Reachability Analysis in Micro-Stipula

Cosimo Laneve
Department of Computer Science and Engineering, University of Bologna
Bologna, Italy
cosimo.laneve@unibo.it

ABSTRACT

Micro-Stipula is a stateful calculus defining clauses that may be either invoked by the external environment or triggered by time expressions. Because of the interplay between states, time and non-determinism, establishing whether a clause will be ever executed – the *reachability problem* – is difficult. In this paper we define an analyzer that spots unreachable clauses and demonstrate its soundness.

CCS CONCEPTS

• **Software and its engineering** → **State systems**; **Software verification**; **Automated static analysis**; **Semantics**; • **Theory of computation** → **Timed and hybrid models**.

KEYWORDS

Calculus with states, functions and events; Operational Semantics; Cyclic behaviours; Reachability analysis; Fixpoint analysis; Stipula.

ACM Reference Format:
Cosimo Laneve. 2024. Reachability Analysis in Micro-Stipula . In *Proceedings of the 26th Symposium on Principles and Practice of Declarative Programming, PPDP 2024, Milano, Italy, September 10-11, 2024.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/NNNNNNN.NNNNNNN

1 INTRODUCTION

Micro-*Stipula*, noted *μStipula*, is a basic calculus defining *contracts*, namely sets of clauses that are either (*a*) *parameterless functions*, to be invoked by the external environment, or (*b*) *events* that are triggered at given times. The calculus has been devised to study the presence of unreachable clauses in legal contracts written in *Stipula* [8, 9], that is clauses that can never be applied because of unreachable circumstances or of wrong time constraints. In the legal contract domain, removing such clauses when the contract is drawn up is substantial because they might be considered too oppressive by parties and make the legal relationship fail.

Spotting and dropping unreachable code is a very common optimization in compiler construction of programming languages and the literature already reports techniques for affording this issue [1, 5]. However, the presence of events in *μStipula* makes the optimization complex. In particular, when the time expressions are logically inconsistent with the contract behaviour, the corresponding event (and its continuation) becomes unreachable.

To address time anomalies, we define an analyzer that uses *logical times*, which are symbolic expressions that approximate a runtime entity at static-time: the clock value of the system. Then the analyzer computes the set of reachable clauses by means of a closure operation based on a fixpoint technique. In particular, a clause is added to the set only if the corresponding logical time is consistent with *the logical time of the computation* leading to that clause. When the constraints on logical times are unsolvable, the list of the corresponding clauses (which are therefore unreachable) is returned.

In this context, the main criticality is the presence of cyclic behaviours because they stem computations that are infinite. Hence, since the foregoing consistency check cannot be performed directly on the set of computations, we decided to use *linear traces*, which are surrogates of computations where clauses can occur at most once. While linear traces support static time arguments because they are finite, they are not expressive enough to faithfully describe computations. In particular, in *μStipula*, contracts may display very subtle reachability dependencies that are achieved by several instances of the same clause.

Therefore, in order to cover as much cases as possible, we opted for an algorithm combining time analysis and standard, untimed one: time analysis is used for acyclic clauses, while time expressions are overlooked in the other clauses. We demonstrate the soundness of the algorithm for the reachability analysis in *μStipula* and discuss the cases that do not allow us to attain completeness. We also extend the technique to cover time expressions with names that are instantiated at runtime (which is very common in *Stipula*). To cope with this feature, we consider logical times with expressions on such names and the analyzer returns *constraints* that guarantee reachability. In this last case, we take advantage of the (additive) format of time expressions to simplify and partially evaluate the constraints.

The reachability analyzer has been prototyped (actually it covers the full *Stipula* language) and the prototype, together with all the code samples discussed in this paper (and many others), is available at [11].

The paper is organized as follows. Section 2 gives a light introduction to *μStipula* and to the reachability analyzer through a bunch of simple examples. The syntax and semantics of *μStipula* are defined in Section 3. The simple theory underneath the analyzer and that overlooks time expressions is developed in Section 4; the full theory is discussed in Section 5. Section 6 covers the extension of *μStipula* with more expressive time expressions. We report the related literature in Section 7 and conclude in Section 8. To ease the reading, the proofs of the main results in the paper have been collected in the Appendix.

PPDP '24, September 10-11, 2024, Milano, Italy

ACM ISBN 9-8-4007-0969-297
https://doi.org/10.1145/NNNNNNN.NNNNNNN

2 *μSTIPULA* AND THE REACHABILITY ANALYZER

Let us introduce the features of *μStipula* with a simple contract, the PingPong:

```
1  stipula PingPong {
2     init Q0
3     @Q0 ping {
4         now + 1 ≫ @Q1 ⇒ @Q2
5     } ⇒ @Q1
6     @Q2 pong {
7         now + 2 ≫ @Q3 ⇒ @Q0
8     } ⇒ @Q3
9  }
```

The contract is defined by the keyword `stipula` and is initially in the state specified by the clause `init`. Two functions may be invoked by the external environment: `ping` and `pong`. In particular `ping` is invoked if the contract is in the state `Q0`, `pong` when the contract is in `Q2`. Functions (*i*) make the contract transit in the state specified by the term "⇒ `@Q`" (see lines 5 and 8) and (*ii*) make the events in their body to be scheduled. In particular, an event `now + k ≫ @Q ⇒ @Q'` (see lines 4 and 7) is a timed continuation that can run when the time *is equal* to `now + k` (`now` represents the clock value when the function is called) and the state of the contract is `Q`. The only effect of executing an event is the change of the state. When no event can be executed in a state either *the time elapses* (a tick occurs) or *a function can be invoked*. The elapsing of time does not modify a state.

In the `PingPong` contract, the initial state is `Q0` where only `ping` may be invoked; no event is present because they are created by executing functions. The invocation of `ping` makes the contract transit to `Q1` and creates the event at line 4, noted ev_4. In `Q1` there is still a unique possibility: executing ev_4. However, to execute it, it is necessary to wait 1 minute (one clock tick must elapse) – the time expression `now + 1`. Then the state becomes `Q2` indicating that `pong` may be invoked, thus letting the contract transit to `Q3` where, after 2 minutes (the expression `now + 2`), the event at line 7 can be executed and the contract returns to `Q0`. In `PingPong`, every clause is reachable; in fact the analyzer returns

```
"unreachable_code": []
```

To discuss the difficulties we found with the reachability analysis, let us consider more complex contracts. The following one

```
1  stipula Sample {
2     init Init
3     @Init f {
4         now + 5 ≫ @Go ⇒ @End
5     } ⇒ @Run
6     @Init g { } ⇒ @Go
7  }
```

consists of two functions at lines 3 and 6, called `f` and `g`, respectively. The two functions may be invoked in `Init`, however the invocation of one of them excludes the other because their final states are not `Init`. Therefore the event at line 4, which is inside `f`, is unreachable since it can run only if `g` is executed. In this case the analyzer returns:

```
"unreachable_code": [ Go.ev_4.End ] .
```

The difficult cases are those where unreachability is due to time expressions that are incompatible. Consider `SampleTime`:

```
1  stipula SampleTime {
2     init Init
3     @Init f {
4        now + 1 ≫ @Cont ⇒ @Run
5        now + 2 ≫ @Comp ⇒ @End
6     } ⇒ @Cont
7
8     @Run g {
9        now + 2 ≫ @Go ⇒ @Comp
10    } ⇒ @Go
11 }
```

This contract has an unreachable event: the one at line 5. To explain this, let us discuss the flow of execution by computing the times of the clauses. Let the time of `f` be ζ_{f}; therefore the times of ev_4 and ev_5 are $\zeta_{\mathsf{f}}+1$ and $\zeta_{\mathsf{f}}+2$, respectively. Because of the matching final state/initial state, the clause that is executed after `f` is ev_4 and then `g`. It is possible that `g` starts after some time the ev_4 is terminated, say ζ_{g} ticks; therefore the time becomes $\zeta_{\mathsf{f}}+\zeta_{\mathsf{g}}+1$; then ev_9 may be triggered at $\zeta_{\mathsf{f}}+\zeta_{\mathsf{g}}+3$. Since the final state of ev_9 matches with the initial state of ev_5, the time of ev_9 must be smaller or equal to that of ev_5. Hence we derive $\zeta_{\mathsf{f}}+\zeta_{\mathsf{g}}+3 \leq \zeta_{\mathsf{f}}+2$. This constraint is clearly *unsolvable* because all the time variables must be nonnegative. Said otherwise, when `ev.5` might be executed, the time is already elapsed. Hence the analyzer gives

```
"unreachable_code": [ Comp.ev_5.End ] .
```

The above arguments may lead to wrong conclusions when contracts have cycles. For example, in the contract `Ugly`

```
1  stipula Ugly {
2     init Q0
3     @Q0 f {
4         now + 1 ≫ @Q3 ⇒ @Q4
5         now + 2 ≫ @Q2 ⇒ @Q3
6     } ⇒ @Q1
7     @Q1 g {
8         now + 1 ≫ @Q1 ⇒ @Q2
9     } ⇒ @Q0
10    @Q4 h { } ⇒ @Q5
11 }
```

the ev_4 seems unreachable because it is caused by ev_5 that happens at a later time (as a consequence `C.h` is unreachable, as well). In fact, we derive the constraint $\zeta_{\mathsf{f}}+\zeta_{\mathsf{g}}+3 \leq \zeta_{\mathsf{f}}+\zeta_{\mathsf{g}}+2$, which is unsolvable. However, consider the following flow of execution assuming that the contract starts at time 0. The first clause that is executed is `f`, which creates ev_4 to be executed at time 1 (we note it ${\mathsf{ev}_4}^1$) and ev_5 to be executed at time 2 (we note it ${\mathsf{ev}_5}^2$). `f` ends in `Q1` and the next clause that can be executed is `g`, still at time 0. `g` creates ev_8 to be executed at 1 and the state go back to `Q0`. Now there is the critical moment: a tick occurs and `f` is executed again at time 1. `f` creates ev_4 to be executed at time 2 (we note it ${\mathsf{ev}_4}^2$) and ev_5 to be executed at time 3 (we note it ${\mathsf{ev}_5}^3$). Then we can execute ev_8, a tick, ${\mathsf{ev}_5}^2$, ev_4^2 and `h`. That is, `h` is reached by events created in two different invocations of `f`. Hence, every clause of `Ugly` is reachable.

Therefore, when a function is cyclic, we have decided to drop the time analysis and to deem "reachable" its events if they are so with arguments that do not take into account time expressions.

The prototype analyzer. There is a prototype implementation of the analyzer presented in this paper [11]. The prototype addresses the full *Stipula* language, therefore its input contracts are a bit more verbose. In particular, function names must be replaced by "`A:f()[]`" and events must have "`{ }`" after the initial state. The reader may try the examples in this paper (and many others in the github repository) as well as experiment her/his own ones.

3 SYNTAX AND SEMANTICS OF *μSTIPULA*

A contract is declared in *μStipula* by the term

$$\texttt{stipula C \{ init Q} \quad F \texttt{ \}}$$

where C is the name of the contract, Q is the *initial state* and a F is a sequence of *functions*. We use a set of *states*, ranged over $\texttt{Q}, \texttt{Q}', \cdots$; and a set of *function names* $\texttt{f}, \texttt{g}, \cdots$.

The syntax of functions F, events W and time expressions t is the following:

Functions	F	::=	_ | @Q f { W } ⇒ @Q′ F	
Events	W	::=	_ | t ≫ @Q ⇒ @Q′ W	
Time expressions	t	::=	now + κ	($\kappa \in$ Nat)

A *μStipula* contract may transit from one state to another either by invoking a *function* or by running an *event*. Functions are invoked by the *external environment* and define the state when the invocation is admitted.

Events W are sequences of *timed continuations* that are created by functions and schedule a transition in future execution. More precisely, the term $\texttt{t} \gg \texttt{@Q} \Rightarrow \texttt{@Q}'$ schedules a transition from Q to Q′ if, at a time that is the value of t, the contract is in the state Q. The time expressions are additions $\texttt{now} + \kappa$, where κ is a natural constant (representing *minutes*); now is a place-holder that will be replaced by the current global time during the execution, see rule [State-Change] in Table 1. We always shorten $\texttt{now} + \texttt{0}$ into now.

Restriction and notations. We assume that *a function is uniquely determined by the tuple* $\texttt{Q}\cdot\texttt{f}\cdot\texttt{Q}'$, that is the initial and final states and the function name. Similarly, an event is uniquely determined by the tuple $\texttt{Q}\cdot\texttt{ev}_\texttt{n}\cdot\texttt{Q}'$, where n is the line-code of the event. We use φ to range over f and $\texttt{ev}_\texttt{n}$ and tuples $\texttt{Q}\cdot\varphi\cdot\texttt{Q}'$ are called *clauses*.

