Some Notes on Realtime Semantics

Himyanshu Anand, John Havlicek, Scott Little

Freescale Semiconductor, Inc

2010-04-13

Abstract Syntax and semantics are proposed for realtime (i.e., continuous time domain) sequences. The new realtime sequence forms include realtime boolean; realtime concatenation; realtime fusion; realtime sequence without an event; boolean smear; and realtime goto. The realtime sequences contain the digital sequences of SVA as a subset (with the exception of local variables and `first_match`). The realtime semantics reduces to the existing SVA semantics on the digital subset when the set of points where clocking events occur has no limit point in $\mathbb{R}$. This subsumption of the existing digital semantics is an important goal of the definitions. It is proved that if a singly clocked digital sequence matches over a bounded interval, then the set of points in the interval at which the clock event occurs is finite. It is proved that for singly clocked digital sequences, the digital fusion and concatenation operators can be obtained in terms of realtime operators. Sanity checks on the semantics are carried out, including proofs of associativity of the new concatenation operators.

1 Introduction

Over a number of years, assertion-based verification has been growing in importance as part of industrial verification methodologies. Part of this growth is evidenced by the standardization of industrially focused assertion languages like SVA and PSL. SVA, in particular, has grown significantly in capabilities through revisions in Accellera and IEEE. The reckoning of time in SVA is currently based upon discrete clocking events. The use of clocking events to define units of discrete time works well for the majority of digital circuit verification, although complex timing properties can be challenging to write using the clock-based assertions. The notion of clocking events does not apply as readily to analog/mixed-signal (AMS) circuits. For the expression of many AMS properties, a first class notion of continuous time is needed in the assertion language.

There has been previous work on realtime (i.e., continuous time domain) extensions to LTL operators that provides a clear roadmap for extending the analogous features of SVA [1]. This document focuses on defining realtime extensions to SVA sequences, for which we know of no research defining a suitable framework. An important characteristic of the definition is that it indeed be an extension of the current SVA sequences. This entails providing a realtime semantics for the clocked digital sequences that is equivalent, through a suitable notion of sampling, to the existing discrete semantics. The realtime semantics for the digital sequences was not immediately obvious to us and required a number of iterations before we discovered the definitions presented here. A second characteristic of the definition of realtime sequences is that it contains a small number of additional operators with which we believe that the preponderance of realtime sequences of practical interest can be defined. We are not in a position today to make a strong case to this effect, but we have studied a number of examples from willing contributors that lead us to this conjecture.

The new realtime sequence forms include realtime boolean; realtime concatenation; realtime fusion; realtime sequence without an event; boolean smear; and realtime goto. Of these new operators, realtime concatenation and reatltime goto can be derived from the other operators.

The semantics of realtime sequences is given by a matching relation that defines when a realtime sequence matches over a bounded realtime interval of a realtime trace. The interval can be empty, open, closed, or half-open.

Intuitive descriptions of the realtime sequence operators are given below. In the descriptions below b is a boolean expression, c is a clocking event expression, and R is a realtime sequence.

- The realtime boolean (b) is satisfied on an interval that is a single point provided the boolean expression is true at that point.
- A sequence without an event (R `without @(`c`)`) is satisfied over an interval where the sequence is satisfied and the event does not occur.
- The boolean smear (b`[*`α`[+]:`β`[-]]`) is satisfied when a boolean expression is true throughout the real interval.
- The realtime fusion (`#0`) operator matches over the concatenation of a pair of intervals over which its operand sequences match. Two non-empty intervals can be concatenated if they abut at a given point and the point is in at least one of the intervals.[1] In general, two intervals can be concatenated if their union is an interval and their intersection is at most a single point.
- The concatenation with real delay operator (R `#[`α`[+]:`β`[-]]` R') allows two realtime sequences to be connected by a specified time range. This operator is derived in terms of realtime fusion and boolean smear.
- The realtime goto operator (b`[->1]`) carries sequence evaluation to the nearest concurrent or future point in time where its boolean operand is true. The point is not aligned with a clock but is simply the nearest time when the boolean is true. This operator is derived in terms of realtime boolean, realtime fusion, and boolean smear.

