Reasoning over Streams of Events with Delayed Effects: Supplementary Material

Periklis Mantenoglou, Manolis Pitsikalis and Alexander Artikis

Section 1 provides the reasoning algorithms of RTEC$^{\rightarrow}$ that were omitted from the paper. Section 2 contains complete proofs for the propositions concerning the correctness of RTEC$^{\rightarrow}$ and its complexity. Moreover, we outline the cost of reasoning using rules (α)–(γ) of the paper, i.e., a naive implementation that is not used by RTEC$^{\rightarrow}$. Section 3 demonstrates that RTEC$^{\rightarrow}$ supports events that may lead to future terminations of domain properties. Section 4 provides instructions for reproducing the experiments of the empirical evaluation.

1 Reasoning Algorithms

We provide the reasoning algorithms of RTEC$^{\rightarrow}$ that were omitted from the paper. First, we present the algorithm of RTEC$^{\rightarrow}$ for processing the attempt events that were cached at the previous query time (see line 2 of Algorithm 2 of the paper). Second, we present the algorithm that caches the attempt events that may lead to future initiations at the next query time (see line 8 of Algorithm 2). Third, we demonstrate how RTEC$^{\rightarrow}$ processes cd-components that contain cycles with f-edges (see line 13 of Algorithm 1 of the paper).

1.1 Processing cached attempt events

Given $\mathsf{fi}(F=V, F=V', R)$, Algorithm I presents the steps for determining which one of the cached attempt events, that were generated by an initiation of $F=V$ and cached at the previous query time q_{i-1}, if any, may mark a future initiation of $F=V'$ at the current query time q_i. ω denotes the window size. $Attempts[F=V]$ is a sorted list containing the time-stamps of the attempt events that were generated by initiations of $F=V$ at q_{i-1}. $I_{q_{i-1}}[F=V]$ is a sorted list containing the intervals of $F=V$ that were computed by RTEC$^{\rightarrow}$ at q_{i-1}.

Algorithm I findAttempt

Input: $F=V, R$
Global: $q_i, \omega, Attempts[F=V], I_{q_{i-1}}[F=V]$
Output: T_{att}

```
1:  T_att ← null
2:  if q_i−ω+1 in I_{q_{i-1}}[F=V] then
3:      if p(F=V) then
4:          for each T in Attempts[F=V] do
5:              if T > q_i−ω and T−R ≤ q_i−ω then T_att ← T
6:              else if T−R > q_i−ω then return T_att
7:      else
8:          [T_s, T_f) ← getIntervalContainingTimepoint(I_{q_{i-1}}[F=V], q_i−ω+1)
9:          for each T_att in Attempts[F=V] do
10:             if T_att > q_i−ω and T_att−R == T_s then return T_att
11: return T_att
```

First, RTEC$^{\rightarrow}$ checks whether there is an interval in $I_{q_{i-1}}[F=V]$ that contains the beginning of the current window $q_i-\omega+1$ (see line 2 of Algorithm I). If there is no such interval, then $F=V$ was 'broken' before $q_i-\omega+1$, and thus the cached attempts of initiating $F=V'$ have been cancelled. As a result, RTEC$^{\rightarrow}$ discards all cached attempt events (see lines 1 and 11). If there is an interval in $I_{q_{i-1}}[F=V]$ that contains $q_i-\omega+1$, then $F=V$ holds up to $q_i-\omega+1$, and thus there may be a cached attempt of initiating $F=V'$ that is not cancelled. If the future initiations of $F=V'$ may be postponed, then RTEC$^{\rightarrow}$ returns the time-stamp T_{att} of

the last attempt event *att* that takes place in the window, while the initiation of $F = V$ that generated *att* took place before the window (see lines 3–6). Earlier attempts of initiating $F = V'$ are postponed by the initiation of $F = V$ that generated attempt *att*, and are thus discarded. If the future initiations of $F = V'$ may not be postponed, then we keep the first attempt event *att* that was generated before the window and takes place in the window. *att* was produced by the initiation of $F = V$ that coincides with the starting point of the interval that contains $q_i - \omega + 1$. Therefore, we retrieve that interval from $I_{q_{i-1}}[F = V]$ and keep the cached attempt event *att* that was generated by the initiation of $F = V$ that coincides with the starting point of the retrieved interval (see lines 7–10). Subsequent attempts of initiating $F = V'$ will be cancelled, at the latest, by T_{att}, i.e., the time-stamp of the derived attempt *att*, due to the potential future initiation of $F = V'$ at T_{att}.

1.2 Caching attempt events

Algorithm II shows the steps of RTEC$^{\rightarrow}$ for deriving, at q_i, the attempt events that may initiate $F = V'$, based on $\mathsf{fi}(F = V, F = V', R)$, at the next query time q_{i+1}. The sorted lists $I[F = V]$ and $IP[F = V]$ contain, respectively, the intervals and the initiation points of $F = V$ that were derived at q_i.

Algorithm II computeAttempts

Input: $F = V, R$
Global: $I[F = V], IP[F = V]$
Output: $Attempts[F = V]$

```
1: Attempts[F = V] ← [ ]
2: if p(F = V) then
3:     for each T in IP[F = V] do
4:         Attempts[F = V].add(T+R)
5: else
6:     for each [T_s, T_f) in I[F = V] do
7:         Attempts[F = V].add(T_s+R)
8: cache(Attempts[F = V])
```

We distinguish between two cases, depending on whether the future initiations of $F = V'$ may be postponed or not. If they may be postponed, then we have an attempt event R time-points after every initiation point of $F = V$ (see lines 2–4 of Algorithm II). Otherwise, if the future initiations of $F = V'$ may not be postponed, then only the initiation points of $F = V$ that are starting points of maximal intervals of $F = V$ may induce a future initiation of $F = V'$. Any initiation point occurring within an interval of $F = V$ would not postpone the future initiation of $F = V'$ that was generated by the initiation of $F = V$ at the starting point of that interval, and thus does not need to be considered. Therefore, for each starting point T_s of an interval of $F = V$, we add time-point $T_s + R$ in $Attempts[F = V]$ (see lines 5–7). RTEC$^{\rightarrow}$ caches the time-stamps in $Attempts[F = V]$, so that they will be available at the next query time (see line 8).