With an abuse of notation, the contract code is addressed by using the contract name and we write $\texttt{Q f Q}' \in \texttt{C}$ if the code $\texttt{@Q f \{ } W \texttt{ \}} \Rightarrow \texttt{@Q}'$ is in the contract C (also noted $\texttt{@Q f \{ } W \texttt{ \}} \Rightarrow \texttt{@Q}' \in \texttt{C}$). We also write $\texttt{Q}_1\cdot\texttt{ev}_\texttt{n}\cdot\texttt{Q}_2 \in \texttt{Q}\cdot\texttt{f}\cdot\texttt{Q}'$ if there is an event $\texttt{t} \gg \texttt{@Q}_1 \Rightarrow \texttt{@Q}_2$ in the function $\texttt{@Q f \{ } W \texttt{ \}} \Rightarrow \texttt{@Q}'$ that starts at line-code n. In this case we also write $\texttt{Q}_1\cdot\texttt{ev}_\texttt{n}\cdot\texttt{Q}_2 \in \texttt{C}$ and $\texttt{t} \gg_\texttt{n} \texttt{@Q}_1 \Rightarrow \texttt{@Q}_2 \in \texttt{C}$.

3.1 The operational semantics

The meaning of *μStipula* primitives is defined operationally by means of a transition relation. Let $\texttt{C}(\texttt{Q}, \Sigma, \Psi)$ be a tuple where

- C is the contract name;
- Q is the current state of the contract;
- Σ is either _ or a term $W \Rightarrow \texttt{Q}$. Σ represents an empty body (hence, a function/event can be executed or the time may elapse) or a continuation where (the time expressions of) a set of event W must be evaluated;
- Ψ is a (possibly empty) multiset of *pending events* that have been already scheduled for future execution but not yet triggered. In particular, Ψ is either _, when there are no pending events, or it is $W_1 \mid \cdots \mid W_n$ where each $W_i = \mathbb{t}_i \gg_{\texttt{n}_i} \texttt{Q}_i \Rightarrow \texttt{Q}_i'$. The *time guard* $\mathbb{t}_i$ is the *absolute time*: it is the evaluation of the time expression $\texttt{t}_i$ of the event when now is replaced by the value of the global clock (see below and rule [State-Change]). The index $\texttt{n}_i$ is the *line-code* of the event; it is set by the function $\texttt{LC}_{\texttt{Q}\cdot\texttt{f}\cdot\texttt{Q}'}(W)$ (see rule [Function], the definition is omitted). This function also drops the "@" from the states.

Tuples $\texttt{C}(\texttt{Q}, \Sigma, \Psi)$ are ranged over by $\mathbb{C}, \mathbb{C}', \cdots$. A *configuration* is a pair $\langle \mathbb{C}, \mathbb{t} \rangle$, where $\mathbb{t}$ is the time value of the system's global clock. The transition relation of *μStipula* is $\langle \mathbb{C}, \mathbb{t} \rangle \xrightarrow{\mu} \langle \mathbb{C}', \mathbb{t}' \rangle$, where μ is either *empty* _ or f or $\texttt{ev}_\texttt{n}$ (the label $\texttt{ev}_\texttt{n}$ indicates the event at line n; it has been added in this paper for easing the arguments in Section 4). The formal definition of $\langle \mathbb{C}, \mathbb{t} \rangle \xrightarrow{\mu} \langle \mathbb{C}', \mathbb{t}' \rangle$ is given in Table 1 using the predicate $\Psi, \mathbb{t} \nrightarrow$ that is *true* whenever $\Psi = \mathbb{t}_1 \gg_{\texttt{n}_1} \texttt{Q}_1 \Rightarrow \texttt{Q}_1' \mid \cdots \mid \mathbb{t}_h \gg_{\texttt{n}_h} \texttt{Q}_h \Rightarrow \texttt{Q}_h'$ and, for every $1 \le i \le h$, $\mathbb{t}_i \neq \mathbb{t}$; *false* otherwise.

A discussion about the four rules follows. Rule [Function] defines invocations: the label specifies the function name f. The transition may occur provided (*i*) the contract is in the state Q that admits invocations of f and (*ii*) no event can be triggered – *cf.* the premise $\Psi, \mathbb{t} \nrightarrow$ (event's execution preempts function invocation). Rule [State-Change] says that a contract changes state by adding the sequence of events W to the multiset of pending events once their time expressions have been evaluated (now is replaced by the current value of the clock). Rule [Event-Match] specifies that an event handler may run provided Σ is _ and the time guard of the event has exactly the value of the global clock $\mathbb{t}$. Rule [Tick] defines the elapsing of time. This happens when the contract has an empty Σ and no event can be triggered.

The rules [Function] and [State-Change] might have been squeezed in one rule only. We have preferred to keep them apart for compatibility with *Stipula* (where functions' and events' bodies may contain statements, too) and for having simpler rules. The contract's semantics and the following arguments about reachability analysis do not change. Finally, it is worth to observe that *μStipula* has three *causes for nondeterminism*: (*i*) two functions can be invoked in a state, (*ii*) either a function may be invoked or the time may elapse (in this case the function may be invoked at a later time), and (*iii*) two events may be invoked at the same time, then one is chosen and executed.

Remark 1. *Rule* [Tick] *defines the elapsing of time. This happens when the contract has no function or event to conclude, and no event can be triggered. As a consequence, the complete execution of a function or of an event cannot last more than a single time unit. It is worth to notice that this semantics admits the paradoxical phenomenon that an endless sequence of function invocations does not make time elapse. The phenomenon, which is also present in process calculi with time [13, 16], follows by rules that are more nondeterministic. As a*

$$\text{[Function]}\quad \frac{\texttt{@Q f\{ } W \texttt{ \}} \Rightarrow \texttt{@Q}' \in \texttt{C} \quad W' = \mathsf{LC}_{\texttt{Q}\cdot\texttt{f}\cdot\texttt{Q}'}(W) \quad \Psi, \mathbb{t} \not\rightarrow}{\langle \texttt{C(Q, \_, } \Psi\texttt{)}, \mathbb{t}\rangle \xrightarrow{\texttt{f}} \langle \texttt{C(Q, } W' \Rightarrow \texttt{Q}', \Psi\texttt{)}, \mathbb{t}\rangle}$$

$$\text{[State-Change]}\quad \frac{W = \left(\texttt{t}_i \gg_{\texttt{n}_i} \texttt{Q}_i \Rightarrow \texttt{Q}'_i\right)^{i\in 1..h} \quad \left(\mathbb{t}_i = \texttt{t}_i\{^{\mathbb{t}}/_{\texttt{now}}\}\right)^{i\in 1..h} \quad \Psi' = \mathbb{t}_1 \gg_{\texttt{n}_1} \texttt{Q}_1 \Rightarrow \texttt{Q}'_1 \mid \cdots \mid \mathbb{t}_h \gg_{\texttt{n}_h} \texttt{Q}_h \Rightarrow \texttt{Q}'_h}{\langle \texttt{C(Q, } W \Rightarrow \texttt{Q}', \Psi\texttt{)}, \mathbb{t}\rangle \longrightarrow \langle \texttt{C(Q}', \texttt{\_}, \Psi' \mid \Psi\texttt{)}, \mathbb{t}\rangle}$$

$$\text{[Event-Match]}\quad \frac{\Psi = \mathbb{t} \gg_{\texttt{n}} \texttt{Q} \Rightarrow \texttt{Q}' \mid \Psi'}{\langle \texttt{C(Q, \_, } \Psi\texttt{)}, \mathbb{t}\rangle \xrightarrow{\texttt{ev}_{\texttt{n}}} \langle \texttt{C(Q, \_} \Rightarrow \texttt{Q}', \Psi'\texttt{)}, \mathbb{t}\rangle}$$

$$\text{[Tick]}\quad \frac{\Psi, \mathbb{t} \not\rightarrow}{\langle \texttt{C(Q, \_, } \Psi\texttt{)}, \mathbb{t}\rangle \longrightarrow \langle \texttt{C(Q, \_, } \Psi\texttt{)}, \mathbb{t}+1\rangle}$$

Table 1: The operational semantics of *μStipula*

consequence, the sets of reachable configurations with the rules of Table 1 are larger than other sets obtained by more restrictive semantics (henceforth, the following results about unreachability also hold for these last semantics). We finally remark that the actual implementation of μStipula (and of Stipula) prevents the foregoing paradoxical behaviours by using a fair algorithm deciding between executing functions and executing rule [Tick].

The initial configuration of a *μStipula* contract

$$\texttt{stipula C \{ init Q } \quad F \texttt{ \}}$$

is $\langle \texttt{C(Q, \_, \_)}, \mathbb{t}\rangle$, where $\mathbb{t}$ can be any value because it corresponds to the absolute time. We write $\langle \mathbb{C}, \mathbb{t}\rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}'\rangle$, called *computation*, if there are $\mu_1, \cdots, \mu_h$ such that $\langle \mathbb{C}, \mathbb{t}\rangle \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_h} \langle \mathbb{C}', \mathbb{t}'\rangle$ (as said above, μ_i, $i \in 1..h$, may be either $\_$ or a function name or an event).

To illustrate the semantics of *μStipula*, we discuss the computations of SampleTime in Section 2. Let $\mathbb{t}' = \mathbb{t} + k$ and let $\texttt{Ev}_4 = \mathbb{t}' + 1 \gg_4 \texttt{Cont} \Rightarrow \texttt{Run}$ and $\texttt{Ev}_5 = \mathbb{t}' + 2 \gg_5 \texttt{Comp} \Rightarrow \texttt{End}$. The contract may initially perform a number of [Tick] transitions, say k, and then a [Function] one. Therefore we have (on the right we write the rule that is used):

$$\begin{array}{ll}
\langle \texttt{SampleTime(Init, _, _)}, \mathbb{t}\rangle & \\
\quad \longrightarrow^k \langle \texttt{SampleTime(Init, _, _)}, \mathbb{t}'\rangle & \text{[Tick]} \\
\quad \xrightarrow{\texttt{f}} \langle \texttt{SampleTime(Init, Ev}_4\ \texttt{Ev}_5 \Rightarrow \texttt{Cont, _)}, \mathbb{t}'\rangle & \text{[Function]} \\
\quad \longrightarrow \langle \texttt{SampleTime(Cont, _, Ev}_4\texttt{|Ev}_5\texttt{)}, \mathbb{t}'\rangle & \text{[State-Change]} \\
\quad \longrightarrow \langle \texttt{SampleTime(Cont, _, Ev}_4\texttt{|Ev}_5\texttt{)}, \mathbb{t}'+1\rangle & \text{[Tick]} \\
\quad \xrightarrow{\texttt{ev}_4} \langle \texttt{SampleTime(Cont, _} \Rightarrow \texttt{Run, Ev}_5\texttt{)}, \mathbb{t}'+1\rangle & \text{[Event-Match]} \\
\quad \longrightarrow \langle \texttt{SampleTime(Run, _, Ev}_5\texttt{)}, \mathbb{t}'+1\rangle & \text{[State-Change]}
\end{array}$$

At this stage the contract may again perform a number of [Tick] transitions, say k', and then a [Function] executing g. Let $\mathbb{t}'' = \mathbb{t}' + 1 + k'$ and $\texttt{Ev}_9 = \mathbb{t}'' + 2 \gg_9 \texttt{Go} \Rightarrow \texttt{Comp}$, then the computation continues as follows:

$$\begin{array}{ll}
\longrightarrow^{k'} \langle \texttt{SampleTime(Run, _, _)}, \mathbb{t}''\rangle & \text{[Tick]} \\
\quad \xrightarrow{\texttt{g}} \langle \texttt{SampleTime(Run, Ev}_9 \Rightarrow \texttt{Go, _)}, \mathbb{t}''\rangle & \text{[Function]} \\
\quad \longrightarrow \langle \texttt{SampleTime(Go, _, Ev}_5\texttt{|Ev}_9\texttt{)}, \mathbb{t}''\rangle & \text{[State-Change]} \\
\quad \longrightarrow^2 \langle \texttt{SampleTime(Go, _, Ev}_5\texttt{|Ev}_9\texttt{)}, \mathbb{t}''+2\rangle & \text{[Tick]} \\
\quad \xrightarrow{\texttt{ev}_9} \langle \texttt{SampleTime(Go, _} \Rightarrow \texttt{Comp, Ev}_5\texttt{)}, \mathbb{t}''+2\rangle & \text{[Event-Match]} \\
\quad \longrightarrow \langle \texttt{SampleTime(Comp, _, Ev}_5\texttt{)}, \mathbb{t}''+2\rangle & \text{[State-Change]}
\end{array}$$

Notice that, in the last configuration, $\texttt{Ev}_5$ is garbage because the time expression of $\texttt{Ev}_5$ is $\mathbb{t}'+2$ and $\mathbb{t}''+2 = \mathbb{t}'+1+k'+2 > \mathbb{t}'+2$.

Definition 3.1. Let C be a *μStipula* contract with initial configuration $\langle \mathbb{C}, \mathbb{t}\rangle$ and let

$$\langle \mathbb{C}, \mathbb{t}\rangle \longrightarrow^* \xrightarrow{\mu} \langle \texttt{C(Q, } W \Rightarrow \texttt{Q}', \Psi\texttt{)}, \mathbb{t}'\rangle\ .$$

If $\mu = \texttt{f}$ then we say that the $\texttt{Q}\cdot\texttt{f}\cdot\texttt{Q}' \in \texttt{C}$ is *reachable* (from $\langle \mathbb{C}, \mathbb{t}\rangle$); if $\mu = \texttt{ev}_{\texttt{n}}$ then we say that the event $\texttt{Q}\cdot\texttt{ev}_{\texttt{n}}\cdot\texttt{Q}'$ is *reachable* (from $\langle \mathbb{C}, \mathbb{t}\rangle$; in this case $W = \_$).