To help further illustrate the meanings of these operators several examples of realtime sequences are provided below.

- a is true and b is false continuously for 10.5 time units.

  ```
  (a && !b) [*10.5]
  ```

- a is true and 9.7 time units later c is true.

  ```
  a #9.7 c
  ```

- From the beginning of the interval, advance to the first time where a is high, then find b and c high 1.6 time units later, and also ensure that b subsequently stays high continuously for 5.1 time units.

  ```
  a[->1] #1.6 (b && c) #0 b[*5.1]
  ```

[1] "Abut at a given point" means that the point is the supremum of one of the intervals and the infimum of the other.

- The digital SVA semantics is embedded in the realtime semantics. For instance, if R and R' are digital sequences singly clocked by c and neither admits empty match, then R `##1` R' is equivalent in the realtime semantics to R `#0 (1[*0.0+:$] without @(`c`)) #0` R'.

We intend to consider the addition of local variables and `first_match` to this framework in future work.

2 Preliminaries and Notation

$\mathbb{R}$ denotes the set of real numbers. $\mathbb{B}$ denotes the set $\{0, 1\}$ of boolean values. $\mathbb{N}$ denotes the set of non-negative integers.

Let A and B be finite sets. A will be understood as the set of *analog variables*, and B will be understood as the set of *boolean variables*. A *state* (*of the variables*) is an assignment of a real number to each analog variable and a boolean value to each boolean variable. A state may be identified with an element of the set

$$\Sigma = \mathbb{R}^A \times \mathbb{B}^B$$

Σ may be considered as the set of states of the variables. We may think of Σ as the set of pairs (s_A, s_B), where s_A is a function $A \to \mathbb{R}$ (the analog state) and s_B is a function $B \to \mathbb{B}$ (the boolean state). This amounts to thinking of $\mathbb{R}^A$ as the set of functions $A \to \mathbb{R}$ and of $\mathbb{B}^B$ as the set of functions $B \to \mathbb{B}$ in the usual way.

Let $s \in \Sigma$ and $a \in A$ and $b \in B$. By $s(a)$, or equivalently $a[s]$, we denote the element of $\mathbb{R}$ assigned to a in state s, and by $s(b)$, or equivalently $b[s]$, we denote the element of $\mathbb{B}$ assigned to b in state s. In other words, if we identify s with the pair (s_A, s_B) as above, then for $a \in A$, $s(a) = s_A(a)$, and for $b \in B$, $s(b) = s_B(b)$.

A *boolean expression* (*over the variables*) assigns to each state of the variables an element in $\{0, 1\}$. A boolean expression may be identified with an element of the set

$$\mathbb{B}^\Sigma = \mathbb{B}^{\mathbb{R}^A \times \mathbb{B}^B}$$

$\mathbb{B}^\Sigma$ may be considered as the set of boolean expressions over the variables. We may think of $\mathbb{B}^\Sigma$ as the set of functions $\Sigma \to \mathbb{B}$ in the usual way, so that if e is a boolean expression and s is a state, then $e(s) \in \mathbb{B}$. We write $s \models e$ iff $e(s) = 1$. For example, let $a, a' \in A$ and let $b, b' \in B$. Then the following define boolean expressions:

1. a `>` a'
2. `(`a `>` a'`) ||` b
3. `(`a `>` a'`) || (`b `&& !`b'`)`

Using the notations above, for $s \in \Sigma$,

$$(a \texttt{ > } a')(s) = (s(a) > s(a')) = (a[s] > a'[s])$$

Similarly,

$$((a \texttt{ > } a') \texttt{ || } b)(s) = ((s(a) > s(a')) \vee s(b)) = ((a[s] > a'[s]) \vee b[s])$$