1.3 Computing future initiations with cycles

When processing a cd-component that contains a cycle with f-edges, RTEC$^{\rightarrow}$ employs an extended version of RTEC$_{\circ}$, featuring an algorithm for evaluating future initiations in the presence of cycles (see cyclicFI in line 13 of Algorithm 1 of the paper). For each fluent-value pair (FVP) $F = V$ in the cd-component, cyclicFI evaluates whether or not $F = V$ is initiated at each time-point T of the window using initiatedAtCyclic, i.e., an algorithm for evaluating an initiation of an FVP that is in a cd-component with cycles. When evaluating an initiatedAt rule defining $F = V$, initiatedAtCyclic does not assume that the maximal intervals of the FVPs appearing in body of the rule are certainly in the cache. Moreover, when evaluating a future initiation of $F = V$, based on, e.g., $\mathsf{fi}(F = V^d, F = V, R)$, where $V^d \neq V$, initiatedAtCyclic does not assume that all the initiation points of $F = V^d$ that may lead to a future initiation of $F = V$ are in the cache. Additionally, initiatedAtCyclic does not assume that all the points that may cancel a future initiation of $F = V$ are cached.

Algorithm III presents initiatedAtCyclic. First, we make sure that time-point T falls within the current window (see line 1 of Algorithm III). If so, RTEC$^{\rightarrow}$ checks whether the initiation of FVP $F = V$ at T has

Algorithm III initiatedAtCyclic($F = V$, T)

Input: $q_i, \omega, AttemptOfFVP$.
Output: true/false.

```
1:  if T ≤ q_i−ω or T > q_i then return false
2:  if cachedInit(F = V, T, +) then return true
3:  if cachedInit(F = V, T, −) then return false
4:  for each body in bodiesOf(initiatedAt(F = V, T)) do
5:      if sat(body) then
6:          updateCache(cachedInit(F = V, T, +))
7:          return true
8:  for each fi(F = V^d, F = V, R) do
9:      if T==AttemptOfFVP[F = V^d] then
10:         if not cancelledCyclic(F = V^d, q_i−ω, T) then
11:             updateCache(cachedInit(F = V, T, +))
12:             return true
13:     if T > q_i−ω+R and initiatedAtCyclic(F = V^d, T−R) then
14:         if not cancelledCyclic(F = V^d, T−R, T) then
15:             updateCache(cachedInit(F = V, T, +))
16:             return true
17: updateCache(cachedInit(F = V, T, −))
18: return false
```

been evaluated at a previous step (see lines 2–3). If the cache of RTEC$^{\rightarrow}$ contains cachedInit($F = V$, T, $+$) (resp. cachedInit($F = V$, T, $-$)), then it has been evaluated that $F = V$ is initiated (not initiated) at T. Otherwise, the initiation of $F = V$ at T has not been assessed at a previous step, and thus RTEC$^{\rightarrow}$ proceeds with its evaluation, starting with the initiatedAt rules of $F = V$ (see lines 4–7). The intervals of some FVPs appearing in the conditions of these rules may not be in the cache, due to cyclic dependencies in the definition of $F = V$. To address this issue, the truth values of holdsAt conditions are derived using RTEC$_\circ$. If an initiatedAt rule of $F = V$ is satisfied at T, then RTEC$^{\rightarrow}$ updates the cache with the derived initiation point (see line 6).

If an initiation of $F = V$ at T could not be computed based on the initiatedAt rules of $F = V$, then RTEC$^{\rightarrow}$ iterates over the fi facts that have $F = V$ as their second argument, in order to check whether one of them leads to a future initiation of $F = V$ at T (see line 8). Given the fact fi($F = V^d$, $F = V$, R), we have a future initiation of $F = V$ at T if (i) there is a cached attempt of initiating $F = V$, due to an earlier initiation of $F = V^d$, the time-stamp of the attempt, i.e., $AttemptOfFVP[F = V^d]$, coincides with T, and the future initiation of $F = V$ is not cancelled between the start of the window and T (see lines 9–12), or (ii) $F = V^d$ is initiated at $T-R$, and the future initiation of $F = V$ is not cancelled between $T-R$ and T (see lines 13–16). In both cases, RTEC$^{\rightarrow}$ consults the cache to avoid redundant evaluations of cancellation points (see cancelledCyclic in lines 10 and 14), and updates the cache with the derived initiation point, if any (see lines 11 and 15). If RTEC$^{\rightarrow}$ failed to compute an initiation point of $F = V$ at T, then we conclude that T is not an initiation point of $F = V$, and update the cache accordingly (see line 17).

2 Proofs of Correctness and Complexity

We provide complete proofs of Propositions 3 and 4 of the paper, which correspond to Propositions I and II of this document.

2.1 Correctness of RTEC$^{\rightarrow}$

Proposition I (Correctness of RTEC$^{\rightarrow}$): RTEC$^{\rightarrow}$ computes the maximal intervals of the FVPs of an event description, and no other interval. ▲

We show that Proposition I holds for an FVP $F = V$, such that fi($F = V$, $F = V'$, R), by proving the correctness of the novel operations of RTEC$^{\rightarrow}$. We assume that RTEC$^{\rightarrow}$ evaluates initiatedAt and terminatedAt rules (see

lines 4–5 of Algorithm 1 of the paper) and constructs the maximal intervals of FVPs (see line 10 of Algorithm 1) correctly, because these operations are inherited from RTEC.

First, we prove that RTEC$^{\rightarrow}$ transfers the attempt events between windows that are required for correct FVP interval computation, and no other attempt event (see Lemma I). Second, we show that RTEC$^{\rightarrow}$ computes all future initiations of $F = V'$, and no other future initiation (see Lemma II). Third, at the time of constructing the maximal intervals of $F = V$, lists $IP[F = V]$ and $TP[F = V]$ contain the initiations and terminations of $F = V$, and no other point.

