

CSE4312F12 Project Solution ROI

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

November 27, 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.
- You 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

1	Context Diagram	3
2	Dictionary	4
3	E/R-descriptions	5
3.1	E-descriptions	5
3.2	R-descriptions	7
4	Mathematical model	9
A	REGEXP	18

List of Figures

1	Context diagram for the ROI system	4
2	Module specification of return on investment	10
3	Type input-csv	11
4	Type REGEXP for regular expressions over printing characters	18

List of Tables

10	Mathematical model for the ROI system	14
11	Function table for ROI system (error, warning and whole input)	15
12	Function table for ROI system (evaluation period)	16

1 Context Diagram

The following diagram is the context diagram for the ROI system.

The monitored variables (which are the content of the CSV file, provided by the user), are :

- an optional header, which is composed of a description of the file and information about the customer (account number, email, address and phone number)
- the evaluation dates (*start* and *end*)
- the tuple data (*date*, *market value*, *cash flow*, *agent fees* and *benchmark*).

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).

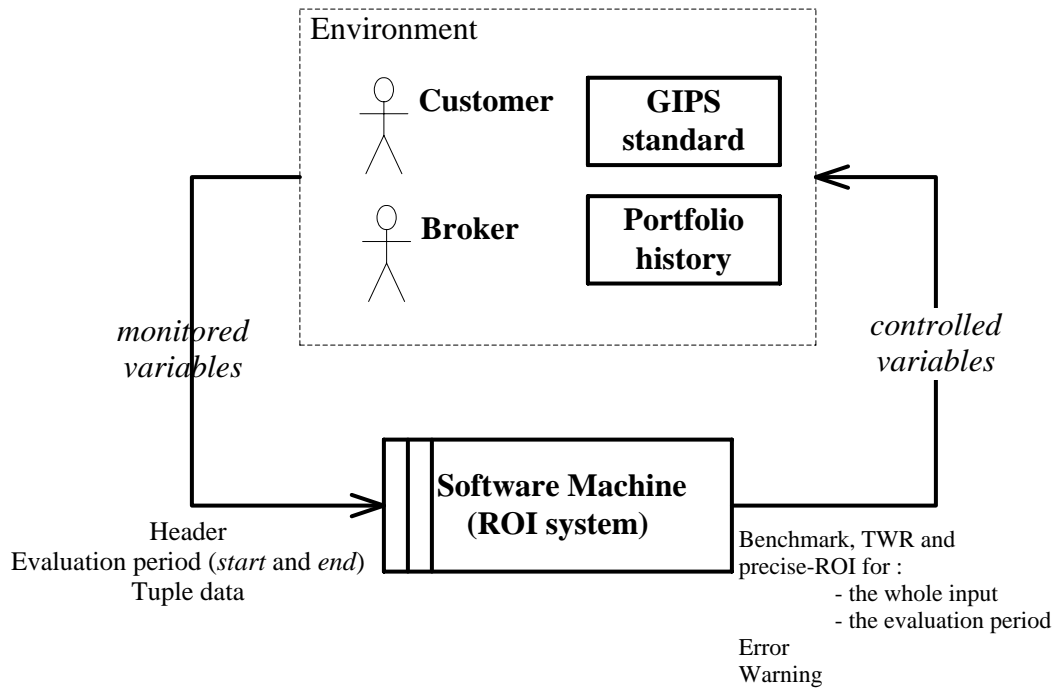


Figure 1: Context diagram for the ROI system

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	Optionally, every portfolio history has a description, account number, email, address, and phone number fields.	
E4	A portfolio history records investment performance in a non-empty sequence of tuple data, each tuple having the fields: date (required), market value (required), cash flow (optional), agent fees (optional) and benchmark (optional).	See <i>tr</i> of TWR_ROI_CALCULATION (Fig. 10)
E5	When there is a customer contribution, the cash flow is a positive number. For a withdrawal, the number is negative.	

