Runtime Monitoring

In this exercise sheet, we will explore the basics of runtime monitoring. A runtime monitor receives the events that a running system emits and reports every violation of a specification. At any moment, it receives only a finite *prefix* of the run, so it must decide from this prefix which violations have occurred, and it must report each of them as soon as the growing prefix enables it. In this sheet, we build such a monitor for metric first-order temporal logic (MFOTL), following Verimon's approach [2, 3]. We proceed as follows:

1. *Traces* (Section 1). Monitor's inputs: time-stamped sets of events.
2. *First-order formulas* (Section 2). Formulas about one time-point and their tables.
3. *Monitor's outputs* (Section 3). The tables of first-order formulas and their operations.
4. *Temporal operators* (Section 4). Formulas about several time-points and their tables.
5. *Examples* (Section 5). Monitoring two policies by hand.
6. *The algorithm* (Section 6). How the monitor updates its state with each new database.

1 Traces, time-points and time-stamps

A monitor does not receive the monitored system as input. Instead, it receives the events of the system's behaviour. An *event* consists of a name p and a list of *parameters* $d_1, \ldots, d_k$ from a domain $\mathbb{D}$ of values. Formally, it is a pair $(p, (d_1, \ldots, d_k)) \in \mathbb{E}$, where $\mathbb{E}$ is the set of all such pairs, but we will write them as $p[d_1, \ldots, d_k]$. A *database* $\Gamma \in \mathcal{P}(\mathbb{E})$ is a finite set of events. At each *time-point* $i \in \mathbb{N}$, the monitor receives simultaneous events as a single database, together with a *time-stamp* $\tau \in \mathbb{N}$. Thus, a *trace* is a function $\sigma : \mathbb{N} \to \mathcal{P}(\mathbb{E}) \times \mathbb{N}$ that maps each time-point to a pair $\sigma(i) = (\Gamma_i, \tau_i)$. We write it unfolded as

$$\sigma = (\Gamma_0, \tau_0),\ (\Gamma_1, \tau_1),\ (\Gamma_2, \tau_2),\ \ldots$$

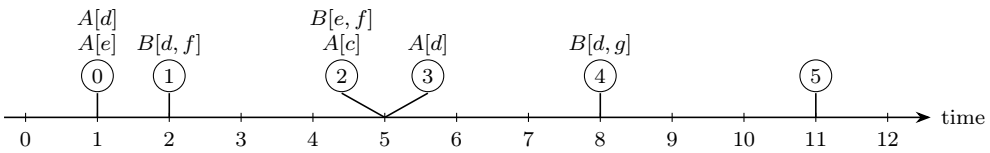
The time-stamps of a trace must satisfy two constraints [1]:

(T1) *Monotonicity:* $\tau_i \leq \tau_{i+1}$ for every $i \in \mathbb{N}$.

(T2) *Unboundedness:* for every $t \in \mathbb{N}$ there is an $i \in \mathbb{N}$ with $t \leq \tau_i$.

A *prefix* of length m is the finite sequence $(\Gamma_0, \tau_0), \ldots, (\Gamma_{m-1}, \tau_{m-1})$ of the first m pairs of a trace. After m steps, a monitor has received a prefix of length m.

Exercise 1 (Reading a trace). The diagram shows the beginning of the trace σ_{ex}. The axis shows time units. Each circle is a time-point, connected to its time-stamp, with its database above it. Complete the table. The first column is the example.



Time-point i	0	1	2	3	4	5
Time-stamp τ_i	1	____	____	____	____	____
Database Γ_i	$\{A[d], A[e]\}$	_____	_____	_____	_____	____

Exercise 2 (Checking the constraints). Indicate with a checkmark (✓) or a cross (×) whether each sequence satisfies (T1) and (T2). For every cross, write a counterexample: an i with $\tau_i > \tau_{i+1}$, or a bound t with $\tau_i < t$ for all i. The first row is the example.

Time-stamps $\tau_0, \tau_1, \ldots$	(T1)?	Counterexample	(T2)?	Counterexample
$0, 1, 2, 3, \ldots, i, \ldots$	(✓)		(✓)	
$1, 2, 5, 5, 8, 11, \ldots, 3i-4, \ldots$	( )	_____	( )	_____
$0, 0, 0, 0, \ldots, 0, \ldots$	( )	_____	( )	_____
$3, 2, 3, 4, \ldots, i+1, \ldots$	( )	_____	( )	_____
$1, 2, 1, 2, \ldots, 1+(i \bmod 2), \ldots$	( )	_____	( )	_____
$0, 0, 1, 1, 2, 2, \ldots, \lfloor i/2 \rfloor, \ldots$	( )	_____	( )	_____

Which of these sequences could be the time-stamps of a trace? _____

2 First-order formulas

As in model checking, we need a language to describe the properties that hold at a given time-point. We will use first-order specifications/formulas φ as inputs to the monitor, and we expect it to output the values for which the specification holds. We start by defining the syntax. Let $\mathbb{V}$ be a set of variables. The *terms* t are variables $x \in \mathbb{V}$ and constants $c \in \mathbb{D}$. A *predicate* consists of a name p and a list of terms $t_1, \ldots, t_k$. Like an event, it is formally a pair $(p, (t_1, \ldots, t_k))$, written $p[t_1, \ldots, t_k]$, and formulas are built from predicates and equations:

$$t ::= x \mid c, \quad \text{and} \quad \varphi ::= p[t_1, \ldots, t_k] \mid t_1 = t_2 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x.\, \varphi.$$

We will abbreviate $c = c$ as $\top$ for a fixed constant c and recursively define the *free variables* of a term and of a formula as

$$\begin{aligned} \mathrm{fv}(x) &= \{x\}, & \mathrm{fv}(c) &= \emptyset, \\ \mathrm{fv}(p[t_1, \ldots, t_k]) &= \mathrm{fv}(t_1) \cup \cdots \cup \mathrm{fv}(t_k), & \mathrm{fv}(t_1 = t_2) &= \mathrm{fv}(t_1) \cup \mathrm{fv}(t_2), \\ \mathrm{fv}(\neg\varphi) &= \mathrm{fv}(\varphi), & \mathrm{fv}(\varphi \wedge \psi) &= \mathrm{fv}(\varphi \vee \psi) = \mathrm{fv}(\varphi) \cup \mathrm{fv}(\psi), \\ \mathrm{fv}(\exists x.\, \varphi) &= \mathrm{fv}(\varphi) \setminus \{x\}. \end{aligned}$$

Exercise 3 (Free variables). Indicate with a checkmark (✓) or a cross (×) whether the given set is the set of free variables of each formula. For every cross, write the set of free variables. The first row is the example.