Lemma I (Correctness of transferring attempt events): RTEC$^{\rightarrow}$ transfers the attempt events between windows that are required for correct FVP interval computation, and no other attempt event. ▲

Proof. We prove that, at query time q_i, RTEC$^{\rightarrow}$ selects the attempt event that may mark a future initiation of $F = V'$, if any, as specified in the definition of fi (see Definition 5 of the paper). According to this definition, we have a future initiation of $F = V'$ at T_{att} iff there is an initiation of $F = V$ at $T_{att}-R$ and there is no cancellation point of the future initiation of $F = V'$ between $T_{att}-R$ and T_{att}. Given window size ω, we will prove that RTEC$^{\rightarrow}$ derives an attempt event *att* iff its time-stamp T_{att} satisfies the following conditions:

1. $T_{att} > q_i-\omega$.
2. $T_{att}-R \leq q_i-\omega$.
3. $F = V$ is initiated at $T_{att}-R$.
4. The future initiation of $F = V'$ is not cancelled between $T_{att}-R$ and the start of the window $q_i-\omega+1$.
5. The future initiation of $F = V'$ is not cancelled between $q_i-\omega+1$ and T_{att} by a future initiation of $F = V'$ that was generated before $q_i-\omega+1$.

Conditions 1 and 2 express that the attempt *att* falls inside the current window, while the initiation of $F = V$ that generated *att* is before the window. If this initiation of $F = V$ were also inside the window, then we would be able to perform the computation at q_i without transferring *att*. Conditions 3 and 4 express that the future initiation of $F = V'$ at T_{att} was staged and not cancelled up to $q_i-\omega+1$. The information before $q_i-\omega+1$ does not change at the q_i, and thus, if conditions 3 and 4 were violated at q_{i-1}, they are still violated at q_i. Condition 5 expresses that we should not select attempt event *att* if there is another attempt that satisfies conditions 1–4 and cancels the future initiation of $F = V'$ at T_{att}. This condition addresses the cases where we have several initiations of $F = V$ at $T_1, \ldots T_n$, where $T_n-T_1 < R$, and the future initiations of $F = V'$ may not be postponed. In such cases, only the attempt generated by the initiation of $F = V$ at T_1 is selected, because the remaining ones will be cancelled, at the latest, by the future initiation of $F = V'$ at T_1+R.

RTEC$^{\rightarrow}$ derives the attempt event that should be kept using Algorithm I. Suppose that the output T_{att} of Algorithm I is not *null*, i.e., T_{att} is the time-stamp of an attempt event. If the future initiations of $F = V'$ may be postponed, then we have $T_{att} > q_i-\omega$ and $T_{att}-R \leq q_i-\omega$ (see line 4 of Algorithm I). If the future initiations of $F = V'$ may not be postponed, then we have $T_{att} > q_i-\omega$, while $T_{att}-R$ coincides with the starting point of an interval that contains the start of the window (see lines 8 and 10), and thus $T_{att}-R \leq q_i-\omega$. In both cases, conditions 1 and 2 hold for T_{att}.

Next, we prove that $F = V$ is initiated at $T_{att}-R$. T_{att} is one of the time-stamps that were cached at previous query time by Algorithm II of RTEC$^{\rightarrow}$. If the future initiations of $F = V'$ may be postponed, then Algorithm II caches all time-points $T+R$, such that T is an initiation point of $F = V$. Otherwise, if the future initiations of $F = V'$ may not be postponed, then Algorithm II caches all time-points T_s+R, such that T_s is the starting point of an interval of $F = V$. Since all starting points of intervals of $F = V$ are initiation points of $F = V$, it holds that, if Algorithm II caches time-point T_s+R, then $F = V$ is initiated at T_s. As a result, all time-stamps in list $Attempts[F = V]$ of Algorithm I are R time-points after an initiation of $F = V$. Since the time-stamp T_{att} computed by Algorithm I is one of the items of list $Attempts[F = V]$, $T_{att}-R$ is an initiation point of $F = V$, and thus condition 3 holds.

T_{att} satisfies conditions 1–3. We prove that RTEC$^{\rightarrow}$ derives an attempt *att* with time-stamp T_{att} iff the future initiation of $F = V'$ is not cancelled between $T_{att}-R$ and the start of the window $q_i-\omega+1$ (condition 4), and it is not cancelled between $q_i-\omega+1$ and T_{att} by a future initiation of $F = V'$ that was generated before

$q_i{-}\omega{+}1$ (condition 5). Assume that RTEC$^{\rightarrow}$ derives attempt att. Then, T_{att} is computed by Algorithm I, meaning that there is an interval $[T_s, T_f)$ of $F = V$ that contains $q_i{-}\omega{+}1$ (see line 2 of Algorithm I). The existence of interval $[T_s, T_f)$ implies that $F = V$ is not 'broken' between T_s and $q_i{-}\omega{+}1$, as, otherwise, $[T_s, T_f)$ would have been segmented. If the future initiations of $F = V'$ may not be postponed, then $T_{att}{-}R = T_s$ (see line 10). Therefore, the future initiation of $F = V'$ is not cancelled between $T_{att}{-}R$ and $q_i{-}\omega{+}1$, i.e., condition 4 holds, and it is not cancelled between $q_i{-}\omega{+}1$ and T_{att} by a future initiation of $F = V'$ that was induced earlier, because such a future initiation would have been cancelled before interval $[T_s, T_f)$, i.e., condition 5 holds. Otherwise, if the future initiations of $F = V'$ may be postponed, then $T_{att}{-}R$ coincides with the last initiation of $F = V$ in $[T_s, q_i{-}\omega{+}1)$ (see lines 5–6). Thus, $F = V$ is not 'broken' between $T_{att}{-}R$ and $q_i{-}\omega{+}1$, i.e., condition 4 holds, and earlier initiations of $F = V$ do not cancel the future initiation of $F = V'$ at T_{att}, as such earlier initiations would be postponed by the initiation of $F = V$ at $T_{att}{-}R$, i.e., condition 5 holds. Therefore, if RTEC$^{\rightarrow}$ computes an attempt event att, then its time-stampt T_{att} satisfies conditions 4 and 5.