E6	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.	
E7	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. 10)

Evaluation period		
E8	The evaluation period is in range.	See Invariant 1 of TWR_ROI_CALCULATION (Fig. 10)

CSV file		
E.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. 10)
E.2	Dates in the tuples are unique and ordered.	See Invariant 3 of TWR_ROI_CALCULATION (Fig. 10)
E.3	No withdrawal in the tuple data can be greater than the market value.	See Invariant 4 of TWR_ROI_CALCULATION (Fig. 10)
E.4	An account cannot grow from zero market value and cash flow.	See Invariant 5 of TWR_ROI_CALCULATION (Fig. 10)
E.5	For each tuple, the market value plus cash-flow plus agent-fees must be non-zero.	See precondition 3 of feature <i>twr</i> of TWR_ROI_CALCULATION (Fig. 10)
E.6	All dates must be in ISO format (yyyy-mm-dd).	

3.2 R-descriptions

ID	Description	Comment
R1	All return on investment calculations shall follow the GIPS standard.	See twr, roi, benchmark (Fig. 10)

Evaluation period		
R2.1	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
R2.2	Error message: 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	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

Calculation of the TWR		
R4.1	The system must provide two TWRs (if each one is calculable) : 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 <i>annual_compounded_TWR</i> of TWR_ROI_CALCULATION (Fig. 10)
R4.3	The annualized TWR shall be reported as a percentage.	See <i>annual_compounded_TWR</i> of TWR_ROI_CALCULATION (Fig. 10)

R4.4	Agent fees are treated like a deposit (the agent fees are <u>added</u> to the market value and the cash flow).	See <i>twr</i> of TWR_ROI_CALCULATION (Fig. 10)
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 must provide two ROIs : 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 <i>roi</i> of TWR_ROI_CALCULATION (Fig. 10)
R5.3	Agent fees are treated like a deposit (the agent fees are <u>added</u> to the cash flow).	See <i>roi</i> of TWR_ROI_CALCULATION (Fig. 10)

Calculation of the Benchmark		
R6.1	The system must provide two benchmarks (if each one is calculable) : 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 <i>benchmark</i> of TWR_ROI_CALCULATION (Fig. 10)
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