	Formula φ	Set	Is it $\mathrm{fv}(\varphi)$?	If not, compute $\mathrm{fv}(\varphi)$
	$P[x, y]$	$\{x, y\}$	(✓)	
1.	$P[x, 3]$	$\{x, 3\}$	( )	_____
2.	$\exists y.\, P[x, y]$	$\{x, y\}$	( )	_____
3.	$(\exists x.\, P[x, y]) \wedge Q[x]$	$\{y\}$	( )	_____
4.	$\exists x.\, (P[x, y] \wedge Q[x])$	$\{y\}$	( )	_____
5.	$\neg Q[z] \wedge x = 3$	$\{x, z\}$	( )	_____
6.	$\exists x.\, \exists y.\, P[x, y]$	$\emptyset$	( )	_____
7.	$\exists z.\, Q[x]$	$\{x\}$	( )	_____
8.	$P[x, y] \vee \neg Q[z]$	$\{x, y\}$	( )	_____

A *valuation* $v : \mathbb{V} \to \mathbb{D}$ assigns a value to every variable. We extend it to terms with $v(c) = c$, and $v[x \mapsto d]$ is the valuation that maps x to d and every other variable y to $v(y)$. The satisfaction of a formula at a time-point i of a trace σ is

$$\begin{aligned}
\langle \sigma, v, i \rangle \models p[t_1, \ldots, t_k] &\iff p[v(t_1), \ldots, v(t_k)] \in \Gamma_i, \\
\langle \sigma, v, i \rangle \models t_1 = t_2 &\iff v(t_1) = v(t_2), \\
\langle \sigma, v, i \rangle \models \neg\varphi &\iff \text{not } \langle \sigma, v, i \rangle \models \varphi, \\
\langle \sigma, v, i \rangle \models \varphi \wedge \psi &\iff \langle \sigma, v, i \rangle \models \varphi \text{ and } \langle \sigma, v, i \rangle \models \psi, \\
\langle \sigma, v, i \rangle \models \varphi \vee \psi &\iff \langle \sigma, v, i \rangle \models \varphi \text{ or } \langle \sigma, v, i \rangle \models \psi, \\
\langle \sigma, v, i \rangle \models \exists x.\, \varphi &\iff \langle \sigma, v[x \mapsto d], i \rangle \models \varphi \text{ for some } d \in \mathbb{D}.
\end{aligned}$$

Exercise 4 (Satisfaction at a time-point). Let $\mathbb{D} = \mathbb{N}$. A trace starts with

$$\Gamma_0 = \{P[1,2],\ P[1,3],\ P[2,3],\ Q[1],\ Q[3]\} \qquad \text{and} \qquad \Gamma_1 = \{P[2,1],\ Q[2]\}.$$

Indicate with a checkmark (✓) or a cross (×) whether $\langle \sigma, v, i \rangle \models \varphi$ holds in each row, where the valuation v has the given values of x and y. The first row is the example.

	i	Formula φ	$v(x)$	$v(y)$	$(\langle \sigma, v, i \rangle \models \varphi)$?
	0	$P[x, y]$	1	2	(✓)
1.	0	$P[x, y]$	2	1	( )
2.	1	$P[x, y]$	2	1	( )
3.	0	$P[x, y] \wedge Q[y]$	1	3	( )
4.	0	$P[x, y] \wedge \neg Q[y]$	1	3	( )
5.	1	$P[x, y] \wedge \neg Q[y]$	2	1	( )
6.	0	$\exists y.\, P[x, y]$	2	7	( )
7.	1	$\exists y.\, P[y, x]$	2	7	( )
8.	0	$P[x, y] \vee Q[x]$	3	7	( )
9.	0	$\exists x.\, (P[x, y] \wedge Q[x])$	2	3	( )
10.	1	$\neg Q[x] \wedge x = y$	1	1	( )

Why does the answer in row 6 not depend on $v(y)$, and the answer in row 9 not on $v(x)$?

Tuples. For input φ at time-point i, a monitor should output the values of the variables in $\mathrm{fv}(\varphi)$ for which φ holds. In this sheet, the monitor records these values as a tuple with one entry per variable of φ. We number these variables $0, 1, \ldots, n-1$ and write x, y, z for the variables 0, 1, 2. A *tuple* of length n is a list $v = [v_0, \ldots, v_{n-1}]$ whose entries v_i are values of $\mathbb{D}$ or the symbol `None` [1]. If an entry is `None`, it indicates that the value of the variable is not recorded. The notation $v\,!\,k$ represents the kth entry v_k of v. For a set X of variables, a tuple is *well-formed for* X if the variables in X have values:

$$\textit{wf-tuple}\, n\, X\, v \iff \text{length}\, v = n \text{ and } (v\,!\,k = \texttt{None} \iff k \notin X)\text{, for all } k < n.$$

Exercise 5 (Valuations as tuples).

Part 1 — *From valuations to tuples.* Let $n = 3$, and let $v(x) = 1$, $v(y) = 2$ and $v(z) = 5$. For each set X, write the tuple that is well-formed for X and records the values of v for the variables in X. The first column is the example.

X	$\{x, z\}$	$\{y\}$	$\{x, y, z\}$	$\mathrm{fv}(\exists y.\, P[x, y])$	$\emptyset$
Tuple of v	$[1, \texttt{None}, 5]$	_____	_____	_____	_____

Part 2 — *Well-formed tuples.* Indicate with a checkmark (✓) or a cross (×) whether *wf-tuple* $3\,\{0, 2\}\,v$ holds, and give a reason for every cross. The first row is the example.

Tuple v	Well-formed?	Reason
$[1, \texttt{None}, 2]$	(✓)	
$[1, 2, 3]$	( )	_____
$[\texttt{None}, \texttt{None}, 2]$	( )	_____
$[1, \texttt{None}]$	( )	_____
$[d, \texttt{None}, d]$	( )	_____
$[\texttt{None}, \texttt{None}, \texttt{None}]$	( )	_____

Tables. A *table* with *attributes* X is a finite set of tuples that are well-formed for X. A database system draws it with one column per attribute and one row per tuple. For example, with $n = 3$, the table

x	y
1	2
1	3
2	3

is the set $\{[1, 2, \texttt{None}],\ [1, 3, \texttt{None}],\ [2, 3, \texttt{None}]\}$ with attributes $\{x, y\}$.