Assume that T_{att} satisfies conditions 1–5. Based on condition 4, $F = V$ is not 'broken' between $T_{att}{-}R$ and $q_i{-}\omega{+}1$, and thus there is a maximal interval $[T_s, T_f)$ of $F = V$ that contains the start of the window $q_i{-}\omega{+}1$. If the future initiations of $F = V'$ may not be postponed, then RTEC$^{\rightarrow}$ computes $T_s{+}R$. Since $F = V$ holds continuously in $[T_s, T_f)$ and in $[T_{att}{-}R, T_f)$, and T_s is the starting point of a maximal interval, it holds that $T_{att}{-}R \geq T_s$. If $T_{att}{-}R > T_s$, then, since the future initiations of $F = V'$ may not be postponed, the future initiation of $F = V'$ that was induced at T_s will cancel the future initiation of $F = V'$ at T_{att}, meaning that T_{att} does not satisfy condition 5, which is a contraction. Therefore, we have $T_{att}{-}R = T_s$, and thus RTEC$^{\rightarrow}$ derives time-point T_{att}. If the future initiations of $F = V'$ may be postponed, then RTEC$^{\rightarrow}$ computes the time-stamp T of the last attempt of initiating $F = V'$ that was induced before $q_i{-}\omega{+}1$. Earlier attempts of initiating $F = V'$ are cancelled, because the corresponding future initiations of $F = V'$ are postponed by the initiation of $F = V$ that generated the attempt at T. Thus, these attempts do not satisfy condition 4, and the attempt event at T is only attempt that satisfies conditions 4 and 5. As a result, T coincides with T_{att}. Therefore, if T_{att} satisfies conditions 1–5, then RTEC$^{\rightarrow}$ derives the attempt event att that takes place at T_{att}.

We proved that RTEC$^{\rightarrow}$ derives an attempt event att iff its time-stamp T_{att} satisfies conditions 1–5. ◊

Lemma II (Correctness of future initiation computation): Given fi$(F = V, F = V', R)$, RTEC$^{\rightarrow}$ computes all future initiations of $F = V'$, and no other future initiation. ▲

Proof. We start by presenting the proof for an acyclic cd-component containing the FVPs with fluent F; moreover, we assume that the vertex of FVP $F = V$ has no incoming f-edges, i.e., there are no future initiations of $F = V$. We will relax these assumptions in subsequent steps of the proof.

First, we prove that, at the time of evaluating the future initiations of $F = V'$ based on fi$(F = V, F = V', R)$ (see line 7 of Algorithm 1 of the paper), the list $IP[F = V]$ of cached initiations of $F = V$ contains all initiations of $F = V$, and no other points, and the list $TP[F = V]$ of cached terminations of $F = V$ contains all terminations of $F = V$, except from those stemming from future initiations of $F = V'$, and no other points.

Correctness of $IP[F = V]$. $IP[F = V]$ contains the correct immediate initiations of $F = V$, because RTEC$^{\rightarrow}$ evaluated these initiations earlier, in line 4 of Algorithm 1, using an operation inherited from RTEC, which we have assumed to be correct. Since there are no future initiations of $F = V$, we conclude that $IP[F = V]$ contains all initiations of $F = V$, and no other points.

Correctness of $TP[F = V]$. $TP[F = V]$ contains the correct immediate terminations of $F = V$, because these terminations were computed at an earlier step by RTEC (see line 5 of Algorithm 1). $TP[F = V]$ does not contain future terminations of $F = V$ that stem from future initiations of an FVP $F = V_j$, where $V_j \neq V'$, as such future initiations do not terminate $F = V$. Suppose that we have fi$(F = V_i, F = V_j, R')$, where $V_i \neq V_j \neq V$ and $V_j \neq V'$. (We assume that $V_i \neq V$ without loss of generality, because there may be, at most, one fi with $F = V$ as its first argument. If there were a fact fi$(F = V, F = V'', R'')$, where $R'' > R$, then such a fact would never result in a future initiation of $F = V''$.) In order for a future initiation of $F = V_j$ to not get cancelled, $F = V_i$ must hold up to the time-point of the future initiation. Since fluents have at most one value at a time, it is not possible for $F = V$ to hold at the time of the future initiation of $F = V_j$. Therefore, a future initiation

of $F = V_j$ cannot terminate $F = V$. As a result, the future initiations of $F = V'$ are the only future initiations of F that may terminate $F = V$. Therefore, the only terminations of $F = V$ that are missing from $TP[F = V]$, if any, are the ones stemming from future initiations of $F = V'$.

Second, we prove that, given lists $IP[F = V]$ and $TP[F = V]$, RTEC$^{\rightarrow}$ derives all the future initiations of $F = V'$, and no other time-points. According to Definition 5, there is a future initiation of $F = V'$ based on $\mathsf{fi}(F = V, F = V', R)$ at $T{+}R$ iff $F = V$ is initiated at T and there is no cancellation point of the future initiation of $F = V'$ between T and $T{+}R$.

Correctness of future initiation evaluation at T_{att}. Suppose that there is a cached attempt event *att* with time-stampt T_{att}. RTEC$^{\rightarrow}$ computes a future initiation of $F = V'$ at T_{att} iff there is no cached cancellation point of the future initiation of $F = V'$ between the start of the window $q_i{-}\omega{+}1$ and T_{att} (see lines 3–4 of Algorithm 2 of the paper). Since this is the earliest potential future initiation of $F = V'$ at q_i, $TP[F = V]$ contains all the termination points of $F = V$ that are between $q_i{-}\omega{+}1$ and T_{att}, and no other points. Therefore, the cache contains all cancellation points of future initiations of $F = V'$ between $q_i{-}\omega{+}1$ and T_{att}, and no other points. Moreover, there is no cancellation point between $T_{att}{-}R$ and T_{att} that was generated before $q_i{-}\omega{+}1$ (see Lemma I). Therefore, RTEC$^{\rightarrow}$ computes a future initiation at T_{att} iff there is no cancellation point between $T_{att}{-}R$ and T_{att}.

