fees must be non-zero.	See precondition 3 of feature *twr* of TWR_ROI_CALCULATION (Fig. 7)

R3.6	Error message: If the CSV file is not valid (i.e. if any of the conditions mentioned above do not hold), then the following error message shall be displayed to the user: "Invalid_file".	See Function table
R3.7	All dates must be in ISO format (yyyy-mm-dd).	

Calculation of the TWR		
R4.1	The system should provide two TWR : one for the evaluation period, and one for the whole input.	See Function Table
R4.2	If the evaluation period is less than a year, then the TWR shall be reported in absolute terms as a percentage return (i.e. it is not annualized). If the evaluation period is a year or more, then the TWR is annualized to a percentage per year.	See postcondition of *annual_compounded_TWR* of TWR_ROI_CALCULATION (Fig. 7)
R4.3	The annualized TWR shall be reported as a percentage.	See *annual_compounded_TWR* of TWR_ROI_CALCULATION (Fig. 7)
R4.4	Agent fees are treated like a deposit.	See *annual_compounded_TWR* of TWR_ROI_CALCULATION (Fig. 7)
R4.5	Warning message: If the TWR is not calculable, then a warning message shall be displayed to the user.	See Function Table

Calculation of the ROI		
R5.1	The system should provide two ROI : one for the evaluation period, and one for the whole input.	See Function Table
R5.2	The ROI shall be reported as a percentage.	See *roi* of TWR_ROI_CALCULATION (Fig. 7)
R5.3	Agent fees are treated like a deposit.	See *roi* of TWR_ROI_CALCULATION (Fig. 7)

Calculation of the Benchmark		
R6.1	The system should provide two benchmarks : one for the evaluation period, and one for the whole input.	See Function Table
R6.2	The benchmark shall be reported as a compounded ROI, if the benchmark figures are available for the evaluation period.	See *benchmark* of TWR_ROI_CALCULATION (Fig. 7)
R6.3	Warning message: If the benchmark is not calculable, then a warning message shall be displayed to the user.	See Function Table