We remark that, according to Definition 3.1, the reachability predicate uses an *existential quantification* on computations. The notion of *underlying clause of a transition* will be often used in the following sections: the *underlying clause* of

$$\langle \texttt{C(Q, \_, } \Psi\texttt{)}, \mathbb{t}\rangle \xrightarrow{\texttt{f}} \langle \texttt{C(Q, } W \Rightarrow \texttt{Q}', \Psi\texttt{)}, \mathbb{t}\rangle$$

is $\texttt{Q}\cdot\texttt{f}\cdot\texttt{Q}'$; the *underlying clause* of

$$\langle \texttt{C(Q, \_, } \Psi\texttt{)}, \mathbb{t}\rangle \xrightarrow{\texttt{ev}_{\texttt{n}}} \langle \texttt{C(Q, \_} \Rightarrow \texttt{Q}', \Psi\texttt{)}, \mathbb{t}\rangle$$

is $\texttt{Q}\cdot\texttt{ev}_{\texttt{n}}\cdot\texttt{Q}'$.

4 THE THEORY OF THE REACHABILITY ANALYZER – CASE WITHOUT TIME

The reachability analyzer returns a set of clauses that are reachable by computations starting in the initial state of a *μStipula* contract. We split the analysis in two steps: the first one is coarse-grained because it overlooks time expressions, the second one, which is defined in Section 5, refines the first analysis by also considering time.

When we disregard time, determining a set of clauses that are reachable from a state amounts to compute a closure operation on a set A that iteratively extends the set with $\texttt{Q}\cdot\varphi\cdot\texttt{Q}'$ such that Q is in the final state of some clause in A. Notice that the iteration always terminate because the set of clauses of a *μStipula* contract is finite.

However, in the following, instead of reasoning on sets, we argue on *linear traces* of clauses, ranged over by $\mathbb{A}$, $\mathbb{A}'$, ..., that are sequences of clauses without repetitions $\texttt{Q}_1\cdot\varphi_1\cdot\texttt{Q}'_1\ ;\ \ldots\ ;\ \texttt{Q}_n\cdot\varphi_n\cdot\texttt{Q}'_n$ (as a matter of facts, linear traces are linearizations of sets). This will allow us to easily extend the simple saturation technique to cover time expressions, as well. We assume that $\varepsilon\ ;\ \mathbb{A} = \mathbb{A}\ ;\ \varepsilon = \mathbb{A}$. Notice that, due to the linearity constraint, *linear traces are finitely many.*

We will use the following auxiliary operations and notions:

- $\texttt{Q}\cdot\varphi\cdot\texttt{Q}' \in \mathbb{A}$ if there is an element of $\mathbb{A}$ that is equal to $\texttt{Q}\cdot\varphi\cdot\texttt{Q}'$;

- the *extension* $\mathbb{A} \triangleleft \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}'$ of a linear trace $\mathbb{A}$ with the clause $\mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}'$ is

$$\mathbb{A} \triangleleft \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}' \stackrel{\text{def}}{=} \begin{cases} \mathbb{A}\ ;\ \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}' & \text{if } \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}' \notin \mathbb{A} \\ \mathbb{A} & \text{if } \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}' \in \mathbb{A} \end{cases}$$

  Notice that $\mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}'$ is not added to $\mathbb{A}$ when $\mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}' \in \mathbb{A}$ (otherwise the linearity of traces does not hold anymore). We also write $\{\mathbb{A}_1, \cdots, \mathbb{A}_n\} \triangleleft \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}' \stackrel{\text{def}}{=} \{\mathbb{A}_1 \triangleleft \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}', \cdots, \mathbb{A}_n \triangleleft \mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}'\}$.
- let $\mathbb{R}'$ and $\mathbb{R}''$ be two maps from clauses to sets of linear traces. We define $\mathbb{R}' \leq \mathbb{R}''$ if, for every $\mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}'$, $\mathbb{R}'(\mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}') \subseteq \mathbb{R}''(\mathtt{Q}{\cdot}\varphi{\cdot}\mathtt{Q}')$. It turns out that the domain of maps with the partial order $\leq$ is a *lattice* [10] and this lattice, given a *μStipula* contract, is always *finite*.

The closure operation works as follows. For every clause $\mathtt{Q}_1{\cdot}\varphi{\cdot}\mathtt{Q}_2$, we compute the set of linear traces beginning at the initial state $\mathtt{Q}$ and ending with the clause, noted $\mathbb{R}_{\mathtt{Q}}(\mathtt{Q}_1{\cdot}\varphi{\cdot}\mathtt{Q}_2)$. The computation uses a fixpoint technique that stops when sets are saturated.

Definition 4.1. Let $\mathtt{C}$ be a *μStipula* contract with initial state $\mathtt{Q}$. The sequence of maps ${\mathbb{R}_{\mathtt{Q}}}^{(0)}$, ${\mathbb{R}_{\mathtt{Q}}}^{(1)}, \cdots$ that take clauses in $\mathtt{C}$ and return sets of linear traces is defined as follows:

(1) $${\mathbb{R}_{\mathtt{Q}}}^{(0)}(\mathtt{Q}_1{\cdot}\varphi{\cdot}\mathtt{Q}_2) = \begin{cases} \{\ \mathtt{Q}_1{\cdot}\varphi{\cdot}\mathtt{Q}_2\ \} & \text{if } \mathtt{Q} = \mathtt{Q}_1 \text{ and } \varphi = \mathtt{f} \text{ and } \mathtt{Q}{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \in \mathtt{C} \\ \varnothing & \text{otherwise} \end{cases}$$

(2) $${\mathbb{R}_{\mathtt{Q}}}^{(i+1)}(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) = {\mathbb{R}_{\mathtt{Q}}}^{(i)}(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) \cup \Big(\bigcup_{\mathtt{Q}_3{\cdot}\varphi'{\cdot}\mathtt{Q}_1 \in \mathtt{C}} \big\{ \mathbb{A} \triangleleft \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \mid \mathbb{A} \in {\mathbb{R}_{\mathtt{Q}}}^{(i)}(\mathtt{Q}_3{\cdot}\varphi'{\cdot}\mathtt{Q}_1) \big\} \Big)$$

(3) if $\mathtt{Q}_1{\cdot}\mathtt{ev}_n{\cdot}\mathtt{Q}_2 \in \mathtt{Q}'{\cdot}\mathtt{f}{\cdot}\mathtt{Q}''$ then

$${\mathbb{R}_{\mathtt{Q}}}^{(i+1)}(\mathtt{Q}_1{\cdot}\mathtt{ev}_n{\cdot}\mathtt{Q}_2) = {\mathbb{R}_{\mathtt{Q}}}^{(i)}(\mathtt{Q}_1{\cdot}\mathtt{ev}_n{\cdot}\mathtt{Q}_2) \cup \Big(\bigcup_{\mathtt{Q}_3{\cdot}\varphi'{\cdot}\mathtt{Q}_1 \in \mathtt{C}} \big\{ \mathbb{A} \triangleleft \mathtt{Q}_1{\cdot}\mathtt{ev}_n{\cdot}\mathtt{Q}_2 \mid \mathbb{A} \in {\mathbb{R}_{\mathtt{Q}}}^{(i)}(\mathtt{Q}_3{\cdot}\varphi'{\cdot}\mathtt{Q}_1) \text{ and } \mathtt{Q}'{\cdot}\mathtt{f}{\cdot}\mathtt{Q}'' \in \mathbb{A} \big\} \Big)$$

By definition ${\mathbb{R}_{\mathtt{Q}}}^{(i)} \leq {\mathbb{R}_{\mathtt{Q}}}^{(i+1)}$. Since the set of possible functions from clauses to sets of linear traces is a finite lattice, there exists i such that ${\mathbb{R}_{\mathtt{Q}}}^{(i)} = {\mathbb{R}_{\mathtt{Q}}}^{(i+1)}$ [10]. Let $\mathbb{R}_{\mathtt{Q}}$, called the *timeless saturation map*, be such fixpoint.

The map $\mathbb{R}_{\mathtt{Q}}$ takes a clause $\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}''$ and returns the set of linear traces starting at the initial state $\mathtt{Q}$ and ending at $\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}''$. For example, let us compute the timeless saturation map for the contract `PingPong`:

$${\mathbb{R}_{\mathtt{Q0}}}^{(0)}(\mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(0)}(\mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}) = \varnothing$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(0)}(\mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}) = \varnothing$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(0)}(\mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}) = \varnothing$$

$${\mathbb{R}_{\mathtt{Q0}}}^{(1)}(\mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(1)}(\mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(1)}(\mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}) = \varnothing$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(1)}(\mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}) = \varnothing$$

$${\mathbb{R}_{\mathtt{Q0}}}^{(2)}(\mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(2)}(\mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(2)}(\mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(2)}(\mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}) = \varnothing$$

$${\mathbb{R}_{\mathtt{Q0}}}^{(3)}(\mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(3)}(\mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(3)}(\mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(3)}(\mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ ;\ \mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}\ \}$$
$$\cdots$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(6)}(\mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1},\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ ;\ \mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}\ \}$$

$${\mathbb{R}_{\mathtt{Q0}}}^{(6)}(\mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2},\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ ;\ \mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(6)}(\mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3},\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ ;\ \mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}\ \}$$
$${\mathbb{R}_{\mathtt{Q0}}}^{(6)}(\mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}) = \{\ \mathtt{Q0}{\cdot}\mathtt{ping}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Q2}\ ;\ \mathtt{Q2}{\cdot}\mathtt{pong}{\cdot}\mathtt{Q3}\ ;\ \mathtt{Q3}{\cdot}\mathtt{ev}_7{\cdot}\mathtt{Q0}\ \}$$

It turns out that ${\mathbb{R}_{\mathtt{Q0}}}^{(6)} = {\mathbb{R}_{\mathtt{Q0}}}^{(7)}$, therefore ${\mathbb{R}_{\mathtt{Q0}}}^{(6)}$ is the timeless saturation map for `PingPong`.

The timeless saturation map already spots simple unreachabilities. For example, in `Sample`:

$$\mathbb{R}_{\mathtt{Init}}(\mathtt{Init}{\cdot}\mathtt{f}{\cdot}\mathtt{Run}) = \{\ \mathtt{Init}{\cdot}\mathtt{f}{\cdot}\mathtt{Run}\ \}$$
$$\mathbb{R}_{\mathtt{Init}}(\mathtt{Init}{\cdot}\mathtt{g}{\cdot}\mathtt{Go}) = \{\ \mathtt{Init}{\cdot}\mathtt{g}{\cdot}\mathtt{Go}\ \}$$
$$\mathbb{R}_{\mathtt{Init}}(\mathtt{Go}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{End}) = \varnothing$$

meaning that $\mathtt{Go}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{End}$ is unreachable. In particular, the fact that $\mathbb{R}_{\mathtt{Init}}(\mathtt{Go}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{End})$ is empty follows by item 3 of Definition 4.1: the unique transition that may enable $\mathtt{Go}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{End}$ is $\mathtt{Init}{\cdot}\mathtt{g}{\cdot}\mathtt{Go}$; however $\mathtt{Init}{\cdot}\mathtt{g}{\cdot}\mathtt{Go}\ ;\ \mathtt{Go}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{End}$ is an invalid linear trace because $\mathtt{Go}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{End} \notin \mathtt{Init}{\cdot}\mathtt{g}{\cdot}\mathtt{Go}$. More precisely, every set $\mathbb{A} \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}'')$ satisfies a *consistency property*: if an event is in $\mathbb{A}$ then the function that contains it is also in $\mathbb{A}$. For example, in `SampleTime`, the set $\mathbb{R}_{\mathtt{Init}}(\mathtt{Comp}{\cdot}\mathtt{ev}_5{\cdot}\mathtt{End})$ has the linear trace

$$\mathtt{Init}{\cdot}\mathtt{f}{\cdot}\mathtt{Cont}\ ;\ \mathtt{Cont}{\cdot}\mathtt{ev}_4{\cdot}\mathtt{Run}\ ;\ \mathtt{Run}{\cdot}\mathtt{g}{\cdot}\mathtt{Go}\ ;\ \mathtt{Go}{\cdot}\mathtt{ev}_9{\cdot}\mathtt{Comp}\ ;\ \mathtt{Comp}{\cdot}\mathtt{ev}_5{\cdot}\mathtt{End}$$

where $\mathtt{ev}_4$ follows $\mathtt{f}$ and $\mathtt{ev}_9$ and $\mathtt{ev}_5$ follow $\mathtt{g}$. We also notice that there may be $\mathbb{A} \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}'')$ where $\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}''$ is not the last clause of $\mathbb{A}$. This is evident when cyclic computations are present because the linearity constraint drops tailing clauses that are already present in traces. For example, in the contract `Ugly`

$$\mathtt{Q0}{\cdot}\mathtt{f}{\cdot}\mathtt{Q1}\ ;\ \mathtt{Q1}{\cdot}\mathtt{g}{\cdot}\mathtt{Q0} \in \mathbb{R}_{\mathtt{Q0}}(\mathtt{Q0}{\cdot}\mathtt{f}{\cdot}\mathtt{Q1})$$

($\mathtt{Q0}{\cdot}\mathtt{f}{\cdot}\mathtt{Q1}$ is not added twice). We finally notice that, in `SampleTime`, $\mathbb{R}_{\mathtt{Init}}(\mathtt{Comp}{\cdot}\mathtt{ev}_5{\cdot}\mathtt{End})$, which means that $\mathtt{Comp}{\cdot}\mathtt{ev}_5{\cdot}\mathtt{End}$ is reachable according to $\mathbb{R}_{\mathtt{Init}}$ while it is actually unreachable. It turns out

that the timeless saturation map is an over-approximation of the reachability predicate in Definition 3.1.