Correctness of future initiation evaluation after T_{att}. Suppose that $T_1, \ldots, T_k$ are the temporally sorted initiations of $F = V$. We show that RTEC$^{\rightarrow}$ computes a future initiation of $F = V'$ at $T_i{+}R$, where $1 \leq i \leq k$, iff the future initiation of $F = V'$ is not cancelled between T_i and $T_i{+}R$. For the base case, according to lines 5–7 of Algorithm 2 of the paper, RTEC$^{\rightarrow}$ computes a future initiation of $F = V'$ at $T_1{+}R$ iff there is no cancellation point between T_1 and $T_1{+}R$. Note that there are no initiations of $F = V$ before T_1, and thus no future initiations of $F = V'$ before $T_1{+}R$. In other words, the cache has all cancellation points of the future initiation of $F = V'$ between T_1 and $T_1{+}R$. Next, assume that RTEC$^{\rightarrow}$ has evaluated correctly the future initiations of $F = V'$ up to $T_{n-1}{+}R$, where $1 < n \leq k$. Since RTEC$^{\rightarrow}$ caches the future initiations that it derives (line 7 of Algorithm 2), the cache contains all cancellation points of the future initiations of $F = V'$ up to $T_{n-1}{+}R$. As a result, RTEC$^{\rightarrow}$ computes a future initiation of $F = V'$ at $T_n{+}R$ iff there is no cancellation point between T_n and $T_n{+}R$.

Third, we generalise the proof for the case where the vertex of $F = V$ has incoming f-edges, i.e., we may have future initiations of $F = V$. In this case, we may have additional initiations of $F = V$, and thus we have to re-establish the correctness of $IP[F = V]$.

Correctness of $IP[F = V]$ with future initiations of $F = V$. We consider the case where the vertex of $F = V$ has incoming f-edges, i.e., we may have future initiations of $F = V$, and prove that $IP[F = V]$ contains all initiations of $F = V$, and no other points. Suppose that we have n facts $\mathsf{fi}(F = V_i, F = V, R_i)$, where $1 \leq i \leq n$, and all values V_i are pairwise different. Since there are no cycles in the dependency graph, RTEC$^{\rightarrow}$ processes FVPs $F = V_1, \ldots, F = V_n$ before $F = V$ (see Definition 12 of the paper). For each FVP $F = V_i$, if we may not have future initiations of $F = V_i$, then RTEC$^{\rightarrow}$ evaluates the future initiations of $F = V$ based on $\mathsf{fi}(F = V_i, F = V, R_i)$ correctly, as proven above, and stores them in list $IP[F = V]$ (see lines 7 and 9 of Algorithm 1). Otherwise, if there are some fi facts defining $F = V_i$, then, based on the acyclicity of the cd-component, we may use an inductive proof on the vertices of the cd-component, in order to show that the future initiations of $F = V$ based on $\mathsf{fi}(F = V_i, F = V, R_i)$ are derived correctly. As a result, when processing FVP $F = V$, $IP[F = V]$ contains all the future initiations of $F = V$.

Fourth, we generalise the proof for the case where we have a cd-component with cycles. In this case, we show that future initiation evaluation is reduced to initiatedAt/terminatedAt rule evaluations, which are inherited from RTEC, in conjunction with caching intermediate derivations.

Correctness of future initiation evaluation with cycles. Suppose that there is a cycle of f-edges that includes f-edge $(v_{F=V}, v_{F=V'})$. In this case, RTEC$^{\rightarrow}$ does not employ Algorithm 2 of the paper; instead, it uses Algorithm III to evaluate the initiations of all FVPs in the cycle. When evaluating the future initiations of $F = V'$, Algorithm III does not assume that all initiations of $F = V$ and all cancellation points of future initiations of $F = V'$ have been evaluated and cached. To address this issue, Algorithm III follows a direct implemenation of

the definition of fi (see Definition 5 of the paper), according to which any initiation/termination point that is required to evaluate a future initiation of $F = V'$ and has not been evaluated in a previous step, based on the cache, is determined using initiatedAt/terminatedAt rules. As a result, the correctness of Algorithm III follows from the correctness of rule evaluation, which we have assumed to be correct. Therefore, RTEC$^{\rightarrow}$ evaluates correctly future initiations with cycles. ◊

In the proof of Lemma II, we demonstrated that, at the time of evaluating the future initiations of $F = V'$, the cache contains all initations of $F = V$, and no other initiation points, and all terminations of $F = V$, except from those stemming from future initiations of $F = V'$, and no other termination points. Moreover, we proved that RTEC$^{\rightarrow}$ computes the future initiations of $F = V'$, and no other points. According to line 8 of Algorithm 1 of the paper, the derived future initiations of $F = V'$ are added in the list of cached terminations of $F = V$. As a result, at the time of constructing the maximal intervals of $F = V$, the cache contains the initiations and terminations of $F = V$, and no other points, leading to correct maximal interval construction. Therefore, RTEC$^{\rightarrow}$ computes the maximal intervals of all FVPs of an event description, and no other interval.

2.2 Complexity of RTEC$^{\rightarrow}$

Proposition II (Complexity of RTEC$^{\rightarrow}$): The cost of evaluating the future initiations of all FVPs in a cd-component is $\mathcal{O}(n_v(\omega{-}R)log(\omega))$, where n_v is the number of possible values of an FVP, ω is the window size and R is the delay of a future initiation. ▲

First, we demonstrate the complexity of transferring attempt events between windows (see Lemma III). Second, we outline the cost of evaluating the future initiations of all FVPs in an acyclic cd-component (see Lemma IV). Third, we discuss the case of a cd-component with cycles (see Lemma V).