The *empty table* $\emptyset$ has no rows. The tuple $\langle\rangle_n = [\texttt{None}, \ldots, \texttt{None}]$ is the only tuple that is well-formed for $\emptyset$, so the *unit table* $\mathbf{1}_n = \{\langle\rangle_n\}$ is the table with no columns and one row. At time-point i, the monitor outputs the set of tuples [1]

$$[\![\varphi]\!]_i = [\![\varphi]\!]_{i,n}^{\mathrm{fv}(\varphi)} = \{\, v \mid \langle \sigma, v, i\rangle \models \varphi \ \text{ and } \ \textit{wf-tuple}\, n\, (\mathrm{fv}(\varphi))\, v \,\},$$

where the entries `None` of v may stand for any value, since they belong to variables that are not free in φ. It is a table if it is finite.

Exercise 6 (Tables as sets of tuples).

Part 1 — *From rows to tuples.* Let $n = 3$. Write the following tables, with the attributes $\{y, z\}$ and $\{y\}$, as sets of tuples.

y	z
b	d
c	d
c	e

y
c

$B =$ _____, $C =$ _____

Part 2 — *Tables and their attributes.* Let $n = 3$. Indicate with a checkmark (✓) or a cross (×) whether each set is a table, and give its attributes. The first row is the example.

Set of tuples	Table?	Attributes
$\{[1, \texttt{None}, 2],\ [3, \texttt{None}, 3]\}$	(✓)	$\{x, z\}$
$\{[1, \texttt{None}, 2],\ [3, 4, \texttt{None}]\}$	( )	_____
$\{[\texttt{None}, b, \texttt{None}],\ [\texttt{None}, c, \texttt{None}]\}$	( )	_____
$\{[\texttt{None}, \texttt{None}, \texttt{None}]\}$	( )	_____
$\{[a, b, \texttt{None}],\ [a, b]\}$	( )	_____

Part 3 — *Tables of formulas.* Let $n = 2$ and $\mathbb{D} = \mathbb{N}$, and let the database at time-point 0 be, as in Exercise 4,

$$\Gamma_0 = \{P[1, 2],\ P[1, 3],\ P[2, 3],\ Q[1],\ Q[3]\}.$$

Compute the tables. The first one is the example.

$$[\![P[1, y]]\!]_0 = \{[\texttt{None}, 2],\ [\texttt{None}, 3]\}, \qquad [\![\exists y.\, P[y, x]]\!]_0 = \text{_____},$$

$$[\![P[x, 1]]\!]_0 = \text{_____}, \qquad [\![\exists x.\, Q[x]]\!]_0 = \text{_____}, \qquad [\![Q[2]]\!]_0 = \text{_____}.$$

The last two formulas have no free variables. How does their table show whether they hold?

---

3 Monitor's outputs for first-order formulas

At each time-point i, the monitor outputs the table $[\![\varphi]\!]_i$ of the monitored formula. This section shows how to compute it from the tables of the predicates and equations.

Exercise 7 (Formulas and their tables). Let $n = 2$ and $\mathbb{D} = \mathbb{N}$, and let the database at time-point 0 be, as in Exercise 4,

$$\Gamma_0 = \{P[1,2],\ P[1,3],\ P[2,3],\ Q[1],\ Q[3]\}.$$

Match each formula φ with its table $[\![\varphi]\!]_0$, drawn with one column per attribute: write the formula's letter in the brackets under the table. An example is provided.

a) $P[x,y]$

b) $P[x,y] \wedge Q[x]$

c) $P[x,y] \wedge \neg Q[y]$

d) $\exists y.\, P[x,y]$

e) $(\exists y.\, P[x,y]) \vee Q[x]$

f) $x = 3$

g) $\exists x.\, P[x,y]$

x
1
2
3

( )

x	y
1	2

( )

x	y
1	2
1	3
2	3

(a)

y
2
3

( )

x
1
2

( )

x
3

( )

x	y
1	2
1	3

( )

The valuations that satisfy $P[x,y] \vee Q[x]$ at time-point 0 cannot be written as a table. Why?

Conjunction and join. The monitor computes the table of a formula from the tables of its subformulas. A valuation satisfies $\varphi \wedge \psi$ if it satisfies φ and ψ, so each tuple of $[\![\varphi \wedge \psi]\!]_i$ combines a tuple of $[\![\varphi]\!]_i$ with a tuple of $[\![\psi]\!]_i$ that has the same values for the variables free in both. Two tuples u and w of length n *agree* if they coincide at every position where both are non-`None`. Formally, for every $k < n$, $u\,!\,k = \texttt{None}$ or $w\,!\,k = \texttt{None}$ or $u\,!\,k = w\,!\,k$. Their *join* $u \bowtie w$ is the tuple of length n with $(u \bowtie w)\,!\,k = u\,!\,k$ if $u\,!\,k \neq \texttt{None}$, and $(u \bowtie w)\,!\,k = w\,!\,k$ otherwise. [2, 3] The *natural join* of two tables is

$$A \bowtie B = \{\, u \bowtie w \mid u \in A,\ \text{and } w \in B,\ \text{and } u \text{ and } w \text{ agree} \,\},$$

and its attributes are those of A together with those of B. Thus $[\![\varphi \wedge \psi]\!]_i = [\![\varphi]\!]_i \bowtie [\![\psi]\!]_i$ [1].

Exercise 8 (Joining tables). Let $n = 3$ and consider the tables

$$A = \{[a, b, \texttt{None}],\ [a, c, \texttt{None}],\ [b, c, \texttt{None}]\},$$

$$B = \{[\texttt{None}, b, d],\ [\texttt{None}, c, d],\ [\texttt{None}, c, e]\}, \qquad C = \{[\texttt{None}, c, \texttt{None}]\} \qquad \text{(as in Exercise 6)}.$$

Part 1 — *Attributes.* Write the attributes of each table.

Table A: _____, table B: _____, table C: _____.

Part 2 — *Join.* Compute

$A \bowtie B =$ _____.