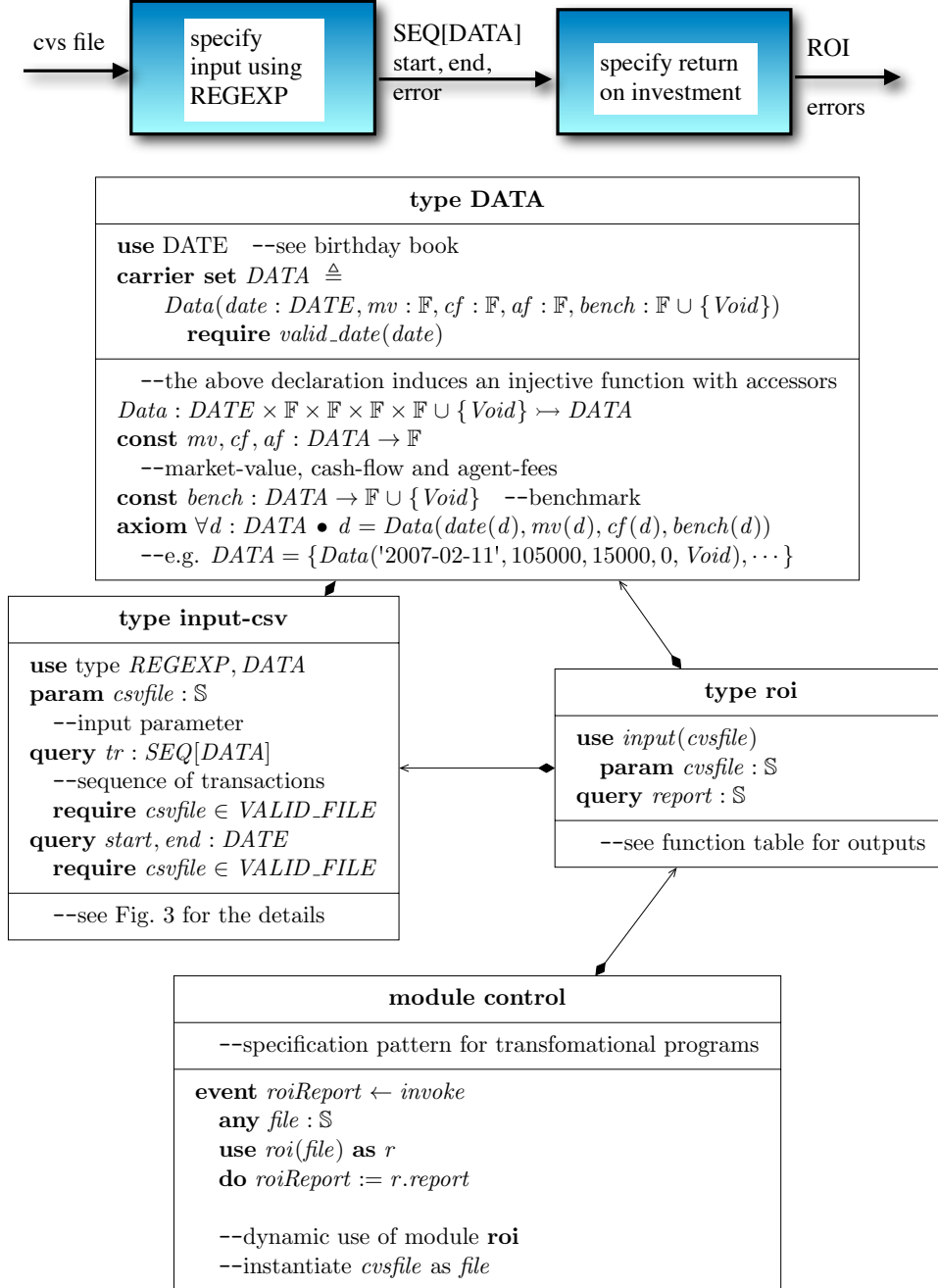


Figure 2: Module specification of return on investment

type input-csv
<pre> use type <i>REGEXP</i>, <i>DATA</i>, <i>DATE</i> --we let $\epsilon = \{“”\}$, $\text{eol} = \{\backslash\text{n}\}$ etc. carrier set <i>DATA</i> $\triangleq \text{Data}(\text{date} : \text{DATE}, \text{mv} : \mathbb{F}, \text{cf} : \mathbb{F}, \text{af} : \mathbb{F}, \text{bench} : \mathbb{F} \cup \{\text{Void}\})$ param <i>csvfile</i> : \mathbb{S} --input parameter query <i>tr</i> : <i>SEQ</i>[<i>DATA</i>] --sequence of transactions defined by axiom below require <i>csvfile</i> $\in \text{VALID_FILE}$ query <i>start</i>, <i>end</i> : <i>DATE</i> require <i>csvfile</i> $\in \text{VALID_FILE}$ const <i>VALID_FILE</i> : <i>REGEXP</i> $\triangleq \text{HEADER} \cdot \text{PARAMETERS} \cdot \text{eol} \cdot \text{ROW} \cdot *(\text{eol} \cdot \text{ROW}) \cdot *(", \text{eol})$ const <i>HEADER</i> : <i>REGEXP</i> $\triangleq *(\text{HLINE} \cdot \text{eol})$ const <i>HLINE</i> : <i>REGEXP</i> $\triangleq *(\Sigma \backslash \text{eol}) \backslash (\text{EV_PER} \cdot * \Sigma)$ const <i>PARAMETERS</i> : <i>REGEXP</i> $\triangleq \text{EV_PER} \cdot \text{DATE_STR} \cdot \text{"_to_"} \cdot \text{DATE_STR} \cdot * ", " \cdot \text{eol} \cdot \text{COL_HEAD}$ const <i>COL_HEAD</i> : <i>REGEXP</i> $\triangleq + ", " \cdot \text{eol} \cdot$ "Transaction_Date,Market_Value,Cash_Flow,Agent_Fees,Benchmark" $\cdot * ", "$ const <i>EV_PER</i> : <i>REGEXP</i> $\triangleq \text{"Evaluation_Period:_"}$ const <i>ROW</i> : <i>REGEXP</i> $\triangleq (\text{DATE_STR} \cdot ", " \cdot \text{FLOAT} \cdot ", " \cdot (\text{FLOAT} \epsilon) \cdot ", " \cdot (\text{FLOAT} \epsilon) \cdot$ ", " $\cdot (\text{FLOAT} \cdot \text{"\%" } \epsilon) \cdot * ", "$) const <i>s2d</i> : <i>DATE_STR</i> $\rightarrow \text{DATE}$ --see birthday book for <i>DATE</i> const <i>s2f</i> : <i>FLOAT</i> $\rightarrow \mathbb{F}$ --deferred, <i>FLOAT</i> is the string version of \mathbb{F} const <i>f2s</i> : $\mathbb{F} \rightarrow \text{FLOAT}$ --deferred, see your favourite programming language const <i>d2s</i> : <i>DATE</i> $\rightarrow \text{DATE_STR}$ --deferred const <i>s2optf</i>[<i>G</i>] : $(\text{FLOAT} \epsilon) \times G \rightarrow \mathbb{F} \cup G$ --string-to-optional float where $\forall G \bullet s2optf \in (\text{FLOAT} \epsilon) \times