To state the soundness of $\mathbb{R}_{\mathtt{Q}}$, we need to connect the notion of linear trace and computation. Informally, we take a computation $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$ and we only consider the transitions labelled either $\mathtt{f}$ or $\mathtt{ev}_{\mathtt{n}}$. Then, from left to right, we take the corresponding clause and add it to the trace if it is not already present – operation "$\triangleleft$" (the trace has to be linear). The formal definition is the following one.

Definition 4.2. Let $\mathtt{C}$ be a *μStipula* contract with initial configuration $\langle \mathbb{C}, \mathbb{t} \rangle$. The *underlying linear trace* of $\langle \mathbb{C}, \mathbb{t} \rangle \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} \langle \mathbb{C}', \mathbb{t}' \rangle$ is $\mathbb{A}_n$ such that

$$\begin{array}{ll} \mathbb{A}_0 = \varepsilon & \\ \mathbb{A}_{i+1} = \mathbb{A}_i & \text{if } \mu_{i+1} = \_ \\ \mathbb{A}_{i+1} = \mathbb{A}_i \triangleleft \mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'' & \text{if the underlying clause of } \mu_{i+1} \text{ is } \mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}''. \end{array}$$

For example, the contract `Ugly` in Section 2 has the computation

$$\langle \mathbb{C}, \mathbb{t} \rangle \xrightarrow{\mathtt{f}} \xrightarrow{\mathtt{g}} \xrightarrow{\_} \xrightarrow{\mathtt{f}} \xrightarrow{\mathtt{ev}_8} \langle \mathbb{C}', \mathbb{t}+1 \rangle$$

(the $\_$-labelled transition corresponds to a tick) whose underlying linear trace is

$$\mathtt{Q0}\cdot\mathtt{f}\cdot\mathtt{Q1}\ ;\ \mathtt{Q1}\cdot\mathtt{g}\cdot\mathtt{Q0}\ ;\ \mathtt{Q1}\cdot\mathtt{ev}_8\cdot\mathtt{Q2}\ .$$

Theorem 4.3 (Soundness of $\mathbb{R}_{\mathtt{Q}}$). *Let* $\mathtt{C}$ *be a μStipula contract with initial state* $\mathtt{Q}$ *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle \xrightarrow{\mu} \langle \mathbb{C}'', \mathbb{t}'' \rangle$$

and $\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}''$ *be the underlying clause of* μ. *Then there is* $\mathbb{A} \triangleleft \mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'' \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'')$ *where* $\mathbb{A}$ *is the underlying linear trace of* $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$. *Therefore* $\mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'') \neq \varnothing$.

The completeness of the timeless saturation map $\mathbb{R}_{\mathtt{Q}}$ is false: the foregoing set $\mathbb{R}_{\mathtt{Init}}(\mathtt{Comp}\cdot\mathtt{ev}_5\cdot\mathtt{End})$ in `SampleTime` is a simple counterexample. It is worth to remark that $\mathbb{R}_{\mathtt{Q}}$ defines the reachable clauses and Theorem 4.3 guarantees that the set of reachable clauses according to $\mathbb{R}_{\mathtt{Q}}$ is a superset of the actual ones. This means that the set of *unreachable clauses* according to $\mathbb{R}_{\mathtt{Q}}$, that is, all those clauses such that $\mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'') = \varnothing$, is a *subset* of the actual ones.

The definition of $\mathbb{R}_{\mathtt{Q}}$ is used to formalize cyclic behaviours of contract functions, a relevant predicate in the following because we will spot timed-out events only when the corresponding functions are acyclic. Of course, we cannot verify the cyclicity of a function $\mathtt{f}$ by looking for a $\mathbb{A} \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'')$ where the clause $\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''$ occurs twice because of the linearity constraint. Rather, we consider the existence of a cycle $\mathtt{Q}_1\cdot\varphi\cdot\mathtt{Q}_1'\ ;\ \mathbb{A}'\ ;\ \mathtt{Q}_2\cdot\varphi'\cdot\mathtt{Q}_1$ such that $\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'' \in \mathbb{A}'$. This is what is required in the following definition.

Definition 4.4. Let $\mathtt{C}$ be a *μStipula* contract with initial state $\mathtt{Q}$. Let

$$\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') = \begin{cases} \mathit{true} & \text{if there is } \mathbb{A}\ ;\ \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''\ ;\ \mathbb{A}' \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') \\ & \text{such that } \mathtt{Q}_1\cdot\varphi\cdot\mathtt{Q}_2 \in \mathbb{A}\ ;\ \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'' \\ & \text{and } \mathtt{Q}_3\cdot\varphi'\cdot\mathtt{Q}_1 \in \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''\ ;\ \mathbb{A}' \\ \mathit{false} & \text{otherwise} \end{cases}$$

($\circlearrowleft_{\mathtt{Q}}$ is undefined on events.)

For example, in the `Ugly` contract,

$$\mathtt{Q0}\cdot\mathtt{f}\cdot\mathtt{Q1}\ ;\ \mathtt{Q1}\cdot\mathtt{g}\cdot\mathtt{Q0} \in \mathbb{R}_{\mathtt{Q0}}(\mathtt{Q0}\cdot\mathtt{f}\cdot\mathtt{Q1})$$

and we deem $\circlearrowleft_{\mathtt{Q0}}(\mathtt{Q0}\cdot\mathtt{f}\cdot\mathtt{Q1}) = \mathit{true}$ by taking $\mathtt{Q}_1\cdot\varphi\cdot\mathtt{Q}_2 = \mathtt{Q0}\cdot\mathtt{f}\cdot\mathtt{Q1}$ and $\mathtt{Q}_3\cdot\varphi'\cdot\mathtt{Q}_1 = \mathtt{Q1}\cdot\mathtt{g}\cdot\mathtt{Q0}$. Additionally, we also have

$$\mathtt{Q0}\cdot\mathtt{f}\cdot\mathtt{Q1}\ ;\ \mathtt{Q1}\cdot\mathtt{g}\cdot\mathtt{Q0} \in \mathbb{R}_{\mathtt{Q0}}(\mathtt{Q1}\cdot\mathtt{g}\cdot\mathtt{Q0})$$

and we deem $\circlearrowleft_{\mathtt{Q0}}(\mathtt{Q1}\cdot\mathtt{g}\cdot\mathtt{Q0}) = \mathit{true}$ by taking the same instances of $\mathtt{Q}_1\cdot\varphi\cdot\mathtt{Q}_2$ and $\mathtt{Q}_3\cdot\varphi'\cdot\mathtt{Q}_1$.

The predicate $\circlearrowleft_{\mathtt{Q}}$ is sound; the proof uses the Theorem 4.3.

Proposition 4.5. *Let* $\mathtt{C}$ *be a μStipula contract with initial state* $\mathtt{Q}$ *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \xrightarrow{\mu} \longrightarrow^* \xrightarrow{\mu} \langle \mathbb{C}', \mathbb{t}' \rangle$$

such that the underlying clause of the transitions labelled μ *is* $\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''$. *Then* $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') = \mathit{true}$.

The converse of Proposition 4.5 is false: since $\circlearrowleft_{\mathtt{Q}}$ is over-approximating, it is possible that $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') = \mathit{true}$ and there is no computation manifesting the corresponding cyclic behaviour.

5 REMOVING TIMED-OUT EVENTS

We now refine $\mathbb{R}_{\mathtt{Q}}$ in order to address time expressions and drop those events (and their continuations) whose time expressions are inconsistent with the contract behaviour. We introduce the technique by discussing the contract `SampleTime` in Section 2.

In `SampleTime`, the set $\mathbb{R}_{\mathtt{Init}}(\mathtt{Comp}\cdot\mathtt{ev}_5\cdot\mathtt{End})$ has a linear trace that does not correspond to any computation because the time when $\mathtt{ev}_5$ should be executed is already elapsed (see also the foregoing discussion). To remove these kind of traces, we need to record the times when clauses are executed. However, this is not possible at static time because the absolute time $\mathbb{t}$ is a runtime concept. Therefore we use symbolic time expressions, called *logical times* below, as surrogates of absolute times. These logical times are associated to clauses of linear traces and the consistency of a trace is assessed by verifying the compatibility of the corresponding logical times. In this context, the critical point is the presence of cyclic behaviours (*cf.* the discussion about the contract `Ugly` in Section 2) because linear traces do not faithfully represent computations (a single clause may correspond to several instances of a function that are executed at different times). In fact, in these cases, linear traces and logical times may lead to wrong conclusions about reachability (as discussed for `Ugly`). For this reason we have decided to restrict the refinement of $\mathbb{R}_{\mathtt{Q}}$ to acyclic functions only.

We begin the formal development with the definition of logical time and with three auxiliary functions, one `EV` returns the set of events of a function, the other `TE` returns the time expressions (without `now`) of a set of events, the last returns the clauses within a set that are in a linear trace:

- the *logical time* is a term of the form $\zeta_1 + \cdots + \zeta_h + k$, where ζ_i are variables representing the times when functions are invoked and k is a constant in Nat;

- let C be a *μStipula* contract and $\mathtt{Q{\cdot}f{\cdot}Q'} \in \mathtt{C}$. In order to have a lighter notation, we keep C implicit:

$$\mathsf{EV}(\mathtt{Q{\cdot}f{\cdot}Q'}) \stackrel{\text{def}}{=} \{\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}\ \mid\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2} \in \mathtt{Q{\cdot}f{\cdot}Q'}\ \}$$

$$\mathsf{TE}(\mathtt{A}) \stackrel{\text{def}}{=} \{\ \kappa_i\ \mid\ \mathtt{A} = \cup_{i\in 1..m}\{\mathtt{Q}_i\ \mathtt{ev_n}_i\ \mathtt{Q}'_i\} \text{ and } (\mathtt{now} + \kappa_i \gg_{\mathtt{n}_i} \mathtt{@Q}_i \Rightarrow \mathtt{@Q}'_i \in \mathtt{C})^{i\in 1..m}\ \}$$

- the *restriction* of $\mathbb{A}$ to a set A of clauses is

$$\mathbb{A}|_{\mathtt{A}} \stackrel{\text{def}}{=} \begin{cases} \varnothing & \text{if } \mathbb{A} = \varepsilon \\ \mathbb{A}'|_{\mathtt{A}} & \text{if } \mathbb{A} = \mathbb{A}'\ ;\ \mathtt{Q{\cdot}\varphi{\cdot}Q'} \text{ and } \mathtt{Q{\cdot}\varphi{\cdot}Q'} \notin \mathtt{A} \\ \mathbb{A}'|_{\mathtt{A}} \cup \{\mathtt{Q{\cdot}\varphi{\cdot}Q'}\} & \text{if } \mathbb{A} = \mathbb{A}'\ ;\ \mathtt{Q{\cdot}\varphi{\cdot}Q'} \text{ and } \mathtt{Q{\cdot}\varphi{\cdot}Q'} \in \mathtt{A} \end{cases}$$

Next we define two relevant functions for the following (timed) saturation map. The first one, noted $\mathfrak{U}_\nu$, returns the logical time of a trace. It does this by adding the logical time of clauses. Actually the addition is clever: the clauses that are in between a function $\mathtt{Q{\cdot}f{\cdot}Q'}$ and a corresponding event, say $\mathbb{A}''\ ;\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$, are skipped and replaced by the highest κ of the events of $\mathtt{Q{\cdot}f{\cdot}Q'}$ in $\mathbb{A}''\ ;\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$. We will demonstrate that, when $\mathtt{Q{\cdot}f{\cdot}Q'}$ is acyclic, this approximation is sound. The second function, noted $\mathfrak{I}_\nu$, returns the logical times of a set of linear traces.