Part 3 — *A conjunction.* With $n = 2$, write $[\![Q[x]]\!]_0$ and check that $[\![P[x,y]]\!]_0 \bowtie [\![Q[x]]\!]_0$ consists of the rows of the table of $P[x,y] \wedge Q[x]$ in Exercise 7:

$[\![Q[x]]\!]_0 =$ _____.

More operations. To define the other connectives, we need three more operations [2, 3]:

$$A \triangleright B = A \setminus (A \bowtie B) \qquad \text{(antijoin)},$$
$$A[x \mapsto \texttt{None}] = \{\, v[x \mapsto \texttt{None}] \mid v \in A \,\} \qquad \text{(projection)},$$
$$A \cup B = \{v \mid v \in A \text{ or } v \in B\} \qquad \text{(union)},$$

where $v[x \mapsto \texttt{None}]$ is the tuple with $v[x \mapsto \texttt{None}] \,!\, x = \texttt{None}$ and $v[x \mapsto \texttt{None}] \,!\, y = v \,!\, y$ for every $y \neq x$. When the attributes of B are among those of A, the antijoin $A \triangleright B$ consists of the tuples of A that do not agree with any tuple of B. Two tables are *union-compatible* if they have the same attributes, so that their union is again a table.

Exercise 9 (Antijoin and projection).

Part 1 — *Operations.* For the tables of Exercise 8, compute

$$A \triangleright C = _____, \qquad A[y \mapsto \texttt{None}] = _____.$$

Part 2 — *Empty and unit tables.* Compute, for $n = 3$,

$$A \bowtie \mathbf{1}_3 = _____, \quad A \bowtie \emptyset = _____, \quad A \triangleright \emptyset = _____, \quad A \triangleright \mathbf{1}_3 = _____.$$

Then complete the sentence with "true" or "false". With respect to $\bowtie$ and $\triangleright$, the unit table behaves like the truth value _____ and, the empty table, like the truth value _____.

Exercise 10 (Connectives as table operations). Match each formula on the left with its table at time-point i on the right: write the formula's letter in the brackets.

a) $\varphi \wedge \psi$

b) $\varphi \wedge \neg\psi$

c) $\varphi \vee \psi$

d) $\exists x.\, \varphi$

e) $x = c$

f) $c = c'$ (constants)

( ) $\{v \mid v \,!\, x = c$ and $v \,!\, y = \texttt{None}$ for all $y \neq x\}$

( ) $[\![\varphi]\!]_i \cup [\![\psi]\!]_i$

( ) $\mathbf{1}_n$ if $c = c'$, and $\emptyset$ otherwise

(a) $[\![\varphi]\!]_i \bowtie [\![\psi]\!]_i$

( ) $[\![\varphi]\!]_i[x \mapsto \texttt{None}]$

( ) $[\![\varphi]\!]_i \triangleright [\![\psi]\!]_i$

Exercise 11 (Why some sets are infinite). Complete the table for the database

$$\Gamma_0 = \{P[1,2],\ P[1,3],\ P[2,3],\ Q[1],\ Q[3]\}.$$

For each formula, describe the shape of all tuples of $[\![\varphi]\!]_0$. The tuples have length $n = 2$, except for the third formula, where $n = 3$.

Formula	Shape of all tuples
$P[x,y] \vee Q[x]$	$[1,k]$ and $[3,k]$ for all k, and $[2,3]$
$\neg Q[x]$	_____
$P[x,y] \wedge \neg Q[z]$	_____
$x = y$	_____

Complete the sentence with either "$\mathrm{fv}(\psi) \subseteq \mathrm{fv}(\varphi)$" or "$\mathrm{fv}(\varphi) = \mathrm{fv}(\psi)$", one in each blank: The equation for $\varphi \wedge \neg\psi$ of Exercise 10 needs the condition _____, and the one for $\varphi \vee \psi$ needs the condition _____.

These two conditions are part of the definition of *safe* formulas below. These constraints define a collection of formulas whose set of satisfactions is computable [2, 3].

- Every predicate $p[t_1, \ldots, t_k]$ is safe. An equation $t_1 = t_2$ is safe if one of the terms is a constant, and $\neg(c = c')$ is safe for constants c, c'.
- The formula $\varphi \wedge \psi$ is safe if φ and ψ are safe. The formula $\varphi \wedge \neg\psi$, with the negation on the right, is safe if φ and ψ are safe and $\mathrm{fv}(\psi) \subseteq \mathrm{fv}(\varphi)$.
- The formula $\varphi \vee \psi$ is safe if φ and ψ are safe and $\mathrm{fv}(\varphi) = \mathrm{fv}(\psi)$.
- The formula $\exists x.\, \varphi$ is safe if φ is safe.
- No other formula is safe.

For a safe formula, $[\![\varphi]\!]_i$ is a finite table, and the equations of Exercise 10 compute it from the tables of the predicates and equations in the formula.

Exercise 12 (Safe or not). Indicate with a checkmark (✓) or a cross (×) whether each formula is safe. For each safe formula, compute $[\![\varphi]\!]_0$ for the database

$$\Gamma_0 = \{P[1,2],\ P[1,3],\ P[2,3],\ Q[1],\ Q[3]\}.$$

The first row is the example.

Formula	Safe?	Table
$P[x,y] \wedge \neg Q[x]$	(✓)	$\{[2,3]\}$
$Q[x] \wedge \neg P[x,y]$	( )	_____
$\exists y.\, (P[x,y] \wedge \neg Q[y])$	( )	_____
$Q[x] \vee \exists y.\, P[y,x]$	( )	_____
$x = 3 \vee Q[x]$	( )	_____
$Q[x] \vee Q[y]$	( )	_____
$\neg\neg Q[x]$	( )	_____

Exercise 13 (Why safety?). Mark with a checkmark (✓) the reason why a monitor that outputs tables accepts only safe formulas.

( ) Unsafe formulas are unsatisfiable, so the monitor never has anything to report.

( ) The rules are the conditions of the table operations, so safe formulas have finite tables.

( ) A negation can never be computed with tables, whatever their attributes.

( ) Safe formulas are safety properties as in model checking: nothing bad ever happens.