G \rightarrow \mathbb{F} \cup G$ --parameter <i>G</i> is a set such as $\{\text{Void}\}$ or a default value such as $\{0\}$ const <i>f</i> : <i>ROW</i> $\rightarrow \text{DATA}$ dummy <i>w</i> : <i>ROW</i> and <i>s</i>₀, <i>s</i>₁, <i>s</i>₂, <i>s</i>₃ : \mathbb{S} axiom 1: --definition of function <i>f</i> that maps a row string to data $w \in (d2s(d) \cdot ", " \cdot s_0 \cdot ", " \cdot s_1 \cdot ", " \cdot s_2 \cdot ", " \cdot s_3 \cdot * ", ")$ $\wedge (s_4 \cdot \text{"\%" } = s_3 \vee s_4 = s_3 = \epsilon)$ $\Rightarrow f(w) = \text{Data}(d, s2f(s_0), s2optf(s_1, 0), s2optf(s_2, 0), s2optf(s_4, \text{Void}))$ query <i>error</i> : $\mathbb{B} \triangleq \text{textfile} \notin \text{VALID_FILE}$ --definition of <i>tr</i>, <i>start</i>, <i>end</i> axiom 2: --definition of <i>tr</i>, <i>start</i>, <i>end</i> <i>csvfile</i> $\in \text{VALID_FILE} \Rightarrow$ $(\exists h, \text{foot}, s, e : \mathbb{S}; \text{data} : \text{SEQ}[\text{ROW}]$ $h \in \text{HEADER} \cdot \text{EV_PER} \cdot s \cdot \text{"_to_"} \cdot e \cdot * ", " \cdot \text{eol} \cdot \text{COL_HEAD}$ $\wedge \text{data} \in \text{SEQ}[\text{ROW}]$ $\wedge \text{end} \in *(' \text{eol})$ $\wedge \text{textfile} \in h \cdot (\cdot 0 \leq i < \# \text{data} \bullet \text{eol} \cdot \text{data}(i)) \cdot \text{foot}$ $\bullet \text{tr} = (\cdot 0 \leq i < \# \text{data} \bullet < f(\text{data}(i)) >$ $\wedge (\text{start} = s2d(s)) \wedge (\text{end} = s2d(e))$) </pre>

Figure 3: Type input-csv

TWR_ROI_CALCULATION

```

--input (input.csv)

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

count: INTEGER  $\triangleq$  tr.count

dates: SET[DATE]  $\triangleq$  {t  $\in$  tr  $\bullet$  t.date}

start, end: DATE  -- metadata evaluation period

duration: VALUE  $\triangleq$  days(end - start)  $\div$  (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  $\in$  dates
ensure Result  $\in$  tr.domain  $\wedge$  tr[Result].date=d

  --TWR for the period start .. end
twr (a_start, a_end: DATE): VALUE
require
  a_start, a_end  $\in$  dates
  a_end > a_start
   $\forall i \in 2..count \bullet tr[i-1].mv + tr[i-1].cf + tr[i-1].af \neq 0$ 
ensure
  Result  $\triangleq$  ( $\Pi i:INTEGER \mid di(a\_start) < i \leq di(a\_end) \bullet wealth(i)$ ) - 1
  where wealth(i)  $\triangleq$  tr[i].mv  $\div$  (tr[i-1].mv + tr[i-1].cf + tr[i-1].af)