Definition 5.1. Let C be a *μStipula* contract and ν be an injective mapping from clauses $\mathtt{Q{\cdot}f{\cdot}Q'}$ to variables (to simplify the notation we always let $\nu(\mathtt{Q{\cdot}f{\cdot}Q'}) = \zeta_{\mathtt{f}}$). Let

- $\mathfrak{U}_\nu$ be the function from linear traces to logical times defined as follows:

$$\mathfrak{U}_\nu(\mathbb{A}) \stackrel{\text{def}}{=} \begin{cases} \mathfrak{U}_\nu(\mathbb{A}') + \nu(\mathtt{Q{\cdot}f{\cdot}Q'}) & \text{if } \mathbb{A} = \mathbb{A}'\ ;\ \mathtt{Q{\cdot}f{\cdot}Q'} \\ \mathfrak{U}_\nu(\mathbb{A}') + \nu(\mathtt{Q{\cdot}f{\cdot}Q'}) + \kappa & \text{if } \mathbb{A} = \mathbb{A}'\ ;\ \mathtt{Q{\cdot}f{\cdot}Q'}\ ;\ \mathbb{A}''\ ;\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2} \\ & \quad \text{and } \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2} \in \mathtt{Q{\cdot}f{\cdot}Q'} \\ & \quad \text{and } \kappa = \max\left(\mathsf{TE}\left(\mathbb{A}''\ ;\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}|_{\mathsf{EV}(\mathtt{Q{\cdot}f{\cdot}Q'})}\right)\right) \end{cases}$$

- $\mathfrak{I}_\nu$, called *time function*, be the following function from sets of linear traces to sets of logical times:

$$\mathfrak{I}_\nu\left(\left\{\mathbb{A}_1, \cdots, \mathbb{A}_n\right\}\right) \stackrel{\text{def}}{=} \left\{\ \mathfrak{U}_\nu(\mathbb{A}_1), \cdots, \mathfrak{U}_\nu(\mathbb{A}_n)\ \right\}$$

For example, in SampleTime, assuming $\nu(\mathtt{Init{\cdot}f{\cdot}Cont}) = \zeta_{\mathtt{f}}$ and $\nu(\mathtt{Run{\cdot}g{\cdot}Go}) = \zeta_{\mathtt{g}}$, we have

$$\mathfrak{U}_\nu(\mathtt{Init{\cdot}f{\cdot}Cont}) = \zeta_{\mathtt{f}}$$
$$\mathfrak{U}_\nu(\mathtt{Init{\cdot}f{\cdot}Cont\ ;\ Cont{\cdot}ev_4{\cdot}Run}) = \zeta_{\mathtt{f}} + 1$$
$$\mathfrak{U}_\nu(\mathtt{Init{\cdot}f{\cdot}Cont\ ;\ Cont{\cdot}ev_4{\cdot}Run\ ;\ Run{\cdot}g{\cdot}Go}) = \zeta_{\mathtt{f}} + \zeta_{\mathtt{g}} + 1$$
$$\mathfrak{U}_\nu(\mathtt{Init{\cdot}f{\cdot}Cont\ ;\ Cont{\cdot}ev_4{\cdot}Run\ ;\ Run{\cdot}g{\cdot}Go\ ;\ Go{\cdot}ev_9{\cdot}Comp}) = \zeta_{\mathtt{f}} + \zeta_{\mathtt{g}} + 3$$

It is worth to comment the definition of $\mathfrak{U}_\nu(\mathbb{A})$ when $\mathbb{A} = \mathbb{A}'\ ;\ \mathtt{Q{\cdot}f{\cdot}Q'}\ ;\ \mathbb{A}''\ ;\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$ with $\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2} \in \mathtt{Q{\cdot}f{\cdot}Q'}$. This trace might be the underlying linear trace of a computation

$$\langle \mathbb{C}, \mathbb{t}\rangle \longrightarrow^* \xrightarrow{\mathtt{f}} \langle \mathbb{C}', \mathbb{t}'\rangle \longrightarrow^* \xrightarrow{\mathtt{ev_n}} \langle \mathbb{C}'', \mathbb{t}''\rangle$$

(assume that the transition $\xrightarrow{\mathtt{f}}$ is the leftmost labelled f) where $\langle \mathbb{C}', \mathbb{t}'\rangle \longrightarrow^* \xrightarrow{\mathtt{ev_n}} \langle \mathbb{C}'', \mathbb{t}''\rangle$ may contain several transitions labelled f. This means that the event of $\xrightarrow{\mathtt{ev_n}}$ might have been created by a later invocation of f and $\langle \mathbb{C}', \mathbb{t}'\rangle \longrightarrow^* \xrightarrow{\mathtt{ev_n}} \langle \mathbb{C}'', \mathbb{t}''\rangle$ might also contain another transition labelled $\mathtt{ev}_{\mathtt{n}'}$, where $\mathtt{ev}_{\mathtt{n}'}$ is also an event of f. Therefore the clause $\mathtt{ev}_{\mathtt{n}'}$ occurs in $\mathbb{A}''$ (it is ahead of $\mathtt{ev_n}$), even if the time expressions of $\mathtt{ev_n}$ and $\mathtt{ev}_{\mathtt{n}'}$, say $\mathtt{now} + k$ and $\mathtt{now} + k'$, respectively, have $k < k'$ (of course, this cannot happen if f is acyclic). For this reason, in the definition of $\mathfrak{U}_\nu(\mathbb{A})$, we take the maximum of the time expressions in $\mathbb{A}''\ ;\ \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$ of events in $\mathtt{Q{\cdot}f{\cdot}Q'}$.

Definition 5.2. Let T and T' be two sets of logical times. We say that $T \le T'$ *is solvable* if there are $t \in T$, $t' \in T'$ and a ground substitution σ, such that $t\sigma \le t'\sigma$ (σ maps variables to naturals). Otherwise we say that $T \le T'$ *is unsolvable*.

For example $\{x + y + 1\} \le \{x + 2\}$ is solvable (replacing y with either 0 or 1), while $\{x + y + 1\} \le \{x + y\}$ is not solvable.

Everything is in place for the definition of the (timed) saturation map $\mathbb{T}_{\mathtt{Q}}$ that refines $\mathbb{R}_{\mathtt{Q}}$; the definition and its soundness conclude the section.

Definition 5.3. Let C be a *μStipula* contract with initial state Q and let ν be an injective mapping from clauses $\mathtt{Q'{\cdot}f{\cdot}Q''}$ to variables. The sequence of maps $\mathbb{T}_{\mathtt{Q}}{}^{(0)}$, $\mathbb{T}_{\mathtt{Q}}{}^{(1)}, \cdots$ is defined as follows:

(1) $$\mathbb{T}_{\mathtt{Q}}{}^{(0)}(\mathtt{Q_1{\cdot}\varphi{\cdot}Q_2}) = \begin{cases} \{\ \mathtt{Q_1{\cdot}\varphi{\cdot}Q_2}\ \} & \text{if } \mathtt{Q} = \mathtt{Q_1} \text{ and } \varphi = \mathtt{f} \\ & \quad \text{and } \mathtt{Q{\cdot}f{\cdot}Q_2} \in \mathtt{C} \\ \varnothing & \text{otherwise} \end{cases}$$

(2) $$\mathbb{T}_{\mathtt{Q}}{}^{(i+1)}(\mathtt{Q_1{\cdot}f{\cdot}Q_2}) = \mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q_1{\cdot}f{\cdot}Q_2}) \cup\left(\cup_{\mathtt{Q_3{\cdot}\varphi'{\cdot}Q_1}\in\mathtt{C}}\left\{\mathbb{A} \triangleleft \mathtt{Q_1{\cdot}f{\cdot}Q_2}\ \mid\ \mathbb{A} \in \mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q_3{\cdot}\varphi'{\cdot}Q_1})\right\}\right)$$

(3) if $\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2} \in \mathtt{Q'{\cdot}f{\cdot}Q''}$ and $\circlearrowleft_{\mathtt{C}}(\mathtt{Q'{\cdot}f{\cdot}Q''}) = \mathit{true}$, then

$$\mathbb{T}_{\mathtt{Q}}{}^{(i+1)}(\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}) = \mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}) \cup\left(\cup_{\mathtt{Q_3{\cdot}\varphi'{\cdot}Q_1}\in\mathtt{C}}\left\{\mathbb{A} \triangleleft \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}\ \mid\ \mathbb{A} \in \mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q_3{\cdot}\varphi'{\cdot}Q_1}) \text{ and } \mathtt{Q'{\cdot}f{\cdot}Q''} \in \mathbb{A}\right\}\right)$$

(4) if $\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2} \in \mathtt{Q'{\cdot}f{\cdot}Q''}$ and $\circlearrowleft_{\mathtt{C}}(\mathtt{Q'{\cdot}f{\cdot}Q''}) = \mathit{false}$, then

$$\mathbb{T}_{\mathtt{Q}}{}^{(i+1)}(\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}) = \mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}) \cup\left(\cup_{\mathtt{Q_3{\cdot}\varphi'{\cdot}Q_1}\in\mathtt{C}}\left\{\mathbb{A} \triangleleft \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}\ \mid\ \mathbb{A} \in \mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q_3{\cdot}\varphi'{\cdot}Q_1}) \text{ and } \mathtt{Q'{\cdot}f{\cdot}Q''} \in \mathbb{A} \text{ and } \mathfrak{I}_\nu\left(\{\mathbb{A}\}\right) \le \mathfrak{I}_\nu\left(\mathbb{T}_{\mathtt{Q}}{}^{(i)}(\mathtt{Q'{\cdot}f{\cdot}Q''}) \triangleleft \mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}\right) \text{ is solvable}\right\}\right)$$

By definition $\mathbb{T}_{\mathtt{Q}}{}^{(i)} \le \mathbb{T}_{\mathtt{Q}}{}^{(i+1)}$. Since the maps from clauses to sets of linear traces of a contract C is a finite lattice, there exists i such that $\mathbb{T}_{\mathtt{Q}}{}^{(i)} = \mathbb{T}_{\mathtt{Q}}{}^{(i+1)}$ [10]. Let $\mathbb{T}_{\mathtt{Q}}$, called *(timed) saturation map*, be such fixpoint.

The function $\mathbb{T}_{\mathtt{Q}}$ is clearly a refinement of $\mathbb{R}_{\mathtt{Q}}$, adding more constraints in case of events $\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$ that do not belong to cyclic functions. In these cases, it is verified that there is a linear trace $\mathbb{A}$ leading to $\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$ (without the event) such that its logical time is smaller or equal to that of some linear traces of $\mathtt{Q'{\cdot}f{\cdot}Q''}$ (the function containing the event) plus the value κ ($\mathtt{now} + \kappa$ is the time expression of $\mathtt{Q_1{\cdot}ev_n{\cdot}Q_2}$). In case the constraint is not solvable, the linear trace is not added. For example, in SampleTime, where every function is not cyclic, $\mathbb{T}_{\mathtt{Init}}(\mathtt{Comp{\cdot}ev_5{\cdot}End}) = \varnothing$ because

- $\mathbb{T}_{\mathtt{Init}}(\mathtt{Go{\cdot}ev_9{\cdot}Comp}) = \{\ \mathtt{Init{\cdot}f{\cdot}Cont\ ;\ Cont{\cdot}ev_4{\cdot}Run\ ;\ Run{\cdot}g{\cdot}Go\ ;\ Go{\cdot}ev_9{\cdot}Comp}\ \}$;
- then $\mathfrak{I}_\nu\left(\mathbb{T}_{\mathtt{Init}}(\mathtt{Go{\cdot}ev_9{\cdot}Comp})\right) = \zeta_{\mathtt{f}} + \zeta_{\mathtt{g}} + 3$ and $\mathfrak{I}_\nu\left(\mathbb{T}_{\mathtt{Init}}(\mathtt{Init{\cdot}f{\cdot}Cont}) \triangleleft \mathtt{Comp{\cdot}ev_5{\cdot}End}\right) = \zeta_{\mathtt{f}} + 2$;
- and there is no ground substitution such that $\zeta_{\mathtt{f}} + \zeta_{\mathtt{g}} + 3 \le \zeta_{\mathtt{f}} + 2$.

Theorem 5.4 (Soundness of $\mathbb{T}_{\mathtt{Q}}$). *Let* C *be a μStipula contract with initial state* Q *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle \xrightarrow{\mu} \langle \mathbb{C}'', \mathbb{t}' \rangle$$

and $\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}''$ *be the underlying clause of* μ. *Then there is* $\mathbb{A} \triangleleft \mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'' \in \mathbb{T}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'')$ *where* $\mathbb{A}$ *is the underlying linear trace of* $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$. *Therefore* $\mathbb{T}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'') \neq \varnothing$.

Remark. The current reachability analyzer is a trade-off between coverage of cases and complexity of the theory. We have already observed that the precision of the analyzer is reduced in presence of cyclic functions. However, there are also two other sources of imprecision that we have noticed. *μStipula* semantics gives precedence to events when, in a configuration, both functions can be invoked and events can be performed. This means that in the contract `UglyNow`

```
1  stipula UglyNow {
2     init Q0
3     @Q0 f {
4        now ≫ @Q1 ⇒ @Q2
5     } ⇒ @Q1
6     @Q1 g {
7     } ⇒ @Q3
8  }
```

g is unreachable and the analyzer does not spot it. The imprecision of `UglyNow` may be easily recognized because it only appears in contracts that have a time expression `now`. The fix of the theory has not been implemented in order to ease the understanding; however it can be simply solved by replacing the above function `Q0·f·Q1` with a function `Q0·f·Q2` and analyzing the reachability in the new contract.