4 Mathematical model

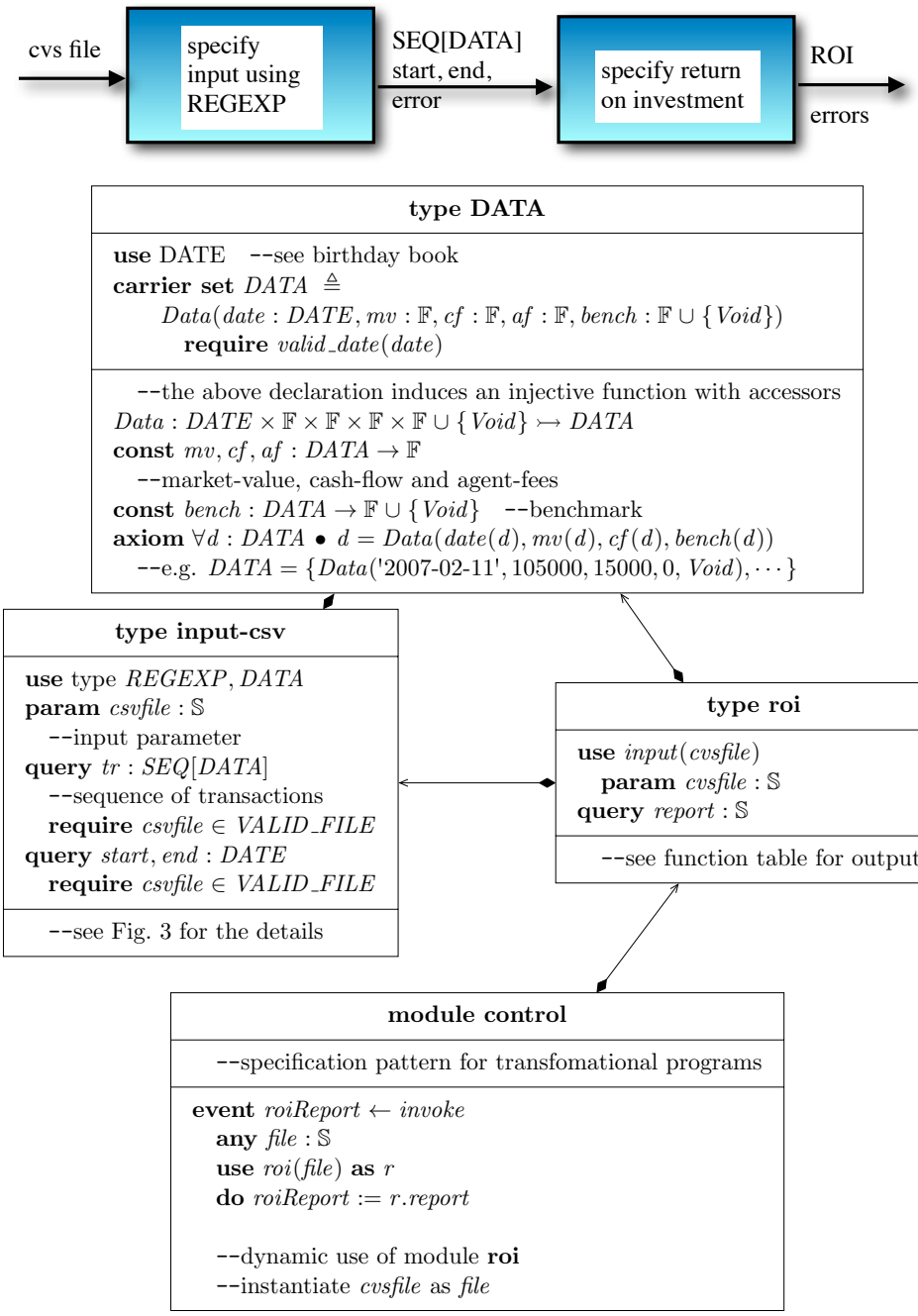

Figure 2: Module specification of return on investment

type input-csv

use type $REGEXP, DATA, DATE$ --we let $\epsilon = \{\text{“”}\}$, eol = {“\n”} etc.
carrier set $DATA \triangleq Data(date : DATE, mv : \mathbb{F}, cf : \mathbb{F}, af : \mathbb{F}, bench : \mathbb{F} \cup \{Void\})$
param $csvfile : \mathbb{S}$ --input parameter
query $tr : SEQ[DATA]$ --sequence of transactions defined by axiom below
  require $csvfile \in VALID\_FILE$
query $start, end : DATE$
  require $csvfile \in VALID\_FILE$

const $VALID\_FILE : REGEXP$
 $\triangleq HEADER \cdot PARAMETERS \cdot \text{eol} \cdot ROW \cdot *(\text{eol} \cdot ROW) \cdot *(\text{“,”}|\text{eol})$
const $HEADER : REGEXP$
 $\triangleq *(HLINE \cdot eol)$
const $HLINE : REGEXP$
 $\triangleq *(\Sigma \backslash \text{eol}) \backslash (EV\_PER \cdot *\Sigma)$
const $PARAMETERS : REGEXP$
 $\triangleq EV\_PER \cdot DATE\_STR \cdot \text{“\_to\_”} \cdot DATE\_STR \cdot *\text{“,”} \cdot \text{eol} \cdot COL\_HEAD$
const $COL\_HEAD : REGEXP$
 $\triangleq + \text{“,”} \cdot eol \cdot$
  “Transaction_Date,Market_Value,Cash_Flow,Agent_Fees,Benchmark” $\cdot *\text{“,”}$
const $EV\_PER : REGEXP \triangleq$ “Evaluation_Period:_”
const $ROW : REGEXP$
 $\triangleq (DATE\_STR \cdot \text{“,”} \cdot FLOAT \cdot \text{“,”} \cdot (FLOAT|\epsilon) \cdot \text{“,”} \cdot (FLOAT|\epsilon) \cdot$
  $\text{“,”} \cdot (FLOAT \cdot \text{“\%”}|\epsilon) \cdot *\text{“,”})$
const $s2d : DATE\_STR \to DATE$ --see birthday book for $DATE$
const $s2f : FLOAT \to \mathbb{F}$ --deferred, $FLOAT$ is the string version of $\mathbb{F}$
const $f2s : \mathbb{F} \to FLOAT$ --deferred, see your favourite programming language
const $d2s : DATE \to DATE\_STR$ --deferred
const $s2optf[G] : (FLOAT|\epsilon) \times G \to \mathbb{F} \cup G$ --string-to-optional float
  where $\forall G \bullet s2optf \in (FLOAT|\epsilon) \times G \to \mathbb{F} \cup G$
  --parameter G is a set such as $\{Void\}$ or a default value such as $\{0\}$
const $f : ROW \to DATA$
dummy $w : ROW$ and $s_0, s_1, s_2, s_3 : \mathbb{S}$
axiom 1: --definition of function f that maps a row string to data
  $w \in (d2s(d) \cdot \text{“,”} \cdot s_0 \cdot \text{“,”} \cdot s_1 \cdot \text{“,”} \cdot s_2 \cdot \text{“,”} \cdot s_3 \cdot *\text{“,”})$
  $\wedge\ (s_4 \cdot \text{“\%”} = s_3 \vee s_4 = s_3 = \epsilon)$
  $\Rightarrow f(w) = Data(d, s2f(s_0), s2optf(s_1, 0), s2optf(s_2, 0), s2optf(s_4, \text{Void}))$
query $error : \mathbb{B} \triangleq textfile \notin VALID\_FILE$ --definition of $tr, start, end$
axiom 2: --definition of $tr, start, end$
  $csvfile \in VALID\_FILE \Rightarrow$
  $(\exists h, foot, s, e : \mathbb{S}; data : SEQ[ROW]$
   $\mid\ h \in HEADER \cdot EV\_PER \cdot s \cdot \text{“\_to\_”} \cdot e \cdot *\text{“,”} \cdot \text{eol} \cdot COL\_HEAD$
   $\wedge\, data \in SEQ[ROW]$
   $\wedge\, end \in *(\text{','}|eol)$
   $\wedge\, textfile \in h \cdot (\cdot i | 0 \le i < \#data \bullet eol \cdot data(i)) \cdot foot$
   $\bullet\ \ tr = (\cdot i | 0 \le i < \#data \bullet < f(data(i)) >$
    $\wedge (start = s2d(s)) \wedge (end = s2d(e))$
)