Lemma III (Complexity of transferring attempt events): The cost of transferring attempt events between windows is $\mathcal{O}(\omega)$. ▲

Proof. Consider fi$(F = V, F = V', R)$. RTEC$^{\rightarrow}$ employs Algorithm II to determine which attempt events of initiating $F = V'$ will be cached. In the worst case, Algorithm II iterates over the list of cached initiation points of $F = V$ once. The size of this list is bounded by the window size ω, and thus the worst case complexity of Algorithm II is $\mathcal{O}(\omega)$. RTEC$^{\rightarrow}$ employs Algorithm I to determine which one the cached attempt events, if any, may mark a future initiation of $F = V'$. In the worst case, Algorithm I iterates over the list of cached attempt events once, and the number of these events is bounded by the window size. As a result, the cost of Algorithm I is $\mathcal{O}(\omega)$. Therefore, the cost transferring attempt events between windows is $\mathcal{O}(\omega)$. ◊

Lemma IV (Complexity of processing an acyclic cd-component): The cost of evaluating the future initiations of all FVPs in an acyclic cd-component is $\mathcal{O}(n_v(\omega{-}R)log(\omega))$, where n_v is the number of possible values of an FVP, ω is the window size and R is the delay of a future initiation. ▲

Proof. Consider fi$(F = V, F = V', R)$ and an an acyclic cd-component that contains the vertex of FVP $F = V$. We compute the cost of evaluating the future initiations of $F = V'$. For each initiation point T of $F = V$, where $T \leq q_i{-}R$, we may have a future initiation of $F = V'$ that falls inside the window. The number of these initiations is bounded by $\omega{-}R$. Moreover, there may be a future initiation of $F = V'$ at the time of the cached attempt event, if any. Therefore, RTEC$^{\rightarrow}$ may evaluate a future initiation of $F = V'$ at most $\omega{-}R{+}1$ times. To evaluate each future initiation of $F = V'$, RTEC$^{\rightarrow}$ performs a cache retrieval operation from the sorted lists of initiation and termination points of $F = V$, in order to check whether one of them cancels the future initiation of $F = V'$. The size of each list is bounded by ω, and thus the cost of determining whether a future initiation is cancelled is $\mathcal{O}(log(\omega))$. Therefore, after simplifications, the cost of evaluating the future initiations of $F = V'$ is $\mathcal{O}((\omega{-}R)\log(\omega))$. In an acyclic cd-component, all vertices are connected with f-edges, and thus correspond to FVPs with the same fluent. As a result, an acyclic cd-component may have at most n_v FVPs. Thus, the cost of evaluating the future initiations of all FVPs in a cd-component is $\mathcal{O}(n_v(\omega{-}R)log(\omega))$. ◊

Lemma V (Complexity of processing a cd-component with cycles): The cost of evaluating the future initiations of all FVPs in an cd-component with cycles is $\mathcal{O}(n_v(\omega{-}R)log(\omega))$, where n_v is the number of possible values of an FVP, ω is the window size and R is the delay of a future initiation. ▲

Proof. Consider $\mathsf{fi}(F=V, F=V', R)$ and that the cd-component that includes the vertices of $F=V$ and $F=V'$ contains cycles with f-edges. In this case, RTEC$^{\rightarrow}$ does not assume that the cancellation points of future initiations of $F=V'$ are available in the cache. To address this issue, RTEC$^{\rightarrow}$ employs Algorithm III, which may, at most, evaluate the future initiation of each FVP in the cd-component once, at each of the $\omega{-}R{+}1$ time-points of the window where we may have such an initiation. This is because Algorithm III uses the cache of RTEC$^{\rightarrow}$ in order to store previous evaluation of FVP initiations, and thus avoids re-computations. As a result, the cost of evaluating the future initiations of all FVPs in a cd-component including cycles with f-edges remains $\mathcal{O}((\omega{-}R)n_v log(\omega))$. ◊

Proposition II presents the cost of RTEC$^{\rightarrow}$ in the worst-case, where we have an initiation/termination of an FVP at each time-point of the window. In practice, the number k of initiations and terminations of an FVP is much smaller than the window size ω, resulting in fewer computations than in the worst-case. Given an acyclic cd-component with $\mathsf{fi}(F=V, F=V', R)$, for each initiation T of $F=V$, RTEC$^{\rightarrow}$ evaluates a future initiation of $F=V'$ at $T{+}R$, resulting in k evaluations. In order to decide whether a future initiation of $F=V'$ is cancelled, RTEC$^{\rightarrow}$ examines a cache of at most k points, leading to a cost of $\mathcal{O}(log(k))$. Moreover, the number of cached attempt events is bounded by k, and thus the cost of windowing becomes $\mathcal{O}(k)$. Moreover, the cache includes at most k time-points that may cancel a future initiation of $F=V'$, and thus the cost of a cache retrieval operation becomes $\mathcal{O}(log(k))$. As a result, when $k{\ll}\omega$, the cost of evaluating the future initiations of all FVPs in a cd-component becomes $\mathcal{O}(n_v klog(k))$, which is signicantly smaller compared to the worst-case. If the cd-component has cycles, then we have to evaluate future initiations of $F=V'$ at $\omega{-}R$ time-points, because some initiations of $F=V$ may not be in the cache. Moreover, the cache contains the results of earlier evaluations of future initiations of $F=V'$ at each time-point of the window, and thus the cost of retrieving a point from the cache is $\mathcal{O}(log(\omega))$. Therefore, in the case of cycles with f-edges, the cost of future initiation evaluation remains $\mathcal{O}((\omega{-}R)n_v log(\omega))$, which shows that, in practice, RTEC$^{\rightarrow}$ processes cd-components without cycles more efficiently than cd-components with cycles, thanks to its processing order of FVPs.

Based on Lemmata III, IV and V, the worst-case cost of evaluating the future initiations of all FVPs in a cd-component, including windowing, is $\mathcal{O}(n_v(\omega{-}R)log(\omega))$.

2.3 Complexity of rules (α)–(γ)

Rules (α)–(γ) of the paper can be used to evaluate future initations of FVPs. However, these rules are not suitable for reasoning over data streams, and thus they are not part of RTEC$^{\rightarrow}$. We compare RTEC$^{\rightarrow}$ with rules (α)–(γ), in order to highlight the benefits of the processing order and the caching mechanism of RTEC$^{\rightarrow}$. Below, we outline the worst-case complexity of rules (α)–(γ).