```

```

annual_compounded_TWR (a_start, a_end: DATE): VALUE
  ensure
    (duration ≥ 1) ⇒ Result = ((1 + twr(a_start, a_end))1 ÷ duration - 1) * 100
    (duration < 1) ⇒ Result = twr(a_start, a_end) * 100

roi (a_start, a_end: DATE): VALUE
  require
    a_start, a_end ∈ dates
    a_end > a_start
  ensure
    (tr[m].mv + tr[m].cf) * (1 + Result ÷ 100)days(a_end - a_start) ÷ 365.2422
      + (Σ i : ℕ | m < i < n • (tr[i].cf + tr[i].af)) *
      (1 + Result ÷ 100)days(a_end - tr[i].date) ÷ 365.2422 - tr[n].mv = 0
    where m = di(a_start)
           n = di(a_end)

benchmark_calculable (a_start, a_end: DATE): BOOL
  require
    {∀ t ∈ tr | t.date = "yyyy - 01 - 01" • t.bench ≠ void}
    ∪ {tr[di(a_end)].bench ≠ 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
    {∀ i ∈ Result • tr[i].bench ≠ void ∧ (di(a_start) < i ≤ di(a_end))}
    -- Result.domain = {1, 2..Result.count}

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

```

$$\begin{aligned}
& (tr[m].mv + tr[m].cf) * (Result + 1)^{days(a_end - a_start) \div 365.2422} \\
& + (\sum k : VALUE | m + 1 \leq k \leq n \bullet tr[k].cf * \\
& \quad (Result + 1)^{days(a_end - tr[k].date) \div 365.2422} = FV
\end{aligned}$$

where $m = di(a_start)$

$n = di(a_end)$

$FV \triangleq$

$$\begin{aligned}
& (tr[m].mv + tr[m].cf) * (\Pi i : VALUE | s = bench_seq(a_start, a_end) \\
& \quad \wedge s[0] = m \wedge i \in s \wedge i \geq 1 \wedge i = s(j) \bullet \\
& \quad tr[i].bench^{days(tr[i].date - tr[s(j-1)].date) \div 365.2422}) \\
& + (\sum k : VALUE | m + 1 \leq k \leq n \bullet (tr[k].cf - tr[k].af) * \\
& \quad (\Pi i : VALUE | s = bench_seq(tr[k].date, a_end) \wedge s[0] = k \wedge i \in s \\
& \quad \wedge i \geq 1 \wedge i = s(j) \bullet tr[k].bench^{days(tr[i].date - tr[s(j-1)].date) \div 365.2422}))
\end{aligned}$$

Invariants

- (1) $(start < end) \wedge (start, end \in dates)$
--metadata evaluation period is in range and valid
- (2) $\forall t \in tr \bullet t.date \neq Void \wedge t.mv \geq 0$
--every row has a date and a non-negative market value
- (3) $\forall i \in 2..count \bullet tr[i].date > tr[i-1].date$
--date are unique and ordered
- (4) $\forall t \in tr \bullet t.mv + t.cf \geq 0$
--Cannot withdraw more than the market value
- (5) $\forall i \in 2..count \mid tr[i-1].mv = 0 \wedge tr[i-1].cf = 0 \bullet tr[i].mv = 0$
--account coannot grow from zero market value and cash flow

Table 10: Mathematical model for the ROI system

		Error	Warning	Whole input			
				TWR	ROI B		
Valid_CSV	Start_Invalid \vee End_Invalid \vee end \leq start	E1	—	—	—		
	Start_Valid \wedge End_Valid \wedge end > start	C1	C2	—	roi(all)	b(all)	
		\neg C2	C4	W1	a_c.TWR(all)	roi(all)	—
			\neg C4		W2	a_c.TWR(all)	roi(all)
		C3	C2	W3	—	roi(all)	b(all)
			\neg C2	C4	W4	—	roi(all)
		\neg C3	\neg C4	W5	—	roi(all)	—
			C2	W6	—	roi(all)	b(all)
		\neg C2	C4	W7	—	roi(all)	—
\neg C4		W8	—	roi(all)	—		
Invalid_CSV		E2	—	—	—		