Figure 3: Type input-csv

TWR_ROI_CALCULATION

--**input** (*input.csv*)

tr: SEQ[TUPLE[*date*: DATE, *mv*: VALUE, *cf*: VALUE, *af*: VALUE, *bm*: VALUE ∪ {void}]]
--sequence of transaction-tuples [*date, market_value, cash_flow, agent_fees,*
--*benchmark*]
--*tr.domain* = {1,2,...,*tr.count*}

count: INTEGER ≙ *tr.count*

dates: SET[DATE] ≙ {*t* ∈ *tr* • *t.date*}

start, end: DATE -- metadata evaluation period

duration: VALUE ≙ *days(end - start)*÷(365.2422)
--years between *start* and *end* calculated by days
--*days*(x) similar to Excel

--**output calculation** (*input.out.csv*)

di (d:DATE): INTEGER
--index into sequence of transaction for date *d*
require *d* ∈ *dates*
ensure *Result* ∈ *tr.domain* ∧ *tr[Result].date=d*

--TWR for the period *start .. end*
twr (a_start, a_end: DATE): VALUE
require
a_start, a_end ∈ *dates*
a_end > *a_start*
∀*i* ∈ *2..count* • *tr[i-1].mv + tr[i-1].cf + tr[i-1].af* ≠ *0*
ensure
Result ≙ (Π*i:INTEGER* | *di(a_start)* <*i*≤ *di(a_end)* • *wealth(i)) - 1*
where *wealth(i)* ≙ *tr(i].mv*÷*(tr[i-1].mv + tr[i-1].cf + tr[i-1].af)*

annual_compounded_TWR (a_start, a_end: DATE): VALUE
ensure
(duration$\geq 1) \Rightarrow$ *Result* $= ((1 + twr(a\_start, a\_end))^{1 \div duration} - 1) * 100$
(duration<1)$\Rightarrow$ *Result= twr(a_start, a_end) * 100*

roi (a_start, a_end: DATE): VALUE
require
a_start, a_end $\in$ *dates*
a_end $>$ *a_start*
ensure
$(tr[m].mv + tr[m].cf) * (1 + Result \div 100)^{days(a\_end - a\_start) \div 365.2422}$
$+ (\Sigma i : \mathbb{N} | m < i < n \bullet (tr[i].cf + tr[i].af) *$
$(1 + Result \div 100)^{days(a\_end - tr[i].date) \div 365.2422}) - tr[n].mv = 0$
where $m = di(a\_start)$
$n = di(a\_end)$

benchmark_calculable (a_start, a_end: DATE): BOOL
require
$\{\forall t \in tr | t.date = "yyyy - 01 - 01" \bullet t.bench \neq void\}$
$\cup \{tr[di(a\_end)].bench \neq void\}$
ensure
Result = TRUE