In another case, fixing the imprecision is more difficult. Consider the contract `TwoEvents`:

```
1  stipula TwoEvents {
2     init Q0
3     @Q0 f {
4        now + 1 ≫ @Q1 ⇒ @Q3
5        now + 2 ≫ @Q1 ⇒ @Q2
6     } ⇒ @Q1
7     @Q2 g { } ⇒ @Q3
8  }
```

Here, $\mathtt{ev}_4$ and $\mathtt{ev}_5$ start in the same state at two different times. It turns out that $\mathtt{ev}_5$ starting at a later time (and its continuation g) will be unreachable and not detected by the analyzer. While this case is easy to spot, the general one where the competing events might belong to different functions or even to different instances of the same function is difficult. A more precise analysis should require to record the events that do not occur in the abstract computation and that compete with each other. The definition of $\mathbb{T}_{\mathtt{Q}}$ should also use this additional set to reduce the imprecision. All these precision issues are relevant research problems that are already on our agenda. However, it is worth to remark that the pattern in `UglyNow` never showed up in the (more than 30 real) legal contracts we have encoded in *Stipula*, while we found only once the pattern in `TwoEvents` (the `Bakery` contract in the repository [11]).

6 EXTENSIONS

The time expressions of *μStipula* are very basic. Here, taking inspiration from *Stipula* [9], we address more expressive time expressions and the corresponding extensions of the theory in Section 5. This extension of *μStipula* has been integrated in the reachability analyzer [11]

Time expressions with field names. Consider the extension of *μStipula* where contracts may also be parametric with respect to names, *e.g.*

```
stipula C(x1, . . . , xn) { init Q  F }
```

and where time expressions also contain field names, *i.e.*

```
t ::= now  |  t + κ  |  t + x
```

The above contract is invoked by replacing parameters with values (natural numbers), say $v_1, \ldots, v_n$ and its semantics is the same of the contract where F is replaced by $F\{^{v_1,\ldots,v_n}/_{\mathtt{x}_1,\ldots,\mathtt{x}_n}\}$. [In *Stipula*, the inizialization of these names happens in the agreement clause involving all the parties in the external environment [9].] This means that the values of parameters $\mathtt{x}_1, \ldots, \mathtt{x}_n$ are undefined when the contract is drawn up. A an example example, take the contract

```
1  stipula Agree(x,y) {
2     init Init
3     @Init f {
4        now + x ≫ @Run ⇒ @Comp
5     } ⇒ @Cont
6     @Cont g {
7        now + y ≫ @Comp ⇒ @End
8     } ⇒ @Run
9  }
```

To cover these situations (which are very common in *Stipula*), the analyzer also deals with symbolic names and with constraints over them. This is not simple and we discuss our solution below.

Without loss of generality, we assume that no field occurring in time expressions has been initialized. We use an alternative form of the function $\mathfrak{V}_v$ in Section 4. The new function, noted $\mathfrak{V}_v^{wn}$ (the superscript *wn* stands for "with names"), returns *sets* T of terms $\zeta_{\mathtt{f}_1} + \cdots + \zeta_{\mathtt{f}_m} + \kappa_1 \times \mathtt{x}_1 + \cdots + \kappa_h \times \mathtt{x}_h + \kappa$, where $\zeta_{\mathtt{f}_1} + \cdots + \zeta_{\mathtt{f}_m}$ is a logical time, $\mathtt{x}_1, \cdots, \mathtt{x}_h$ are fields of the contract and $\kappa_1, \cdots, \kappa_h, \kappa \in \mathsf{Nat}$. We also redefine TE and $\mathfrak{I}_v$ (the function EV is the same). Let C be an (extended) *μStipula* contract; then (the reference to C is omitted in order to have a lighter notation)

- $\mathsf{TE}^{wn}(\bigcup_{i\in 1..m}(\{\mathtt{Q}_i\cdot\mathtt{ev}_{\mathtt{n}_i}\cdot\mathtt{Q}'_i\}) \stackrel{\text{def}}{=} \{t_i \mid i \in 1..m \text{ and } \mathsf{now} + t_i \gg_{\mathtt{n}_i} @\mathtt{Q}_i \Rightarrow @\mathtt{Q}'_i \in \mathtt{C}\}$;
- $\mathfrak{V}_v^{wn}$ be the following function from linear traces to logical times:

$$\mathfrak{V}_v^{wn}(\mathbb{A}) \stackrel{\text{def}}{=} \begin{cases} \mathfrak{V}_v^{wn}(\mathbb{A}') + v(\mathtt{Q}\cdot\mathtt{f}\cdot\mathtt{Q}') & \text{if } \mathbb{A} = \mathbb{A}' \,;\, \mathtt{Q}\cdot\mathtt{f}\cdot\mathtt{Q}' \\ \bigcup_{t\in T}\Big(\mathfrak{V}_v^{wn}(\mathbb{A}') + v(\mathtt{Q}\cdot\mathtt{f}\cdot\mathtt{Q}') + t\Big) & \text{if } \mathbb{A} = \mathbb{A}' \,;\, \mathtt{Q}\cdot\mathtt{f}\cdot\mathtt{Q}' \,;\, \mathbb{A}'' \,;\, \mathtt{Q}_1\cdot\mathtt{ev}_{\mathtt{n}}\cdot\mathtt{Q}_2 \\ & \text{and } \mathtt{Q}_1\cdot\mathtt{ev}_{\mathtt{n}}\cdot\mathtt{Q}_2 \in \mathtt{Q}\cdot\mathtt{f}\cdot\mathtt{Q}' \\ & \text{and } T = \mathsf{TE}^{wn}(\mathbb{A}'' \,;\, \mathtt{Q}_1\cdot\mathtt{ev}_{\mathtt{n}}\cdot\mathtt{Q}_2|_{\mathsf{EV}(\mathtt{Q}\cdot\mathtt{f}\cdot\mathtt{Q}')}) \end{cases}$$

- $\mathfrak{I}_v^{wn}$ be the following function from sets of linear traces to sets of logical times:

$$\mathfrak{I}_\nu^{wn}(\{\mathbb{A}_1, \cdots, \mathbb{A}_n\}) \stackrel{\text{def}}{=} \bigcup_{i\in 1..n} \mathfrak{V}_\nu^{wn}(\mathbb{A}_i)$$

Using the foregoing functions, the analyzer computes $\mathbb{T}_{\mathtt{Q}}$, by gathering constraints between fields. *Mutatis mutandis*, the definition is similar to the one of Definition 5.3. For example, if $\mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Q}_0}(\mathtt{Q}\cdot\varphi\cdot\mathtt{Q}')) = \{t_1, t_2\}$ and $\mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Q}_0}(\mathtt{Q}'\cdot\varphi'\cdot\mathtt{Q}'')) = \{t_1', t_2'\}$ then the analyzer highlights the four constraints $t_1 \le t_1'$, $t_1 \le t_2'$, $t_2 \le t_1'$, $t_2 \le t_2'$ in the returned message.

To view the above functions at work, consider the contract `Agree`. Then

$$\mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Init}}(\mathtt{Run}\cdot\mathtt{ev}_5\cdot\mathtt{Comp})) = \{\zeta_{\mathtt{f}} + \zeta_{\mathtt{g}} + 1 \times \mathtt{x} + 0 \times \mathtt{y} + 0\}$$

and

$$\mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Init}}(\mathtt{Comp}\cdot\mathtt{ev}_8\cdot\mathtt{End})) = \{\zeta_{\mathtt{f}} + \zeta_{\mathtt{g}} + 0 \times \mathtt{x} + 1 \times \mathtt{y} + 0\}\ .$$

Therefore

$$\mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Init}}(\mathtt{Run}\cdot\mathtt{ev}_5\cdot\mathtt{Comp})) \le \mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Init}}(\mathtt{Comp}\cdot\mathtt{ev}_8\cdot\mathtt{End}))$$

is solvable provided $\mathtt{x} \le \mathtt{y}$. In fact, the reachability analyzer [11] returns

```
"reachability constraint": [ x <= y ] .
```

Currently, our analyzer uses a very basic constraint solver that performs partial evaluations taking advantage of the additive format of time expressions. For example, it does not compute transitive closures of constraints. Integrating the analyzer with an off-the-shelf constraint solver is on the to-do list of the prototype development.

A full-fledged set of time expressions. The constant value κ in *μStipula* time expressions represents minutes. Actually, it is possible to have other time units and also absolute times. In particular, consider the following extension of *μStipula* time expressions

```
t ::=  now  |  date  |  t + x
   |  t + κ('Y' | 'M' | 'D' | 'h' | 'm')?
```

where *date* has the format "*year−month−day*". When the time expression is `t + 5M`, it means "add `5` *months* to `t`" (`Y`, `D`, `h` and `m` stand for years, days, hours and minutes, respectively); when we write `t + 5`, as in the whole paper, it is intended `t + 5m`.

The reachability analyzer covers these generic time-expressions by means of a preprocessor that transforms them into those in the restricted syntax of Section 3.

We discuss the case of an event *date* ≫ `@Q` ⇒ `@Q′` in a function $\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2$, the other cases are similar. In this case, the preprocessor (*i*) replaces every *date* with an expression `now+_x`, where `_x` is added to the fields, and (*ii*) returns a warning message if the computer clock is greater than *date*. Then the analyzer computes $\mathfrak{I}_\nu^{wn}$ considering all these new fields `_x`. The set of returned constraints are then extended with those of the form t + `_x` = *date*, for every $t \in \mathfrak{I}_\nu^{wn}(\mathbb{T}_{\mathtt{Q}_0}(\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2))$. For instance, if the code

```
1  stipula OutofTime {
2      init Init
3      @Init f {
4          "2024-01-01" ≫ @Cont { } ⇒ @End
5      } ⇒ @Cont
6  }
```

is executed today, the analyzer (actually the preprocessor) returns the warning `"expired code": [ Cont ev_5 End ] .`

7 RELATED WORKS

Reachability analysis is a very common optimization in compiler construction that uses data-flow analysis over the control flow graph of programs [1, 5]. In turn, this data-flow analysis employs fixpoint techniques to pinpoint parts of codes that cannot be reached. In our case, the control flow graph is the state-transition system of a *μStipula* contract and the fixpoint techniques (in the data-flow analysis) use dependencies event-function and perform the symbolic executions to derive logical times of clauses that might spot unreachable code units (functions or events). In particular, in *μStipula*, the symbolic executions return a set of constraints on (field) names that are evaluated by the analyzer and used to signal wrong instances of names. The technique we have followed is similar to the one described in [4], with the main difference that they use a stronger constraint solver. As we said in Section 6, the integration on the reachability analyzer with an off-the-shelf constraint solver is a future step.

Another technique that is used for reachability analysis is model checking. It undertakes a systematic exploration of the computations to verify a temporal logic formula expressing a reachability property (SPIN [14] and NuSVM [7] are two well-known tools, the latter uses a symbolic technique to reduce the state explosion problem). While model checking is good at catching difficult corner cases, it is problematic because the state space of *μStipula* contracts may be infinite. In particular, time expressions may generate very subtle reachability dependencies (*cf.* the `Ugly` contract) that require several instances of a function. At the time of writing, it is unclear whether there is a relationship between the number of instances of functions and the time expressions of a contract that might bound the reachability analysis.

The standard reference model of systems with clocks is *timed automata* [3]. As regards reachability analysis, one can think to define a compilation pattern from *μStipula* contracts to timed automata with one clock and without the reset operation. Henceforth, the reachability analysis of a *μStipula* is reduced to the reachability problem of this subclass of timed automata that has been proved to be *NLogSpace*-complete [15]. At the time of writing, we have not succeeded in defining any compilation in timed automata (and variants of this model). The main problem we found was the management of events that may increase in number while the computation progress (and no tick is performed) and may disappear when a tick is performed. We will continue to study this issue.

Another relevant issue that is related to reachability is the expressive power of *μStipula*. In particular, a variant of reachability, *coverability*, has also been studied for a large class of transition systems – the well-structured ones [12] (coverability is decidable in the subclass of well-structured transition system with an effective pred-basis and decidable well-quasi-ordering relation). However, our attempts to prove that the *μStipula* model matches with the well-structured constraints have failed. At the same time we have not been able to encode Turing-powerful models, such as 2-counters machines. It turns out that the expressive power of *μStipula* is an open problem: apparently, the combination of automata, time, and function invocations is a novel matter. Time strongly contributes to the complexity: the removal of time makes the expressiveness of

the sublanguage equivalent to that of finite state automata (where reachability is computed by the function $\mathbb{R}_{\mathtt{Q}}$).

8 CONCLUSIONS

We have studied the reachability of clauses in contracts written in *μStipula*, a formal language with a precise syntax and semantics. This problem is knotty because clauses may be triggered by time constraints that must be partially evaluated. We have defined an algorithm and demonstrated its correctness (and prototyped it).

As we said, *μStipula* stems from *Stipula*, a domain-specific language for legal contracts [9]. In Section 6 we have used a basic constraint solver in order to verify reachability of events. Actually, a more powerful constraint solver is needed if we want to address reachability in the full language. Let us explain the issue. *Stipula* admits functions with the following format (we are omitting assets, which are additional parameters, because they are not relevant for this discussion) $\mathtt{@Q}\ \mathtt{f}(\overline{\mathtt{y}})\ (\ E\ )\ \{\ S\ W\ \}\ \Rightarrow\ \mathtt{@Q}'$, with the meaning that, a function invocation can carry arguments that will replace the parameters $\overline{\mathtt{y}}$ and the body $S\ W$ is executed *provided* the condition E holds. That is, if E is always false, the function is never executed and becomes unreachable. Since E may contain field names and formal parameters of the function, understanding whether *every* computation ending at $\mathtt{Q}$ has E unsolvable is pretty difficult.