A *continuous trace* is a function $W : \mathbb{R}_{\geq 0} \to \Sigma$. Typically, W will be assumed to satisfy regularity criteria. For example, to simplify the description of sampling lefthand limits, W might be assumed to be continuous from the left or continuous from the left at all candidate sampling points. A *discrete trace* is a function $w : \mathbb{N} \to \Sigma$. A *sampling* is a strictly increasing function $T : \mathbb{N} \to \mathbb{R}_{\geq 0}$ such that $\lim_{n\to\infty} T(n) = \infty$.

Given a continuous trace W and a sampling T, $W \circ T$ is a discrete trace, where by $W \circ T$, we mean ordinary composition of the function T followed by the function W. Explicitly, $W \circ T$ is the function $\mathbb{N} \to \Sigma$ defined by

$$n \mapsto W(T(n))$$

We say that the continuous trace W is *consistent* with a discrete trace w and a sampling T iff $w = W \circ T$.

Given a boolean expression c and a continuous trace W, the set

$$\{t \in \mathbb{R} : W(t) \models c\}$$

can be thought of as the set of times at which the event "c is true" occurs in the trace W. In this way, we can think of a boolean expression as an event that occurs at the times t in this set.

A sketch of the definition of realtime semantics for regular expressions appears below. Here we give an example from that sketch. Let c, e be boolean expressions and let $0 \leq t < t'$.

$W, [t, t'] \models$ `@`(c)`(`e`)`
iff 1. $[t, t']$ is right closed (which is true since $t < t'$)
 2. $W(t') \models e$
 3. $\{x \in [t, t'] : W(x) \models c\} = \{t'\}$
iff 1. $W(t') \models e$ `&&` c
 2. $\forall x$ such that $t \leq x < t' : W(x) \not\models c$

The fundamental goal of these notes is that the realtime semantics of digital sequences be consistent with the discrete sampled semantics of the same sequences as already standardized in SVA. The primary result to this effect is the following:

Proposition 0: *Let R be a digital sequence singly clocked by c. Let T be a sampling such that $T(\mathbb{N})$ contains the set $\{t \in \mathbb{R} : W(t) \models c\}$. Let $w = W \circ T$. Let I be a bounded interval. If I is empty, let v be the empty word. Otherwise, assume that I is right-closed with* $\sup I \in T(\mathbb{N})$*, and define $v = w^{i..j}$, where i is the minimum and j the maximum natural numbers that are mapped by T into I. Then $W, I \models R$ in the realtime semantics iff $v \models R$ in the SVA semantics.*

Throughout, α, β denote real constants, and m, n denote integer constants.

3 Intervals

Throughout, I,J,K,L (and with primes and subscripts) denote *bounded* intervals in the real line $\mathbb{R}$. Below, we will define sequences R and the relation $W, I \models R$ for general bounded I (including I empty, open, closed, half-open).

Definition 1:

(a) $I \le I'$ *iff* $\forall t \in I\ \forall t' \in I' : t \le t'$.
(b) $I < I'$ *iff* $\forall t \in I\ \forall t' \in I' : t < t'$.

□

Claim 2: *$I \le I'$ iff either*

(a) *I is empty, or*
(b) *I' is empty, or*
(c) *Both I and I' are not empty and* $\sup I \le \inf I'$.

Proof: If I is empty, then (a) above holds and the RHS of Definition 1(a) holds vacuously (there is no $t \in I$). Similarly, if I' is empty, then (b) above holds and the RHS of Definition 1(a) holds vacuously (there is no $t' \in I'$).