Proposition III: The cost of evaluating future initiations using rules (α)–(γ) of the paper is $\mathcal{O}((n_v R)^{\omega-R})$, where n_v is the number of possible values of an FVP, ω is the window size and R is the delay of a future initiation. ▲

Proof. Suppose that we have $\mathsf{fi}(F=V, F=V', R)$, and the time-points of the window are $T_1, T_2, \ldots, T_\omega$. $\mathsf{fiCost}(T_n)$ denotes the cost of evaluating a future initiation of $F=V'$ at T_n and m denotes the cost of computing whether an FVP is initiated or terminated at some time-point based on $\mathsf{initiatedAt}/\mathsf{terminatedAt}$ rules. Based on rule (α), we have:

$$\mathsf{fiCost}(T_n) = \begin{cases} m{+}\mathsf{fiCost}(T_{n-R}){+}cc(T_{n-R}, T_n) & n > R \\ cc(T_1, T_n) & 1 \leq n \leq R, \mathsf{holdsAt}(F=V, T_1) \textbf{ and } T_n = T_{att} \\ \mathcal{O}(1) & otherwise \end{cases}$$

If T_n is after T_R, then there may be a future initiation of $F = V'$ at T_n which is induced by an initiation of $F = V$ at the time-point T_{n-R} of the current window. The cost of evaluating a future initiation of $F = V'$ at T_n is equal to the cost of evaluating whether $F = V$ is initiated at T_{n-R}, i.e., the cost m of evaluating its initiatedAt rules plus the cost $\mathsf{fiCost}(T_{n-R})$ of evaluating whether there is a future initiation of $F = V$ at T_{n-R}, plus the cost $cc(T_{n-R}, T_n)$ of checking whether the future initiations of $F = V'$ is cancelled between T_{n-R} and T_n. If T_n is not after T_R, then there may be a future initiation of $F = V'$ at T_n only if T_n coincides with the time-stamp T_{att} of the derived attempt event. In this case, the cost of $\mathsf{fiCost}(T_n)$ is equal to the cost of $cc(T_1, T_n)$. Otherwise, $\mathsf{fiCost}(T_n)$ simply returns false in $\mathcal{O}(1)$.

The cost of evaluating $cc(T_{n-R}, T_n)$ is:

$$cc(T_{n-R}, T_n) = \sum_{i=n-R+1}^{n-1} (m + \sum_{V^d \neq V} (m + \mathsf{fiCost}(T_i))) \tag{1}$$

For each time-point T_i between T_{n-R} and T_n, we compute whether $F = V$ is terminated at T_i with cost m and, then, for each value V^d of F other than V, we compute whether $F = V^d$ is initiated at T_i based on initiatedAt rules and whether there is a future initiation of $F = V^d$ at T_i.

Because of the caching and indexing techniques of RTEC$^{\rightarrow}$, the cost m remains close to constant. After substituting the cost of $cc(T_{n-R}, T_n)$, according to equation (1), in the definition of $\mathsf{fiCost}(T_n)$, and making some simplifications, the cost $\mathsf{fiCost}(T_n)$, where $n > R$, is:

$$\mathsf{fiCost}(T_n) = \mathsf{fiCost}(T_{n-R}) + \sum_{i=n-R+1}^{n-1} \sum_{V^d \neq V} \mathsf{fiCost}(T_i)$$

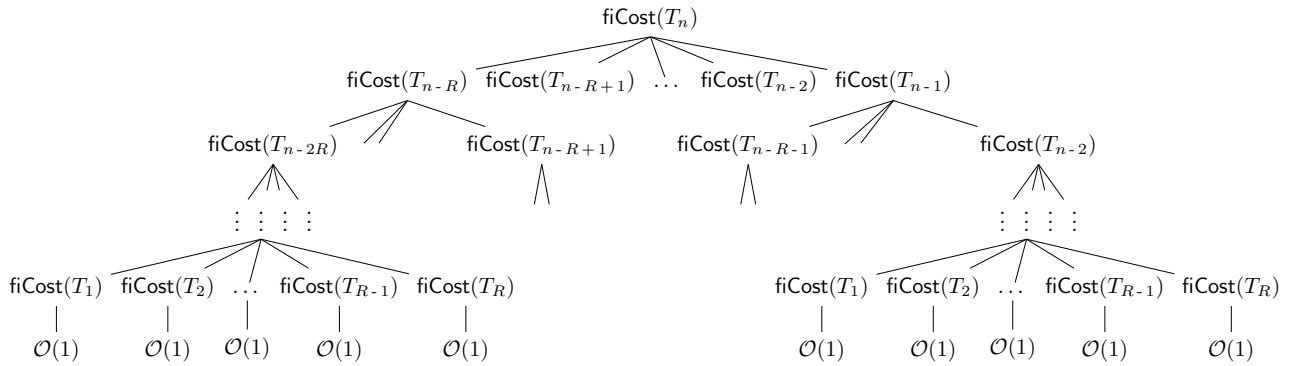

Figure 1: Recursion tree for evaluating a future initiation at time-point T_n, where $n > R$.

Figure 1 shows the recursion tree for $\mathsf{fiCost}(T_n)$, in the simple case where F has two possible values. Suppose that F has n_v possible values. Then, each node $\mathsf{fiCost}(T_n)$, where $n > R$, has $1 + n_v(R - 1)$ children nodes. Moreover, the height of the tree is $\omega - R$. This can be verified in the case of $n = \omega$ by following the rightmost child of each node of the recursion tree, until reaching $\mathsf{fiCost}(T_R)$, which does not have any children. Therefore, after simplifications, the cost of evaluating $\mathsf{fiCost}(T_n)$ with rules (α)–(γ) is $\mathcal{O}((n_v R)^{\omega - R})$. $\Diamond$

3 Future Terminations

RTEC$^{\rightarrow}$ also supports events that may terminate an FVP in the future. We demonstrate that minor extensions to the syntax and the reasoning algorithms of the paper are sufficient to express and reason over such events. The semantics of RTEC$^{\rightarrow}$ are not affected by these extensions; future terminations do not introduce additional dependencies among FVPs, and thus the semantics of RTEC$^{\rightarrow}$ remains locally stratified logic programs.