We are aware of two techniques that might be considered. One is *dynamic symbolic execution* [2] that combines symbolic execution with the collection of constraints on paths and the analysis by means of a solver. Here, the criticality is to gather all the possible computations; perhaps a saturation technique using some (approximation of the) fixpoint might be helpful. Another technique is to encode the operational semantics of *Stipula* contracts into the formal model of Abstract State Machines and then using the corresponding tools for verifying reachability [6]. In this case, while the encoding in Abstract State Machines should be feasible, covering all the possible executions might require over-approximations that threaten the precision of the technique. The detailed study of this solution is under current investigation.

Acknowledgements. I thank Samuele Evangelisti and Alessandro Parenti for the long discussions we had about the reachability analysis. In particular Samuele, which developed the prototype [11] in his master thesis, spotted errors in the preliminary versions of the reachability algorithms. Alessandro has triggered this research by underlying the relevance of finding errors in legal contracts when they are drawn.

REFERENCES

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 2006. *Compilers: Principles, Techniques, and Tools (2nd Edition).* Addison Wesley, Boston, MA, USA.

[2] Eman Alatawi, Harald Søndergaard, and Tim Miller. 2017. Leveraging abstract interpretation for efficient dynamic symbolic execution. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE '17).* IEEE Press, New York, USA, 619–624.

[3] Rajeev Alur and David L. Dill. 1994. A Theory of Timed Automata. *Theor. Comput. Sci.* 126, 2 (1994), 183–235.

[4] Roberto Amadini, Graeme Gange, Peter Schachte, Harald Søndergaard, and Peter J. Stuckey. 2020. Abstract Interpretation, Symbolic Execution and Constraints. In *Recent Developments in the Design and Implementation of Programming Languages (OASIcs, Vol. 86).* Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 7:1–7:19.

[5] Andrew W. Appel and Maia Ginsburg. 1997. *Modern Compiler Implementation in C.* Cambridge University Press, Cambridge, UK.

[6] Paolo Arcaini, Andrea Bombarda, Silvia Bonfanti, Angelo Gargantini, Elvinia Riccobene, and Patrizia Scandurra. 2021. The ASMETA Approach to Safety Assurance of Software Systems. In *Logic, Computation and Rigorous Methods (Lecture Notes in Computer Science, Vol. 12750).* Springer, Cham, Switzerland, 215–238.

[7] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. 2002. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002) (LNCS, Vol. 2404).* Springer, Berlin Heidelberg, Germany, 359–364.

[8] Silvia Crafa and Cosimo Laneve. 2022. Programming Legal Contracts - A Beginners Guide to *Stipula*. In *The Logic of Software. A Tasting Menu of Formal Methods - Essays Dedicated to Reiner Hähnle on the Occasion of His 60th Birthday (Lecture Notes in Computer Science, Vol. 13360).* Springer, Cham, Switzerland, 129–146.

[9] Silvia Crafa, Cosimo Laneve, Giovanni Sartor, and Adele Veschetti. 2023. Pacta sunt servanda: Legal contracts in *Stipula*. *Sci. Comput. Program.* 225 (2023), 102911.

[10] Brian A. Davey and Hilary A. Priestley. 1990. *Introduction to lattices and order.* Cambridge University Press, Cambridge, UK.

[11] Samuele Evangelisti. 2024. Stipula Reachability Analyzer. Available on github: `https://github.com/stipula-language`.

[12] A. Finkel and Ph. Schnoebelen. 2001. Well-structured transition systems everywhere! *Theor. Comput. Sci.* 256 (2001), 63–92. Issue 1-2.

[13] Hans Hansson and Bengt Jonsson. 1990. A Calculus for Communicating Systems with Time and Probabitilies. In *Proceedings of the Real-Time Systems Symposium - 1990.* IEEE Computer Society, New York, USA, 278–287.

[14] Gerard Holzmann. 2003. *Spin model checker, the: primer and reference manual.* Addison-Wesley Professional, Boston, USA.

[15] Franccois Laroussinie, Nicolas Markey, and Philippe Schnoebelen. 2004. Model Checking Timed Automata with One or Two Clocks. In *Proceedings of CONCUR 2004, 15th International Conference in Concurrency Theory (Lecture Notes in Computer Science, Vol. 3170).* Springer, Berlin Heidelberg, Germany, 387–401.

[16] Faron Moller and Chris M. N. Tofts. 1990. A Temporal Calculus of Communicating Systems. In *Proceedings of CONCUR '90 (Lecture Notes in Computer Science, Vol. 458).* Springer, Berlin Heidelberg, Germany, 401–415.

A TECHNICAL MATERIAL

The appendix contains the technical material that has not been included in the paper for ease the reading. Here, we report the proofs with the corresponding statements.

We begin with the soundness of the map $\mathbb{R}_{\mathtt{Q}}$.

Theorem 4.3 (Soundness of $\mathbb{R}_{\mathtt{Q}}$) *Let* $\mathtt{C}$ *be a µStipula contract with initial state* $\mathtt{Q}$ *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle \xrightarrow{\mu} \langle \mathbb{C}'', \mathbb{t}'' \rangle$$

and $\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}''$ *be the underlying clause of* μ. *Then there is* $\mathbb{A} \triangleleft \mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'' \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'')$ *where* $\mathbb{A}$ *is the underlying linear trace of* $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$. *Therefore* $\mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'') \neq \varnothing$.

Proof. By induction on the length of $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$. In the basic case (the length is 0), μ has to be a function name (otherwise the transition $\langle \mathbb{C}', \mathbb{t}' \rangle \xrightarrow{\mu} \langle \mathbb{C}'', \mathbb{t}' \rangle$ cannot have an underlying clause). In this case $\mathtt{Q}' = \mathtt{Q}$ and the statement is immediate by definition of $\mathbb{R}_{\mathtt{Q}}{}^{(0)}(\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}'')$ because the linear trace is $\mathtt{Q}'\cdot\varphi\cdot\mathtt{Q}''$. Assuming the theorem holds for computations of length h, we demonstrate it for computations of length $h+1$.

If $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$ has no transition that is labelled with function names or events (there are only tick transitions) then the proof is similar to the basic case. Otherwise, let $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$ be $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle \xrightarrow{\mu'} \langle \mathbb{C}_2, \mathbb{t}_2 \rangle \xrightarrow{-}{}^* \langle \mathbb{C}', \mathbb{t}' \rangle$ (the transition labelled μ' is the rightmost one that is labelled with a function name or an event) and let $\mathtt{Q}'''\cdot\varphi'\cdot\mathtt{Q}'$ be the underlying clause of the transition μ'. By inductive hypotheses, there is a nonempty $\mathbb{A} \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'''\cdot\varphi'\cdot\mathtt{Q}')$ that is the underlying linear trace of $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle$. There are two subcases:

When $\mu = \mathtt{f}$, by definition of $\mathbb{R}_{\mathtt{Q}}$, $\mathbb{A} \triangleleft \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'' \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'')$.

When $\mu = \mathtt{ev}_{\mathtt{n}}$, let $\mathtt{Q}'\cdot\mathtt{ev}_{\mathtt{n}}\cdot\mathtt{Q}'' \in \mathtt{Q}_1'\cdot\mathtt{f}\cdot\mathtt{Q}_2'$. By the operational semantics, $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle$ must contain a transition $\langle \mathbb{C}_1', \mathbb{t}_1' \rangle \xrightarrow{\mathtt{f}} \langle \mathbb{C}_2', \mathbb{t}_2' \rangle$ where $\mathbb{C}_1' = \mathtt{C}(\mathtt{Q}_1', W \Rightarrow \mathtt{Q}_2', \Psi_1)$ and $W = \mathbb{t}' \gg_{\mathtt{n}} @\mathtt{Q}' \Rightarrow @\mathtt{Q}'' \mid W'$. By inductive hypotheses, $\mathtt{Q}_1'\cdot\mathtt{f}\cdot\mathtt{Q}_2' \in \mathbb{A}$. Hence, by definition of $\mathbb{R}_{\mathtt{Q}}$, $\mathbb{A} \triangleleft \mathtt{Q}'\cdot\mathtt{ev}_{\mathtt{n}}\cdot\mathtt{Q}'' \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{ev}_{\mathtt{n}}\cdot\mathtt{Q}'')$. □

Proposition 4.5. *Let* $\mathtt{C}$ *be a µStipula contract with initial state* $\mathtt{Q}$ *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \xrightarrow{\mu} \longrightarrow^* \xrightarrow{\mu} \langle \mathbb{C}', \mathbb{t}' \rangle \tag{1}$$

such that the underlying clause of the transitions labelled μ *is* $\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''$. *Then* $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') = \mathit{true}$.

Proof. Let the computation (1) be

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle \xrightarrow{\mu} \langle \mathbb{C}_1', \mathbb{t}_1' \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle \xrightarrow{\mu} \langle \mathbb{C}', \mathbb{t}' \rangle$$

and, without loss of generality, assume that $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$ does not contain any transition labelled μ. Let also $\mathbb{A}$ be the underlying linear trace of $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$; $\mu_1, \ldots, \mu_n$ be the transitions labelled with function names or events in $\langle \mathbb{C}_1', \mathbb{t}_1' \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle$ and let $\mathtt{Q}_i\cdot\varphi_i\cdot\mathtt{Q}_{i+1}$, $i \in 1..n$, be the corresponding clauses. Then, by Theorem 4.3,

$$(\mathbb{A} \;;\; \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') \triangleleft \mathtt{Q}_1\cdot\varphi_1\cdot\mathtt{Q}_2 \triangleleft \ldots \triangleleft \mathtt{Q}_n\cdot\varphi_n\cdot\mathtt{Q}_{n+1} \in \mathbb{R}_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'')$$

Let

$$(\mathbb{A} \;;\; \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') \triangleleft \mathtt{Q}_1\cdot\varphi_1\cdot\mathtt{Q}_2 \triangleleft \ldots \triangleleft \mathtt{Q}_n\cdot\varphi_n\cdot\mathtt{Q}_{n+1} = \mathbb{A} \;;\; \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'' \;;\; \mathbb{A}'$$

($\mathbb{A}'$ may be also empty). We define $\mathit{First}(\mathbb{A}) = \{\mathtt{Q} \mid \mathtt{Q}\cdot\varphi\cdot\mathtt{Q}' \in \mathbb{A}\}$. Let $\mathtt{Q}_0\cdot\varphi_0\cdot\mathtt{Q}_1 = \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''$ and let $\mathtt{Q}_j\cdot\varphi_j\cdot\mathtt{Q}_{j+1}$, $0 \leq j \leq n$, be the clause with the smallest j such that $\mathtt{Q}_{j+1} \in \mathit{First}(\mathbb{A} \;;\; \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'')$. One such clause must exists because $\mathtt{Q}_{n+1} = \mathtt{Q}'$. We observe that $\mathtt{Q}_j\cdot\varphi_j\cdot\mathtt{Q}_{j+1} \in \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'' \;;\; \mathbb{A}'$. In fact, if $j = 0$, this is immediate; if $j > 0$ then, by contradiction, assume $\mathtt{Q}_j\cdot\varphi_j\cdot\mathtt{Q}_{j+1} \notin \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'' \;;\; \mathbb{A}'$. This means that, by definition of "$\triangleleft$", $\mathtt{Q}_j\cdot\varphi_j\cdot\mathtt{Q}_{j+1} \in \mathbb{A}$. However, this means that $\mathtt{Q}_j\cdot\varphi_j\cdot\mathtt{Q}_{j+1}$ is not the clause with the least j, since also $\mathtt{Q}_{j-1}\cdot\varphi_{j-1}\cdot\mathtt{Q}_j$ satisfies the constraint that $\mathtt{Q}_j \in \mathit{First}(\mathbb{A} \;;\; \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'')$. Absurd.

Then, there exists $\mathtt{Q}_{j+1}\cdot\varphi'\cdot\mathtt{Q}'''$ such that $\mathtt{Q}_{j+1}\cdot\varphi'\cdot\mathtt{Q}''' \in \mathbb{A} \;;\; \mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}''$. $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}'\cdot\mathtt{f}\cdot\mathtt{Q}'') = \mathit{true}$ follows immediately. □

The correctness of $\mathbb{T}_{\mathtt{Q}}$ requires two preliminary statements about acyclic functions. The first one, Lemma A.1, is a property about the format of underlying linear trace containing acyclic functions. In particular, if $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2) = \mathit{false}$ then every linear trace $\mathbb{A} \;;\; \mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2 \;;\; \mathbb{A}$ has the property that no clause in $\mathbb{A}$ is in $\mathbb{A}'$ and, conversely, no clause in $\mathbb{A}'$ is in $\mathbb{A}$. This decoupling property is relevant for computing the logical times of linear traces containing acyclic functions and establishing the consistency constraint in the definition of $\mathbb{T}_{\mathtt{Q}}$, as highlighted in Lemma A.2.