Suppose now that neither I nor I' is empty. Assume that $I \le I'$. Then each element of I is a lower bound for I'. Since $\inf I'$ is the greatest lower bound for I', it follows that each element of I is $\le \inf I'$. Therefore $\inf I'$ is an upper bound for I. Since $\sup I$ is the least upper bound for I, it follows that $\sup I \le \inf I'$. This establishes (c). Assume now that (c) holds, so that $\sup I \le \inf I'$. Let $t \in I$ and $t' \in I'$. Then $t \le \sup I \le \inf I' \le t'$. Since t, t' are arbitrary, this shows that the RHS of Definition 1(a) holds and so proves that $I \le I'$. □

Claim 3:

(a) *$I \le J \cup K$ iff $I \le J$ and $I \le K$.*
(b) *$J \cup K \le I$ iff $J \le I$ and $K \le I$.*

Proof:

(a) $I \le J \cup K$
iff $\forall t \in I\ \forall t' \in J \cup K : t \le t'$
iff $\forall t \in I : (\forall t' \in J : t \le t')$ and $(\forall t' \in K : t \le t')$
iff $(\forall t \in I\ \forall t' \in J : t \le t')$ and $(\forall t \in I\ \forall t' \in K : t \le t')$
iff $I \le J$ and $I \le K$.
(b) The proof is similar.

□

If I is non-empty, then we write

$$\mathit{length}(I) = \sup I - \inf I$$

We write

$$\mathit{length}(\{\}) = 0$$

4 Digital Sequences

This grammar makes clocks and their scoping explicit by requiring that clocks be attached to booleans. This restriction is to simplify reasoning about the grammar. In an extension to SVA, the clock flow rules would need to be generalized to define the scoping of clocks. It should always be possible to transform a sequence using clock scoping rules into an equivalent one in which clocks are attched only to booleans.

$$
\begin{aligned}
R ::= {} & \texttt{@(}c\texttt{)(}b\texttt{)} \\
& \mid R \;\texttt{\#\#1}\; R \\
& \mid R \;\texttt{\#\#0}\; R \\
& \mid R \;\texttt{or}\; R \\
& \mid R \;\texttt{intersect}\; R \\
& \mid R\texttt{[*0]} \\
& \mid R\texttt{[*1:\$]}
\end{aligned}
$$

Definition 4 (Digital Sequence Semantics):

- $W, I \models \texttt{@(}c\texttt{)(}b\texttt{)}$
  iff 1. I *is non-empty and* $\sup I \in I$ (*i.e.,* I *is right closed*)
  2. $W(\sup I) \models b$
  3. $\{t \in I : W(t) \models c\} = \{\sup I\}$

- $W, I \models R \;\texttt{\#\#1}\; R'$
  iff $\exists J, J'$:
  1. $I = J \cup J'$
  2. $J < J'$
  3. $W, J \models R$ *and* $W, J' \models R'$

- $W, I \models R \;\texttt{\#\#0}\; R'$
  iff $\exists J, J', a$:
  1. $I = J \cup J'$ *and* $J \cap J' = [a, a]$
  2. $J \leq [a, a]$
  3. $[a, a] \leq J'$
  4. $W, J \models R$ *and* $W, J' \models R'$

- $W, I \models R \;\texttt{or}\; R'$
  iff either
  1. $W, I \models R$
  or
  2. $W, I \models R'$

- $W, I \models R \;\texttt{intersect}\; R'$
  iff both
  1. $W, I \models R$
  and
  2. $W, I \models R'$

- $W, I \models R\texttt{[*0]}$
  iff I *is empty*

- $W, I \models R$`[*1:$]`
  iff $\exists n \geq 1$ *and* $J_1, ..., J_n$ *such that*
  1. $\forall 1 \leq i < j \leq n : J_i < J_j$
  2. $I = J_1 \cup ... \cup J_n$
  3. $\forall 1 \leq i \leq n : W, J_i \models R$

□

These definitions of the semantics of the digital sequence operators will apply when those operators are used in general realtime sequences.