4 Temporal operators

Here, we use metric first-order temporal logic (MFOTL) to relate events at different time-points. It distinguishes itself from other temporal logics by using intervals. An *interval* I could be $I = [a, b] = \{t \in \mathbb{N} \mid a \leq t \leq b\}$ or $I = [a, d) = \{t \in \mathbb{N} \mid a \leq t < d\}$ with $a, b \in \mathbb{N}$ and $d \in \mathbb{N} \cup \{\infty\}$. MFOTL adds four temporal operators with an interval I [4, 1]:

$$\varphi \;::=\; \cdots \mid \mathrm{Y}_I\,\varphi \mid \mathrm{X}_I\,\varphi \mid \varphi\,\mathrm{S}_I\,\varphi \mid \varphi\,\mathrm{U}_I\,\varphi$$

That is, *previous*, *next*, *since* and *until*. Their semantics are given by:

$$\begin{aligned}
\langle \sigma, v, i\rangle \models \mathrm{Y}_I\,\varphi &\iff i > 0, \quad \tau_i - \tau_{i-1} \in I \;\text{ and }\; \langle \sigma, v, i-1\rangle \models \varphi,\\
\langle \sigma, v, i\rangle \models \mathrm{X}_I\,\varphi &\iff \tau_{i+1} - \tau_i \in I \;\text{ and }\; \langle \sigma, v, i+1\rangle \models \varphi,\\
\langle \sigma, v, i\rangle \models \varphi\,\mathrm{S}_I\,\psi &\iff \text{there is a } j \leq i \text{ such that } \tau_i - \tau_j \in I, \quad \langle \sigma, v, j\rangle \models \psi\\
&\qquad \text{and } \langle \sigma, v, k\rangle \models \varphi \text{ for all } k \in (j, i],\\
\langle \sigma, v, i\rangle \models \varphi\,\mathrm{U}_I\,\psi &\iff \text{there is a } j \geq i \text{ such that } \tau_j - \tau_i \in I, \quad \langle \sigma, v, j\rangle \models \psi\\
&\qquad \text{and } \langle \sigma, v, k\rangle \models \varphi \text{ for all } k \in [i, j).
\end{aligned}$$

Here $(j, i] = \{k \mid j < k \leq i\}$. The free variables of $\varphi \mathrm{S}_I \psi$ and $\varphi \mathrm{U}_I \psi$ are those of φ and ψ. The operators *eventually* and *once* are abbreviated as usual: $\mathrm{F}_I\,\varphi = \top\,\mathrm{U}_I\,\varphi$ and $\mathrm{P}_I\,\varphi = \top\,\mathrm{S}_I\,\varphi$.

Exercise 14 (Diagrams and formulas). Each diagram below shows a prefix with events $p[\,]$ and $q[\,]$ without parameters (written p and q) and a boxed time-point. In every diagram, the prefix continues with one more time-point at time 20 without events. Each formula below holds at the boxed time-point of exactly one diagram. Write the formula's letter in the brackets under that diagram. An example is provided.

a) $\mathrm{Y}_{[2,3]}\ p$

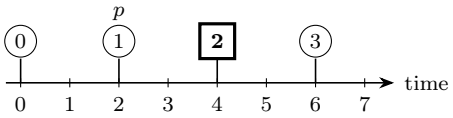
b) $\mathrm{X}_{[0,1]}\ q$

c) $p\,\mathrm{S}_{[2,4]}\ q$

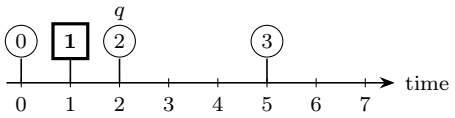
d) $p\,\mathrm{U}_{[1,3]}\ q$

e) $\mathrm{P}_{[0,2]}\ q$

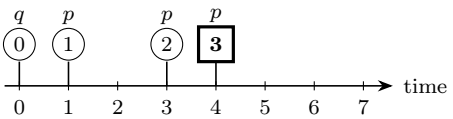
f) $\mathrm{F}_{[3,3]}\ q$



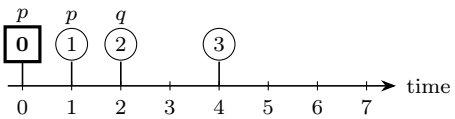
(a)



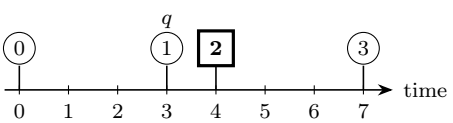
( )



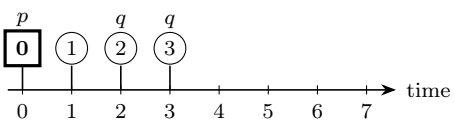
( )



( )



( )



( )

Tables of temporal formulas. To compute the table of $\varphi \,\mathrm{S}_I\, \psi$ at i, the monitor must first know at which time-points j the formula ψ may hold. These are the time-points up to i whose distance to i lies in I (for until, from i on), and we collect them in the sets [1]

$$\downarrow_I i = \{\, j \le i \mid \tau_i - \tau_j \in I \,\} \qquad \text{and} \qquad \uparrow_I i = \{\, j \ge i \mid \tau_j - \tau_i \in I \,\}.$$

Exercise 15 (Time-points within a distance). Compute the following sets for the trace σ_{ex} of Exercise 1. The first row is the example.

i	I	$\downarrow_I i$	i	I	$\uparrow_I i$
4	$[3,7]$	$\{0,1,2,3\}$	0	$[1,2]$	_____
2	$[0,3]$	_____	1	$[0,3]$	_____
3	$[0,0]$	_____	2	$[1,2]$	_____
1	$[2,\infty]$	_____	3	$[1,4]$	_____

Each candidate $j \in \downarrow_I i$ gives one way to satisfy $\varphi \,\mathrm{S}_I\, \psi$ at i: ψ at j, and φ at $j+1, \ldots, i$. If $\mathrm{fv}(\varphi) \subseteq \mathrm{fv}(\psi)$, we replace every "and" by a join and every "or" between the candidates by a union. For example, in the first row of Exercise 15, $\downarrow_{[3,7]} 4 = \{0,1,2,3\}$, so

$$\begin{aligned}[\![\varphi \,\mathrm{S}_{[3,7]}\, \psi]\!]_4 = {} & \big([\![\psi]\!]_3 \bowtie [\![\varphi]\!]_4\big) \cup \big([\![\psi]\!]_2 \bowtie [\![\varphi]\!]_3 \bowtie [\![\varphi]\!]_4\big) \\ & \cup \big([\![\psi]\!]_1 \bowtie [\![\varphi]\!]_2 \bowtie [\![\varphi]\!]_3 \bowtie [\![\varphi]\!]_4\big) \cup \big([\![\psi]\!]_0 \bowtie [\![\varphi]\!]_1 \bowtie \cdots \bowtie [\![\varphi]\!]_4\big).\end{aligned}$$