--the function below return a set of index corresponding to the date with
--a benchmark
bench_seq (a_start, a_end: DATE): SEQ[INTEGER]
require
benchmark_calculable(a_start, a_end)
ensure
$\{\forall i \in Result \bullet tr[i].bench \neq void \wedge (di(a\_start) < i \leq di(a\_end))\}$
-- *Result.domain* = {*1,2..Result.count*}

benchmark (a_start, a_end: DATE): VALUE
require
benchmark_calculable(a_start, a_end)
ensure

$$(tr[m].mv + tr[m].cf) * (Result + 1)^{days(a\_end - a\_start) \div 365.2422}$$
$$+(\Sigma k : VALUE | m + 1 \leq k \leq n \bullet tr[k].cf *$$
$$(Result + 1)^{days(a\_end - tr[k].date) \div 365.2422} = FV$$

where $m = di(a\_start)$
$n = di(a\_end)$
$FV \triangleq$
$(tr[m].mv + tr[m].cf) * (\Pi i : VALUE | s = bench\_seq(a\_start, a\_end)$
$\wedge s[0] = m \wedge i \in s \wedge i \geq 1 \wedge i = s(j) \bullet$
$tr[i].bench^{days(tr[i].date - tr[s(j-1)].date) \div 365.2422})$
$+(\Sigma k : VALUE | m + 1 \leq k \leq n \bullet (tr[k].cf - tr[k].af) *$
$(\Pi i : VALUE | s = bench\_seq(tr[k].date, a\_end) \wedge s[0] = k \wedge i \in s$
$\wedge i \geq 1 \wedge i = s(j) \bullet tr[k].bench^{days(tr[i].date - tr[s(j-1)].date) \div 365.2422}))$

Invariants

(1) *(start<end)∧(start,end∈dates)*
--metadata evaluation period is in range and valid

(2) ∀*t*∈*tr* • *t.date*≠*Void*∧*t.mv*≥*0*
--every row has a date and a non-negative market value

(3) ∀*i*∈*2..count* • *tr[i].date>tr[i-1].date*
--date are unique and ordered

(4) ∀*t*∈*tr* • *t.mv+t.cf*≥*0*
--Cannot withdraw more than the market value

(5) ∀*i*∈*2..count* | *tr[i-1].mv=0*∧*tr[i-1].cf=0* • *tr[i].mv=0*
--account coannot grow from zero market value and cash flow

Table 7: Mathematical model for the ROI system

						Error	*Warning*	Whole input		
								TWR	ROI	B
Valid_CSV	Start_Invalid ∨ End_Invalid ∨ end ≤ start					E1	—	—	—	—
	Start_Valid ∧ End_Valid ∧ end>start	C1	C2			—	—	a_c_TWR(all)	roi(all)	b(all)
			¬C2	C4		—	W1	a_c_TWR(all)	roi(all)	—
				¬C4		—	W2	a_c_TWR(all)	roi(all)	—
		¬C1	C3	C2		—	W3	—	roi(all)	b(all)
				¬C2	C4	—	W4	—	roi(all)	—
					¬C4	—	W5	—	roi(all)	—
			¬C3	C2		—	W6	—	roi(all)	b(all)
				¬C2	C4	—	W7	—	roi(all)	—
					¬C4	—	W8	—	roi(all)	—
Invalid_CSV						E2	—	—	—	—

Table 8: Function table for ROI system (error, warning and whole input)

					Evaluation period		
					TWR	ROI	B
Valid_CSV	Start_Invalid ∨ End_Invalid ∨ end ≤ start				—	—	—
	Start_Valid ∧ End_Valid ∧ end>start	C1	C2		a_c_TWR(start,end)	roi(start,end)	b(start,end)
			¬C2	C4	a_c_TWR(start,end)	roi(start,end)	b(start,end)
				¬C4	a_c_TWR(start,end)	roi(start,end)	—
		¬C1	C3	C2	a_c_TWR(start,end)	roi(start,end)	b(start,end)
				¬C2 C4	a_c_TWR(start,end)	roi(start,end)	b(start,end)
				¬C2 ¬C4	a_c_TWR(start,end)	roi(start,end)	—
			¬C3	C2	—	roi(start,end)	b(start,end)
				¬C2 C4	—	roi(start,end)	b(start,end)
				¬C2 ¬C4	—	roi(start,end)	—
Invalid_CSV					—	—	—

Table 9: Function table for ROI system (evaluation period)

The function tables use some abbreviations:

- a_c_TWR = annual_compounded_TWR (see TWR_ROI_CALCULATION (Fig. 7))
- b = benchmark (see TWR_ROI_CALCULATION (Fig. 7))
- function(all) = function(tr[1].date, tr[tr.count].date)

The function tables use also conditions:

- C1 = $\forall i \in 2..count \bullet tr[i\text{-}1].mv + tr[i\text{-}1].cf + tr[i\text{-}1].af \neq 0$
- C2 = benchmark_calculable(all) (see TWR_ROI_CALCULATION (Fig. 7))
- C3 = $\forall i \in di(start)+1..di(end) \bullet tr[i\text{-}1].mv + tr[i\text{-}1].cf + tr[i\text{-}1].af \neq 0$
- C4 = benchmark_calculable(start,end) (see TWR_ROI_CALCULATION (Fig. 7))
- Start_Valid = ¬ (Start_Invalid) = (start ∈ dates∪{null}) ∧ (start in ISO format)
  (if Start_Valid ∧ start=null, then start=tr[1].date)
- End_Valid = ¬ (End_Invalid) = (end ∈ dates∪{null}) ∧ (end in ISO format)
  (if End_Valid ∧ end=null, then end=tr[tr.count].date)

The function tables provide errors and warnings messages:

- E1 = "Invalid_Evaluation_Period"
- E2 = "Invalid_file"
- W1 = "Benchmark for the whole input is not calculable"
- W2 = "The benchmarks are not calculable"
- W3 = "The TWR for the whole input is not calculable"

- W4 = W1+W3
- W5 = W1+W2+W3
- W6 = "The TWR's are not calculable"
- W7 = W1+W6
- W8 = W2+W6

A REGEXP

A set of strings is used as the model for regular expressions. We use prefix operators for the Kleene closure (e.g. $*x$ where x is a regular expression such as {'hello'}) and iteration at least one or more (e.g. $+x$) rather than suffix

type REGEXP
carrier set $REGEXP$ --set of all regular string expressions **axiom** $REGEXP \subseteq \mathbb{P}(\mathbb{S})$ **carrier set** $\Sigma \triangleq$ {"0", "1", "2", $\cdots$, "a", "b", etc., all printing characters}
dummy $x, y, z : REGEXP$ **dummy** $s, t, u : \mathbb{S}$ **axiom** $\forall s \in \Sigma \bullet \{s\} \in REGEXP$ **const** $0 : REGEXP \triangleq \{\}$ --zero is the unit element of alternation **const** $1 : REGEXP \triangleq \{\text{""}\}$ --1 is the unit element of concatenation --we also use ϵ instead of 1 **const infix** "$\mid$" : $REGEXP \times REGEXP \to REGEXP$ --alternation **const infix** "$\cdot$" : $REGEXP \times REGEXP \to REGEXP$ --concatenation **const prefix** "$*$" : $REGEXP \times REGEXP \to REGEXP$ --iteration zero or more times **const prefix** "+" : $REGEXP \times REGEXP \to REGEXP$ --iteration one or more times **axiom** $s \in x \mid y \equiv s \in x \vee s \in y$ **theorem** $x \mid 0 = 0 \mid x = x$ **axiom** $s \in x \cdot y \equiv (\exists t, u \mid s = t \cdot u \bullet t \in x \wedge u \in y)$ --note that $t \cdot u$ is concatenation over $SEQ[\mathbb{S}]$ **theorem** $1 \cdot x = x \cdot 1 = 1$ --1 is the identity of concatenation **const infix** "^": $REGEXP \times \mathbb{N} \to REGEXP$ --use this operator by raising the second argument like an exponent **axiom** $x^n = (\cdot i \mid 0 \leq i \leq n \bullet x)$ --concatenation quantifier --e.g. $x^3 = x \cdot x \cdot x$ **theorem** $x^0 = 1$ **axiom** $s \in *x \equiv (\exists n : \mathbb{N} \bullet s \in x^n)$ **axiom** $s \in +x \equiv (\exists n : \mathbb{N}_1 \bullet s \in x^n)$

Figure 4: Type REGEXP for regular expressions over printing characters

operators. Note that where there is no confusion we use 'hello' instead of {'hello'} where the set is a singleton.

We may use type REGEXP to specify a *FLOAT_STRING* as follows.

$$
\begin{aligned}
FLOAT\_STRING =& \text{'+'}Inf && (1)\\
&|\text{'-'}Inf && (2)\\
&|NaN && (3)\\
&|(\text{'-'}|\text{'+'}|\epsilon)\cdot({}^{*}d\cdot\text{'.'}|\epsilon)\cdot{}^{*}d\cdot((\text{'e'}\cdot(\text{'-'}|\epsilon)\cdot{}^{+}d)\,|\,\epsilon) && (4)\\
d =& \text{'0'}|\text{'1'}|\cdots|\text{'9'} && (5)
\end{aligned}
$$

In the above we use the convention that 'e', for example, really stands for the single set {'e'}.