Here is an example to test the semantics of changing clocks after `##0`:

$W, I \models$ `@(`b`)(1) ##0 @(`c`)(1)`
iff $\exists J, J', a$:
1. $I = J \cup J'$ and $J \cap J' = [a, a]$
2. $J \leq [a, a]$
3. $[a, a] \leq J'$
4. $W, J \models$ `@(`b`)(1)` and $W, J' \models$ `@(`c`)(1)`

iff $\exists J, J', a$:
1. $I = J \cup J'$ and $J \cap J' = [a, a]$
2. $J \leq [a, a]$
3. $[a, a] \leq J'$
4. a. J and J' are right closed
   b. $\{t \in J : W(t) \models b\} = \{\sup J\} = \{a\}$
   c. $\{t \in J : W(t) \models c\} = \{\sup J'\}$

Definition 5 (Set of terminal clocks):

- *term_clocks*(`@(`c`)(`b`)`) $= \{c\}$
- *If* R_2 *admits empty match, then*

  term_clocks(R_1 `##1` R_2) = *term_clocks*(R_1) ∪ *term_clocks*(R_2)

  Otherwise

  term_clocks(R_1 `##1` R_2) = *term_clocks*(R_2)
- *term_clocks*(R_1 `##0` R_2) = *term_clocks*(R_2)
- *term_clocks*(R_1 `or` R_2) = *term_clocks*(R_1) ∪ *term_clocks*(R_2)
- *term_clocks*(R_1 `intersect` R_2) = *term_clocks*(R_1) ∪ *term_clocks*(R_2)
- *term_clocks*(R`[*0]`) = {}
- *term_clocks*(R`[*1:$]`) = *term_clocks*(R)

□

Proposition 6: *Let S be a digital sequence. If $W, I \models S$ and I is non-empty, then I is right closed and at least one of the terminal clocks of S is satisfied in W at the right endpoint of I.*

Proof: By induction over the structure of S. Assume that $W, I \models S$ and I is non-empty.

- $S = \texttt{@(}c\texttt{)(}b\texttt{)}$. $\mathit{term\_clocks}(S) = \{c\}$.

  $W, I \models \texttt{@(}c\texttt{)(}b\texttt{)}$
  iff 1. I is right closed
  2. $W(\sup I) \models b$
  3. $\{t \in I : W(t) \models c\} = \{\sup I\}$
  $\Rightarrow$ I is right closed and $W(\sup I) \models c$

- $S = R$ `##1` R'.

  $W, I \models R$ `##1` R'
  iff $\exists J, J'$:
  1. $I = J \cup J'$
  2. $J < J'$
  3. $W, J \models R$ and $W, J' \models R'$

  Suppose J' is empty. Then R' admits empty match, so $\mathit{term\_clocks}(R) \subseteq \mathit{term\_clocks}(S)$. Also, $I = J$, so J must be non-empty. By induction, J is right closed and at least one of the clocks in $\mathit{term\_clocks}(R)$ is satisfied at the right endpoint of J.

  Suppose now that J' is non-empty. R' may or may not admit empty match, so we can say that $\mathit{term\_clocks}(R') \subseteq \mathit{term\_clocks}(S)$. By induction, J' is right closed and a clock in $\mathit{term\_clocks}(R')$ is satisfied at the right endpoint of J'. Since $J < J'$, I must also be right closed, and the right endpoint of I is equal to the right endpoint of J'.

- $S = R$ `##0` R'. $\mathit{term\_clocks}(S) = \mathit{term\_clocks}(R')$.

  $W, I \models R$ `##0` R'
  iff $\exists J, J', a$:
  1. $I = J \cup J'$ and $J \cap J' = [a, a]$
  2. $J \leq [a, a]$
  3. $[a, a] \leq J'$
  4. $W, J \models R$ and $W, J' \models R'$

  Since $[a, a] \subseteq J'$, J' is non-empty. By induction, J' is right closed and a clock in $\mathit{term\_clocks}(R')$ is satisfied at the right endpoint of J'. Since $J \leq [a, a]$ and $[a, a] \leq J'$, I must also be right closed, and the right endpoint of I is equal to the right endpoint of J'.