There is no term for $j = 4$, because $\tau_4 - \tau_4 = 0 \notin [3,7]$. To write such tables in general, we write $\bowtie_{k \in K} T_k$ for the join of tables T_k indexed by a finite set K (like $\bigcup$ and $\bigcap$) [1]. Thus[1]

$$[\![\varphi \,\mathrm{S}_I\, \psi]\!]_i \;=\; \bigcup_{j \in \downarrow_I i} \Big([\![\psi]\!]_j \bowtie \bowtie_{k \in (j,i]} [\![\varphi]\!]_k \Big).$$

Exercise 16 (Temporal operators as table operations). Assume that $\mathrm{fv}(\varphi) \subseteq \mathrm{fv}(\psi)$ and $\mathrm{fv}(\varphi') \subseteq \mathrm{fv}(\psi)$. Match each formula on the left with its table at time-point i on the right: write the formula's letter in the brackets.

a) $\varphi \,\mathrm{S}_I\, \psi$

b) $\varphi \,\mathrm{U}_I\, \psi$

c) $\mathrm{Y}_I\, \varphi$

d) $\mathrm{X}_I\, \varphi$

e) $\mathrm{P}_I\, \varphi$

f) $\mathrm{F}_I\, \varphi$

g) $(\neg\varphi') \,\mathrm{S}_I\, \psi$

( ) $\displaystyle\bigcup_{j \in \uparrow_I i} [\![\varphi]\!]_j$

( ) $[\![\varphi]\!]_{i-1}$ if $i > 0$ and $\tau_i - \tau_{i-1} \in I$, and $\emptyset$ otherwise

( ) $\displaystyle\bigcup_{j \in \downarrow_I i} \Big([\![\psi]\!]_j \triangleright \bigcup_{k \in (j,i]} [\![\varphi']\!]_k \Big)$

(a) $\displaystyle\bigcup_{j \in \downarrow_I i} \Big([\![\psi]\!]_j \bowtie \bowtie_{k \in (j,i]} [\![\varphi]\!]_k \Big)$

( ) $\displaystyle\bigcup_{j \in \uparrow_I i} \Big([\![\psi]\!]_j \bowtie \bowtie_{k \in [i,j)} [\![\varphi]\!]_k \Big)$

( ) $\displaystyle\bigcup_{j \in \downarrow_I i} [\![\varphi]\!]_j$

( ) $[\![\varphi]\!]_{i+1}$ if $\tau_{i+1} - \tau_i \in I$, and $\emptyset$ otherwise

[1] Exercise for the curious: prove that the join over the empty set is $\mathbf{1}_n$. Then, the term for $j = i$, if $i \in \downarrow_I i$, is $[\![\psi]\!]_i \bowtie \mathbf{1}_n = [\![\psi]\!]_i$

Why does the table of g) use the tables of φ', and not those of $\neg\varphi'$?

Why does the table of a) need $\mathrm{fv}(\varphi) \subseteq \mathrm{fv}(\psi)$?

Hint. Look at the attributes of the terms of the union for different j.

The answers to these questions justify the extension of safety for temporal operators [2, 3]:

- The formulas $\mathrm{Y}_I\,\varphi$ and $\mathrm{X}_I\,\varphi$ are safe if φ is safe.
- The formulas $\varphi\,\mathrm{S}_I\,\psi$ and $\varphi\,\mathrm{U}_I\,\psi$ are safe if ψ is safe, $\mathrm{fv}(\varphi) \subseteq \mathrm{fv}(\psi)$, and φ is safe or the negation of a safe formula (then the monitor uses an antijoin).

5 Examples

To practice what we have learned, in this section, we monitor the policy

every event $A[x]$ is followed by some event $B[x, y]$ 1 to 2 time units later.

A monitor for this policy must report each time-point i and value x such that $A[x] \in \Gamma_i$ and no event $B[x, y]$ occurs 1 to 2 time units after τ_i.

Exercise 17 (Formalising the policy).

Part 1 — *Formalise.* We write $\varphi \to \psi$ for $\neg\varphi \vee \psi$. Using $\to$ and F_I, write a formula χ with the free variable x that a valuation satisfies at i if the policy holds for x at i.

$$\chi = _____$$

Part 2 — *Negate.* The monitor reports the violations, that is, the time-points and valuations that satisfy $\neg\chi$. Two formulas are *equivalent*, written $\varphi \equiv \psi$, if they have the same satisfying valuations at every time-point of every trace. Use the laws

$$\varphi \to \psi \equiv \neg\varphi \vee \psi, \qquad \neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi, \qquad \neg\neg\varphi \equiv \varphi$$

to rewrite $\neg\chi$ step by step into a formula φ_{ex} without $\to$ and without double negations.

$$\begin{aligned} \neg\chi &\equiv _____ \\ &\equiv _____ \\ &\equiv _____ \wedge \neg\,\mathrm{F}_{[1,2]}\ _____ = \varphi_{ex} \end{aligned}$$

Part 3 — *Safety.* Check with the safety rules that φ_{ex} is safe.

Why can the monitor not use χ directly?

Exercise 18 (Violations manually). Evaluate the formula φ_{ex} of Exercise 17 on the trace σ_{ex} of Exercise 1, with the domain $\mathbb{D} = \{c, d, e, f, g\}$ and $n = 2$ (variables x and y). Assume that the trace continues after time-point 5 with time-stamps larger than 11 and no events named B. Complete the table. The first row is the example.

Hint. The table of $\mathrm{F}_{[1,2]}\ \exists y.\, B[x, y]$ at i collects the tables of $\exists y.\, B[x, y]$ at the time-points of $\uparrow_{[1,2]} i$.

i	$[\![A[x]]\!]_i$	$\uparrow_{[1,2]} i$	$[\![\mathrm{F}_{[1,2]}\ \exists y.\, B[x, y]]\!]_i$	$[\![\varphi_{ex}]\!]_i$
0	$\{[d, \texttt{None}],\ [e, \texttt{None}]\}$	$\{1\}$	$\{[d, \texttt{None}]\}$	$\{[e, \texttt{None}]\}$
1				
2				
3				

List the violations of the policy as pairs (time-point, value of x).