3.1 Syntax

An event description in RTEC$^{\rightarrow}$ may contain facts expressing future terminations of FVPs.

Definition I (Event Description in RTEC$^{\rightarrow}$): We extend Definition 5 of the paper with the following type of facts:

- ft$(F = V, R)$, where R is a positive integer, expressing that the initiation of $F = V$ at a time-point T leads to the ***f**uture **t**ermination* (ft) of $F = V$ at time-point $T+R$, provided that $F = V$ is not 'broken' between T and $T+R$. ■

ft facts express a different type of delayed effects compared to fi facts. Suppose that $F = V$ is initiated at time-point T, and that the delayed effects of events are not cancelled. According to ft$(F = V, R)$, there is a future termination of $F = V$ at time-point $T+R$, which implies that we become agnostic about the value of F after $T+R$. Contrast this with the effects of an fi fact, e.g., fi$(F = V, F = V', R)$, where $V \neq V'$, according to which there is a future initiation of $F = V'$ at time-point $T+R$, meaning that the value of F is V' after $T+R$.

3.2 Reasoning

Since ft facts do not affect the dependency graph of RTEC$^{\rightarrow}$, we do not need to modify the processing order relation presented in the paper. Below, we show the reasoning algorithms of RTEC$^{\rightarrow}$ that incorporate future terminations. Algorithm IV is a modified version of Algorithms 1 of the paper that processes ft facts. Algorithm V shows the steps of RTEC$^{\rightarrow}$ for computing future terminations.

Algorithm IV processCDComponent

Input: cd-component G_{S_i}, cached intervals I.
Output: I, including the intervals of the FVPs in G_{S_i}.

```
1: if G_{S_i} does not contain a cycle then
2:   for each v_{F=V} in G_{S_i} do IP[F=V] ← [ ]
3:   for each v_{F=V} in processingOrder(G_{S_i}) do
4:     IP[F=V].add(RTEC.evalIP(F=V, I))
5:     TP[F=V]←RTEC.evalTP(F=V, I)
6:     if fi(F=V, F=V', R) then
7:       FtIP[F=V']←evalFI(IP[F=V], TP[F=V], V', R)
8:       TP[F=V].add(FtIP[F=V'])
9:       IP[F=V'].add(FtIP[F=V'])
10:    else if ft(F=V, R) then
11:      TP[F=V].add(evalFT(IP[F=V], TP[F=V], R))
12:    I[F=V]←RTEC.mi(IP[F=V], TP[F=V])
13: else if G_{S_i} does not contain an f-edge then
14:   I.updateIntervals(RTEC_∘(G_{S_i}, I))
15: else I.updateIntervals(cyclicFI(G_{S_i}, I))
16: return I
```

If we have an fi fact and an ft fact with the same FVP $F = V$ as their first argument, then only the one with the shorter delay will be meaningful, while the other fact will not generate any results, regardless of the input stream. Therefore, RTEC$^{\rightarrow}$ considers that $F = V$ may be the first argument of either an fi fact or an ft fact (see lines 6 and 10 of Algorithm IV). When we have ft$(F = V, R)$, RTEC$^{\rightarrow}$ computes the future terminations of $F = V$ and adds them in list $TP[F = V]$, which is maintained temporally sorted (see lines 10–11). The reasoning steps of RTEC$^{\rightarrow}$ for computing the future terminations of $F = V$ (see Algorithm V) are the same as the ones presented in Algorithm 2 of the paper for computing future initiations.

As demonstrated above, RTEC$^{\rightarrow}$ supports future terminations of FVPs with only minimal changes in its reasoning algorithms. The correctness proof of RTEC$^{\rightarrow}$ can be modified to incorporate future terminations with only minor extensions. Moreover, the cost of RTEC$^{\rightarrow}$ does not increase with the introduction of ft facts; the tasks of evaluating fi and ft facts have the same worst-case complexity.

Algorithm V evalFT

Input: $IP[F=V]$, $TP[F=V]$, R.
Global: q_i, ω.
Output: The future terminations of $F=V$.

1: $FtTP[F=V] \leftarrow [\,]$
2: $T_{att} \leftarrow$ findAttempt$(F=V, R)$
3: **if** $T_{att} \neq null$ **and** not cancelled$(q_i - \omega, T_{att}, IP[F=V], TP[F=V])$ **then**
4: $\quad FtTP[F=V]$.add(T_{att})
5: **for each** T **in** $IP[F=V]$ **do**
6: $\quad$ **if** $T+R \leq q_i$ **and** not cancelled$(T, T+R, IP[F=V], TP[F=V] \cup FtTP[F=V])$ **then**
7: $\quad\quad FtTP[F=V]$.add$(T+R)$
8: computeAttempts$(F=V, R)$
9: **return** $FtTP[F=V]$

4 Reproducing our Experiments

The code of RTEC$^{\rightarrow}$ is publicly available[1]. We provide the code for reproducing our experiments[2]. Download and save the code in the directory of your choice. Unzip the archive as follows:

```
cd code/directory
unzip rtec_experiments.zip -d rtec_experiments
cd rtec_experiments
```

The easiest way to reproduce our experiments is through a Docker image. Install Docker[3], and then follow these steps:

```
sudo docker build -t experiments . # Build a Docker image named experiments based on Dockerfile.
sudo docker run -it experiments # Run the Docker image in iteractive mode.
./run_all_experiments.sh # Execute the script the runs all experiments.
```

In order to run a specific set of experiments of the paper, you may run the corresponding execution script as follows:

```
cd scripts
ls # See the available scripts.
./run_script_of_your_choice.sh
```

[1] https://github.com/aartikis/rtec
[2] https://owncloud.skel.iit.demokritos.gr/index.php/s/4cOgYrRIsZ3S0T6
[3] https://docs.docker.com/engine/install/