Lemma A.1. *Let* $\mathtt{C}$ *be a µStipula contract with initial state* $\mathtt{Q}$ *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle \xrightarrow{\mu} \langle \mathbb{C}_2, \mathbb{t}_1 \rangle \longrightarrow^* \langle \mathbb{C}_2', \mathbb{t}_2 \rangle \tag{2}$$

with $\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2$ *being the underlying clause of the transition labelled* μ *and* $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2) = \mathit{false}$. *Then the underlying linear trace of* (2) *is* $\mathbb{A} \;;\; \mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2 \;;\; \mathbb{A}'$ *where* $\mathbb{A}$ *and* $\mathbb{A}'$ *contain the underlying clauses in* $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$ *and in* $\langle \mathbb{C}_1', \mathbb{t}_1 \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle$, *respectively.*

As a consequence, no clause in $\mathbb{A}$ *occurs in* $\mathbb{A}'$ *and, conversely, no clause in* $\mathbb{A}'$ *occurs in* $\mathbb{A}$.

Proof. Since $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2) = \mathit{false}$ then, by Corollary 4.5, the computation (2) has a unique transition whose underlying clause is $\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2$. Therefore, the underlying linear trace of (2), which exists by Theorem 4.3, may be written $\mathbb{A} \;;\; \mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2 \;;\; \mathbb{A}'$. By definition of underlying linear trace, $\mathbb{A}$ contains the underlying clauses in $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$.

We demonstrate that every underlying clause $\mathtt{Q}_1'\cdot\varphi\cdot Q_2'$ of transitions in $\langle \mathbb{C}_2, \mathbb{t}_1 \rangle \longrightarrow^* \langle \mathbb{C}_2', \mathbb{t}_2 \rangle$ does not occur in $\mathbb{A}$. By contradiction, assume that $\mathtt{Q}_1'\cdot\varphi\cdot\mathtt{Q}_2' \in \mathbb{A}$. Therefore $\mathbb{A} = \mathbb{A}_1 \;;\; \mathtt{Q}_1'\cdot\varphi\cdot\mathtt{Q}_2' \;;\; \mathbb{A}_2$ and the computation $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$ may be decomposed as follows:

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1', \mathbb{t}_1' \rangle \xrightarrow{\mu'} \langle \mathbb{C}_1'', \mathbb{t}_1' \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$$

where $\langle \mathbb{C}_1', \mathbb{t}_1' \rangle \xrightarrow{\mu'} \langle \mathbb{C}_1'', \mathbb{t}_1' \rangle$ is the leftmost transition whose underlying clause is $\mathtt{Q}_1'\cdot\varphi\cdot\mathtt{Q}_2'$. According to the operational semantics, there are two subcases: either (*a*) there is a transition in $\langle \mathbb{C}_1'', \mathbb{t}_1' \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$ whose underlying clause is $\mathtt{Q}_2'\cdot\varphi'\cdot\mathtt{Q}_2''$ or (*b*) $\mathtt{Q}_2' = \mathtt{Q}_1$ ($\mathtt{Q}_1$ the initial state of $\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2$). Notice that, in case (*a*), $\mathtt{Q}_2'\cdot\varphi'\cdot\mathtt{Q}_2'' \in \mathbb{A}$. We conclude by observing that, by Definition 4.4, both in case (*a*) and in case (*b*), $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}_1\cdot\mathtt{f}\cdot\mathtt{Q}_2) = \mathit{true}$, which is absurd. □

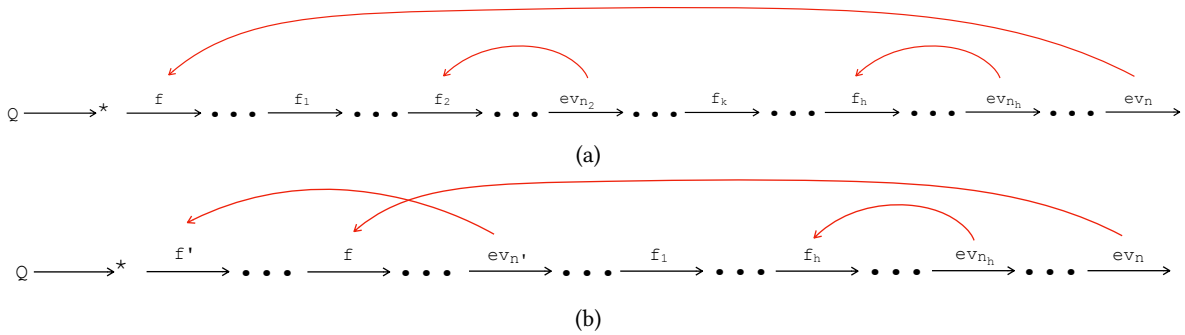

Figure 1: Two computations in *μStipula*

Lemma A.2. *Let* C *be a μStipula contract with initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle \xrightarrow{\mu} \langle \mathbb{C}'_1, \mathbb{t}_1 \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle \xrightarrow{\mu'} \langle \mathbb{C}'_2, \mathbb{t}_2 \rangle \tag{3}$$

where $\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2$ *is the underlying clause of* μ, $\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2$ *is the underlying clause of* μ', *with* $\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2 \in \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2$ *and* $\circlearrowleft_{\mathtt{C}}(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) = \mathit{false}$. *Then let* $\mathbb{A}$ *be the underlying linear trace of* (3):

(*a*) $\mathbb{A} = \mathbb{A}' \ ; \ \mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2$;
(*b*) $\mathfrak{T}_\nu(\mathbb{A}') \leq \mathfrak{T}_\nu(\mathbb{A})$ *is solvable.*

Proof. As regards (*a*), since $\circlearrowleft_{\mathtt{C}}(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) = \mathit{false}$, the computation (3) has exactly one transition whose underlying clause is $\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2$ and, consequently, one transition whose underlying clause is $\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2$. By Definition 4.2, $\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2$ must be the last clause of $\mathbb{A}$ hence the thesis.

As regards (*b*), we may decompose $\mathbb{A}$ as follows

$$\mathbb{A} = \mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'' \ ; \ \mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2 \ .$$

By Lemma A.1, the underlying clauses of $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}_1, \mathbb{t}_1 \rangle$ occur in $\mathbb{A}'$ and those of $\langle \mathbb{C}'_1, \mathbb{t}_1 \rangle \longrightarrow^* \langle \mathbb{C}_2, \mathbb{t}_2 \rangle$ occur in $\mathbb{A}''$. There are two subcases: (*b1*) the events in $\mathbb{A}''$ refer to functions in $\mathbb{A}''$ and (*b2*) there is at least one event in $\mathbb{A}''$ that refers to a function in $\mathbb{A}'$. We discuss them separately.

Subcase (*b1*) is displayed by the drawing in Figure 1(a) where the labels indicate the functions (we omit states) and events and the backwards red arrow indicate the connection of an event with its own function. In this case, by definition of $\mathfrak{U}_\nu$, $\mathfrak{U}_\nu(\mathbb{A}) = \mathfrak{U}_\nu(\mathbb{A}') + \nu(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) + k$, where $\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2 = \mathsf{now} + k \gg_{\mathtt{n}} @\mathtt{Q}'_1 \Rightarrow @\mathtt{Q}'_2$. (Since the function is acyclic and because of the operational semantics, the other events of $\mathtt{f}$ in $\mathbb{A}''$, if any, have smaller time expressions.) Next, let us compute $\mathfrak{U}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'')$; we obtain

$$\begin{aligned}&\mathfrak{U}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'')\\ &\quad = \mathfrak{U}_\nu(\mathbb{A}') + \nu(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) + \left(\textstyle\sum_{i\in 1..m} \nu(\mathtt{Q}_i{\cdot}\mathtt{f}_i{\cdot}\mathtt{Q}'_i)\right) + \left(\textstyle\sum_{j\in 1..m'} k_j\right) .\end{aligned}$$

where $\mathtt{Q}_i{\cdot}\mathtt{f}_i{\cdot}\mathtt{Q}'_i$, $i \in 1..m$, are the clauses in $\mathbb{A}''$ of transitions labelled with function names and k_j, $j \in 1..m$ are the maximal time expressions (without now) of events corresponding to that functions and occurring in $\mathbb{A}''$ ($k_j = 0$ if no event occurrs). By the operational semantics $\sum_{j\in 1..m} k_j \leq k$; therefore $\mathfrak{U}_\nu(\mathbb{A}) \leq \mathfrak{U}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'')$ by instantiating every variable in $\sum_{i\in 1..m} \nu(\mathtt{Q}_i{\cdot}\mathtt{f}_i{\cdot}\mathtt{Q}_i)$ with 0. Hence $\mathfrak{T}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'') \leq \mathfrak{T}_\nu(\mathbb{A})$ is solvable.

Subcase (*b2*) may be displayed by the drawing in Figure 1(b) where we use the same notations as before. Let $\mathtt{Q}'_3{\cdot}\mathtt{ev'_n}{\cdot}\mathtt{Q}'_4$ be the *rightmost* event in $\mathbb{A}''$ that belongs to a function $\mathtt{Q}_3{\cdot}\mathtt{f}'{\cdot}\mathtt{Q}_4$ in $\mathbb{A}'$. In this case we may further decompose $\mathbb{A}'$ into $\mathbb{A}'_1 \ ; \ \mathtt{Q}_3{\cdot}\mathtt{f}'{\cdot}\mathtt{Q}_4 \ ; \ \mathbb{A}'_2$ and $\mathbb{A}''$ into $\mathbb{A}''_1 \ ; \ \mathtt{Q}'_3{\cdot}\mathtt{ev'_n}{\cdot}\mathtt{Q}'_4 \ ; \ \mathbb{A}''_2$. Then, by definition of $\mathfrak{U}_\nu$, we have

$$\begin{aligned}&\mathfrak{U}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'') =\\ &\quad \mathfrak{U}_\nu(\mathbb{A}'_1 \ ; \ \mathtt{Q}_3{\cdot}\mathtt{f}'{\cdot}\mathtt{Q}_4) + k' + \left(\textstyle\sum_{i\in 1..m} \nu(\mathtt{Q}_i{\cdot}\mathtt{f}_i{\cdot}\mathtt{Q}'_i)\right) + \left(\textstyle\sum_{j\in 1..m'} k_j\right) .\end{aligned}$$

where k' is the maximum time expression of events of $\mathtt{Q}_3{\cdot}\mathtt{f}'{\cdot}\mathtt{Q}_4$ in $\mathbb{A}'_2 \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}''_1$ and the two summands are as in the subcase (*b1*). Notice that the variable $\nu(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2)$ does not occur in the above logical time, while it occurs in $\mathfrak{U}_\nu(\mathbb{A})$. Therefore there exists a ground substitution of $\nu(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2)$ such that $\mathfrak{U}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'') \leq \mathfrak{U}_\nu(\mathbb{A})$ is solvable. Hence $\mathfrak{T}_\nu(\mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}'') \leq \mathfrak{T}_\nu(\mathbb{A})$ is solvable, as well. □

Every property that is necessary for the proof of the correctness of $\mathbb{T}_{\mathtt{Q}}$ has been proved. Hence the theorem.

Theorem 5.4 (Soundness of $\mathbb{T}_{\mathtt{Q}}$). *Let* C *be a μStipula contract with initial state* Q *and initial configuration* $\langle \mathbb{C}, \mathbb{t} \rangle$. *Let also*

$$\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle \xrightarrow{\mu} \langle \mathbb{C}'', \mathbb{t}' \rangle$$

and $\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}''$ *be the underlying clause of* μ. *Then there is* $\mathbb{A} \triangleleft \mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}'' \in \mathbb{T}_{\mathtt{Q}}(\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}'')$ *where* $\mathbb{A}$ *is the underlying linear trace of* $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$. *Therefore* $\mathbb{T}_{\mathtt{Q}}(\mathtt{Q}'{\cdot}\varphi{\cdot}\mathtt{Q}'') \neq \varnothing$.

Proof. By induction on the length of $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$. The proof is similar to that of Theorem 4.3 except for the case of $\mu = \mathtt{ev_n}$ and $\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2 \in \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2$ and $\circlearrowleft_{\mathtt{Q}}(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) = \mathit{false}$.

In this case, the underlying linear trace of $\langle \mathbb{C}, \mathbb{t} \rangle \longrightarrow^* \langle \mathbb{C}', \mathbb{t}' \rangle$ is $\mathbb{A} = \mathbb{A}' \ ; \ \mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2 \ ; \ \mathbb{A}''$ and, by Lemma A.2,

$$\mathfrak{T}_\nu(\{\mathbb{A}\}) \leq \mathfrak{T}_\nu(\mathbb{T}_{\mathtt{Q}}(\mathtt{Q}_1{\cdot}\mathtt{f}{\cdot}\mathtt{Q}_2) \triangleleft \mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2)$$

is solvable.

Hence, by definition of $\mathbb{T}_{\mathtt{Q}}$, $\mathbb{A} \triangleleft \mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2 \in \mathbb{T}_{\mathtt{Q}}(\mathtt{Q}'_1{\cdot}\mathtt{ev_n}{\cdot}\mathtt{Q}'_2)$. □