		Evaluation period		
		TWR	ROI	B
Valid_CSV	Start_Invalid \vee End_Invalid \vee end \leq start	—	—	—
	Start_Valid \wedge End_Valid \wedge end > start	a_c_TWR(start,end)	roi(start,end)	b(start,end)
	C1	C2	roi(start,end)	b(start,end)
	\neg C1	\neg C2	roi(start,end)	b(start,end)
		C4	roi(start,end)	—
	\neg C2	\neg C4	roi(start,end)	—
		C2	roi(start,end)	b(start,end)
	\neg C3	C4	roi(start,end)	b(start,end)
		\neg C4	roi(start,end)	—
Invalid_CSV		—	roi(start,end)	b(start,end)
		—	roi(start,end)	b(start,end)
		—	roi(start,end)	—
		—	roi(start,end)	—

Table 12: Function table for ROI system (evaluation period)

The function tables use some abbreviations:

- a_c_TWR = $annual_compounded_TWR$ (see $TWR_ROI_CALCULATION$ (Fig. 10))
- b = $benchmark$ (see $TWR_ROI_CALCULATION$ (Fig. 10))
- $function(all) = function(tr[1].date, tr[tr.count].date)$

The function tables use also conditions:

- $C1 = \forall i \in 2..count \bullet tr[i-1].mv + tr[i-1].cf + tr[i-1].af \neq 0$
- $C2 = benchmark_calculable(all)$ (see $TWR_ROI_CALCULATION$ (Fig. 10))
- $C3 = \forall i \in di(start)+1..di(end) \bullet tr[i-1].mv + tr[i-1].cf + tr[i-1].af \neq 0$
- $C4 = benchmark_calculable(start,end)$ (see $TWR_ROI_CALCULATION$ (Fig. 10))
- $Start_Valid = \neg (Start_Invalid) = (start \in dates \cup \{null\}) \wedge (start \text{ in ISO format})$
(if $Start_Valid \wedge start=null$, then $start=tr[1].date$)
- $End_Valid = \neg (End_Invalid) = (end \in dates \cup \{null\}) \wedge (end \text{ in ISO format})$
(if $End_Valid \wedge end=null$, then $end=tr[tr.count].date$)

The function tables provide errors and warnings messages:

- $E1 = "Invalid_Evaluation_Period"$
- $E2 = "Invalid_file"$
- $W1 = "Benchmark \text{ for the whole input is not calculable}"$
- $W2 = "The benchmarks are not calculable"$
- $W3 = "The TWR \text{ for the whole input is not calculable}"$
- $W4 = W1+W3$
- $W5 = W1+W2+W3$
- $W6 = "The TWR's are not calculable"$
- $W7 = W1+W6$
- $W8 = W2+W6$

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 $\{\text{'hello'}\}$) and iteration at least one or more (e.g. $+x$) rather than

type REGEXP
carrier set $REGEXP$ --set of all regular string expressions axiom $REGEXP \subseteq \mathbb{P}(\mathbb{S})$ carrier set $\Sigma \triangleq \{“0”, “1”, “2”, \dots, “a”, “b”, \text{ etc.}, \text{ all printing characters}\}$
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 $1 : REGEXP \triangleq \{“”\}$ --1 is the unit element of concatenation --we also use ϵ instead of 1 const infix $“ ” : REGEXP \times REGEXP \rightarrow REGEXP$ --alternation const infix $“.” : REGEXP \times REGEXP \rightarrow REGEXP$ --concatenation const prefix $“*” : REGEXP \times REGEXP \rightarrow REGEXP$ --iteration zero or more times const prefix $“+” : REGEXP \times REGEXP \rightarrow REGEXP$ --iteration one or more times axiom $s \in x y \equiv s \in x \vee s \in y$ theorem $x 0 = 0 x = x$ axiom $s \in x \cdot y \equiv (\exists t, u s = t \cdot u \bullet t \in x \wedge u \in y)$ --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 $“^” : REGEXP \times \mathbb{N} \rightarrow REGEXP$ --use this operator by raising the second argument like an exponent axiom $x^n = (\cdot i 0 \leq i \leq n \bullet x)$ --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

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

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

$$FLOAT_STRING = '+' Inf \quad (1)$$

$$| '-' Inf \quad (2)$$

$$| NaN \quad (3)$$

$$| ('-' | '+' | \epsilon) \cdot (* d \cdot '.' | \epsilon) \cdot * d \cdot (('e' \cdot ('-' | \epsilon) \cdot ^+ d) | \epsilon) \quad (4)$$

$$d = '0' | '1' | \dots | '9' \quad (5)$$

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