Violations: _____

Why is $(3, d)$ a violation, although the trace contains the event $B[d, g]$ after time-point 3?

Exercise 19 (Why the delay?). A monitor receives the trace one database at a time. It reports a violation only when the violation holds on every trace that extends the databases received so far. The violation $(2, c)$ holds because no event $B[c, y]$ occurs at time 6 or 7, that is, between 1 and 2 time units after $\tau_2 = 5$. A monitor for φ_{ex} would report $(2, c)$ after time-point 4, but not after time-point 3. Mark the reason with a checkmark (✓).

( ) After time-point 3, the table of $A[x]$ at time-point 2 can still change.

( ) After time-point 3, the next time-point can still have the time-stamp 6 or 7 and an event $B[c, y]$. Time-point 4 has the time-stamp 8, and later time-stamps are at least 8.

( ) The events $A[c]$ and $B[e, f]$ share a database, so $A[c]$ is evaluated later.

A formula is *monitorable* if it is safe and the interval of every until and every next in it has a finite upper bound [2, 3]. Crucially, for monitorable formulas, the table at each time-point depends only on a finite part of the trace, so the monitor can report every violation.

Exercise 20 (A policy about the past). Consider a second policy for the trace σ_{ex}:

every event $B[x, y]$ is preceded by some event $A[x]$ at most 3 time units earlier.

Part 1 — *Formalise.* Write a formula ψ_{ex} whose satisfying valuations at i are the violations of this policy at time-point i, using P_I, and check with the rules that it is monitorable.

$\psi_{ex} =$ _____

Part 2 — *Evaluate.* Compute the tables of ψ_{ex} on σ_{ex}. The first row is the example.
Hint. The table of $\mathrm{P}_{[0,3]}\ A[x]$ at i collects the tables of $A[x]$ at the time-points of $\downarrow_{[0,3]} i$.

i	$[\![B[x,y]]\!]_i$	$\downarrow_{[0,3]} i$	$[\![\mathrm{P}_{[0,3]}\ A[x]]\!]_i$	$[\![\psi_{ex}]\!]_i$
0	$\emptyset$	$\{0\}$	$\{[d, \texttt{None}], [e, \texttt{None}]\}$	$\emptyset$
1				
2				
3				
4				

6 The monitoring algorithm

Storing the entire prefix would be inefficient. Instead, the monitor stores a *state* with the information needed for future outputs. It updates the state with each incoming database, and outputs the tables that it can compute from the received prefix. We follow Verimon's verified algorithm [2, 3]. It consists of two main functions: minit_0, which translates a formula into the initial state, and meval, which updates the state with each database. We study them here. We write a list of tables or time-stamps as $[X_1, \ldots, X_k]$, with the empty list as $[\,]$, and list concatenation as $xs +\!\!+ ys$.

Progress. After time-point j (that is, after receiving the time-points $0, \ldots, j$), the *progress* $\rho_j(\varphi)$ is the number of time-points for which the monitor has computed the table of φ: it has computed $[\![\varphi]\!]_i$ exactly for $i < \rho_j(\varphi)$. Predicates and equations have the progress $j+1$, and the progress of an operator is at most that of its operands [2, 3]:

$$\rho_j(p[t_1, \ldots, t_k]) = \rho_j(t_1 = t_2) = j+1, \qquad \rho_j(\neg\varphi) = \rho_j(\exists x.\, \varphi) = \rho_j(\varphi),$$
$$\rho_j(\varphi \wedge \psi) = \rho_j(\varphi \vee \psi) = \rho_j(\varphi\ \mathrm{S}_I\ \psi) = \min(\rho_j(\varphi), \rho_j(\psi)),$$
$$\rho_j(\varphi\ \mathrm{U}_I\ \psi) = \min\{\, i \mid \tau_j - \tau_i \le t \text{ for some } t \in I \,\} \quad \text{if } \rho_j(\varphi) = \rho_j(\psi) = j+1.$$

If an operand has smaller progress $m \le j$, the time-stamp τ_m replaces τ_j.

Exercise 21 (Progress). Compute the progress of the subformulas of φ_{ex} on σ_{ex}, and the time-points of the tables of φ_{ex} that the monitor outputs after database j: those with $\rho_{j-1}(\varphi_{ex}) \le i < \rho_j(\varphi_{ex})$, where $\rho_{-1} = 0$. The first column is the example.

After time-point j	0	1	2	3	4
$\rho_j(A[x])$	1				
$\rho_j(\exists y.\, B[x,y])$	1				
$\rho_j(\mathrm{F}_{[1,2]}\ \exists y.\, B[x,y])$	0				
$\rho_j(\varphi_{ex})$	0				
Time-points of new $[\![\varphi_{ex}]\!]_i$	none				

States. The monitor's state is a triple $(i, \hat{\varphi}, n)$: a time-point i, a *formula state* $\hat{\varphi}$ resembling the input φ, and the number of variables n. The formula states are

$$\begin{aligned}\hat{\varphi} \;::=\; & \mathsf{Rel}(X) \mid \mathsf{Pred}(p, [t_1, \ldots, t_k]) \mid \mathsf{And}(\hat{\varphi}, pos, \hat{\psi}, buf) \mid \mathsf{Or}(\hat{\varphi}, \hat{\psi}, buf) \mid \mathsf{Ex}(x, \hat{\varphi}) \\ & \mid \mathsf{Prev}(I, \hat{\varphi}, first, buf, nts) \mid \mathsf{Next}(I, \hat{\varphi}, first, nts) \\ & \mid \mathsf{Since}(pos, \hat{\varphi}, I, \hat{\psi}, buf, nts, aux) \mid \mathsf{Until}(pos, \hat{\varphi}, I, \hat{\psi}, buf, nts, aux).\end{aligned}$$

Here X is a table. The flag *first* of previous and next is *true* until the first table of φ has been processed: previous outputs $\emptyset$ for time-point 0 and next discards the table of φ at time-point 0. The flag $pos \in \{true, false\}$ is *false* when the left operand is negated, so that the operator joins when $pos = true$ and antijoins otherwise. We write $X \bowtie_{pos} Y$ for $X \bowtie Y$ if $pos = true$ and for $X \triangleright Y$ if $pos = false$. The two operands of a binary operator may have different progress, so the *buffer* $buf = (xs, ys)$ stores two lists of tables: the tables of one operand for the time-points at which the other operand has no table yet. Previous has only one operand, so its buffer is a single list of tables of φ. The lists *nts* store the time-stamps of buffered time-points, and *aux* represents *auxiliary states* for since and until.

Exercise 22 (The initial state). The function minit_0 translates a monitorable formula into its formula state before the first database, with empty buffers $([], [])$ and empty lists. Write the letter of each case in the brackets of its initial state. An example is provided.

a)	$\text{minit}_0(\varphi \wedge \psi)$	( ) $\mathsf{Since}(false, \text{minit}_0(\varphi), I, \text{minit}_0(\psi), ([], []), [], [])$
b)	$\text{minit}_0(\varphi \wedge \neg\psi)$, $\text{fv}(\psi) \subseteq \text{fv}(\varphi)$	(a) $\mathsf{And}(\text{minit}_0(\varphi), true, \text{minit}_0(\psi), ([], []))$
c)	$\text{minit}_0(\varphi \, \mathrm{S}_I \, \psi)$, with φ safe	( ) $\mathsf{Ex}(x, \text{minit}_0(\varphi))$
d)	$\text{minit}_0((\neg\varphi) \, \mathrm{S}_I \, \psi)$	( ) $\mathsf{Rel}(\mathbf{1}_n)$
e)	$\text{minit}_0(c = c)$	( ) $\mathsf{Since}(true, \text{minit}_0(\varphi), I, \text{minit}_0(\psi), ([], []), [], [])$
f)	$\text{minit}_0(\exists x.\, \varphi)$	( ) $\mathsf{And}(\text{minit}_0(\varphi), false, \text{minit}_0(\psi), ([], []))$

Write the initial formula state of φ_{ex} (here $\top$ is $c = c$):

$$\text{minit}_0(\varphi_{ex}) = _____$$
$$\hat{\psi} = _____$$

Evaluation. Suppose that the monitor has received the time-points $0, \ldots, j-1$ and that its state is $(i, \hat{\varphi}, n)$. The tables of the time-points $0, \ldots, i-1$ have been output, and i is the earliest time-point whose table has not, that is, $i = \rho_{j-1}(\varphi)$. The *incoming* time-point j has the database Γ_j and the time-stamp τ_j. With them, meval updates the formula state $\hat{\varphi}$ to $\hat{\varphi}'$ and returns the list zs of the tables that it can now compute, chronologically: $(zs, \hat{\varphi}') = \text{meval}(\hat{\varphi})$. These are the tables of the time-points $i, \ldots, i+m-1$, where m is the length of zs and $i + m = \rho_j(\varphi)$, so $zs = [[\![\varphi]\!]_i, \ldots, [\![\varphi]\!]_{i+m-1}]$. The monitor outputs the pairs (i', v) with $v \in [\![\varphi]\!]_{i'}$ for $i \le i' < i + m$, and its new state is $(i + m, \hat{\varphi}', n)$.

Exercise 23 (Evaluating first-order states). Let $(xs, \hat{\varphi}') = \text{meval}(\hat{\varphi})$ and $(ys, \hat{\psi}') = \text{meval}(\hat{\psi})$ be the results for the sub-states, with $xs = [X_1, \ldots, X_k]$, and let the buffer be $buf = (xs_0, ys_0)$. The new formula state preserves the original constructor and contains the updated sub-(formula-)states and buffer, for example $\text{meval}(\mathsf{Or}(\hat{\varphi}, \hat{\psi}, buf)) = (zs, \mathsf{Or}(\hat{\varphi}', \hat{\psi}', buf'))$. A binary operator updates the buffer as follows: it appends the new tables of its operands to

its buffer, and then applies a table operation f (join or union) to the pairs of tables at the front of both lists. Recursively:

$$\text{take}_f\left([X] +\!\!+ xs,\ [Y] +\!\!+ ys\right) = \left([f(X,Y)] +\!\!+ zs,\ buf\right) \quad \text{where } (zs, buf) = \text{take}_f(xs, ys),$$
$$\text{take}_f(xs, ys) = \left([\,], (xs, ys)\right) \quad \text{if } xs \text{ or } ys \text{ is empty.}$$

Match each state on the left with the expression for the list zs of tables on the right that meval should return. Here $ts = [t_1, \ldots, t_k]$.

Hint. Exercise 10 gives the table operations.

a) $\mathsf{Rel}(X)$	( )	zs with $(zs, buf') = \text{take}_{\bowtie_{pos}}(xs_0 +\!\!+ xs,\ ys_0 +\!\!+ ys)$
b) $\mathsf{Pred}(p, ts)$	( )	$[X_1[x \mapsto \texttt{None}], \ldots, X_k[x \mapsto \texttt{None}]]$
c) $\mathsf{Or}(\hat{\varphi}, \hat{\psi}, buf)$	(a)	$[X]$
d) $\mathsf{And}(\hat{\varphi}, pos, \hat{\psi}, buf)$	( )	zs with $(zs, buf') = \text{take}_{\cup}(xs_0 +\!\!+ xs,\ ys_0 +\!\!+ ys)$
e) $\mathsf{Ex}(x, \hat{\varphi})$	( )	$[[\![p[t_1, \ldots, t_k]]\!]_j]$

References

[1] Jonathan Julián Huerta y Munive. Relaxing safety for metric first-order temporal logic via dynamic free variables. In *Runtime Verification (RV 2022)*, LNCS 13498. Springer, 2022.

[2] Joshua Schneider, David Basin, Srđan Krstić and Dmitriy Traytel. A formally verified monitor for metric first-order temporal logic. In *Runtime Verification (RV 2019)*, LNCS 11757, pages 310–328. Springer, 2019.

[3] Joshua Schneider and Dmitriy Traytel. Formalization of a monitoring algorithm for metric first-order temporal logic. *Archive of Formal Proofs*, 2019. https://isa-afp.org/entries/MFOTL_Monitor.html.

[4] David Basin, Felix Klaedtke, Samuel Müller and Eugen Zălinescu. Monitoring metric first-order temporal properties. *Journal of the ACM*, 62(2):15:1–15:45, 2015.