- $S = R$ `or` R'. $\mathit{term\_clocks}(S) = \mathit{term\_clocks}(R) \cup \mathit{term\_clocks}(R')$.

  $W, I \models R$ `or` R'
  iff either
  1. $W, I \models R$
  or
  2. $W, I \models R'$

  WLOG, $W, I \models R$. By induction, I is right closed and a clock in $\mathit{term\_clocks}(R) \subseteq \mathit{term\_clocks}(S)$ is satisfied at the right endpoint of I.

- $S = R$ `intersect` R'. $\mathit{term\_clocks}(S) = \mathit{term\_clocks}(R) \cup \mathit{term\_clocks}(R')$.

  $W, I \models R$ `intersect` R'
  iff both
  1. $W, I \models R$
  and
  2. $W, I \models R'$

  By induction, I is right closed and a clock in $\mathit{term\_clocks}(R) \subseteq \mathit{term\_clocks}(S)$ is satisfied at the right endpoint of I.

- $S = R$`[*0]`.

$W, I \models R$`[*0]`
iff I is empty

But I is non-empty.

- $S = R$`[*1:$]`. $\mathit{term\_clocks}(S) = \mathit{term\_clocks}(R)$.

  $W, I \models R$`[*1:$]`
  iff $\exists n \geq 1$ and $J_1, ..., J_n$ such that
  1. $\forall 1 \leq i < j \leq n : J_i < J_j$
  2. $I = J_1 \cup ... \cup J_n$
  3. $\forall 1 \leq i \leq n : W, J_i \models R$

  Since I is non-empty, there exists $1 \leq m \leq n$ such that J_m is non-empty. WLOG, m is the maximum such index, so that for all $m < j \leq n$, J_j is empty. Therefore, $I = J_1 \cup ... \cup J_m$. By induction, J_m is right closed and a clock in $\mathit{term\_clocks}(R)$ is satisfied at the right endpoint of J_m. Since $I = J_1 \cup ... \cup J_m$ and since $J_i < J_j$ for all $1 \leq i < j \leq n$, I is right closed and the right endpoint of I is equal to the right endpoint of J_m.

□

Note: The theory of *term_clocks* can be strengthened by defining tuples of clocks, where `intersect` generates the set of pairs of clocks from its operands. The concept of a tuple of clocks being satisfied is that all the clocks in the tuple are satisfied. □

Lemma 7: *Let R be a digital sequence. Let L be a strict prefix of I such that no clock of R holds in W at any point of L. Then $W, I \models R$ iff $W, I - L \models R$.*

Proof: By induction.

- $R =$ `@(`c`)(`b`)`.

  $W, I \models$ `@(`c`)(`b`)`
  iff 1. I is right closed
  2. $W(\sup I) \models b$
  3. $\{t \in I : W(t) \models c\} = \{\sup I\}$

  iff [
  L is a strict prefix of I.
  For ($\Leftarrow$), c does not hold in W at any point of L.
  ]
  1. $I - L$ is right closed
  2. $W(\sup(I - L)) \models b$
  3. $\{t \in I - L : W(t) \models c\} = \{\sup(I - L)\}$

  iff $W, I - L \models$ `@(`c`)(`b`)`

- $R =$ `(`S `##1` S'`)`. Suppose first that $W, I \models S$ `##1` S'. Then $\exists J, J'$:

  1. $I = J \cup J'$
  2. $J < J'$
  3. $W, J \models S$ and $W, J' \models S'$

  - Case: J empty. Then $J' = I$ and S admits empty match.
    $W, I \models S'$
    iff [induction]
    $W, I - L \models S'$
    $\Rightarrow$ [S admits empty match]
    $W, I - L \models S$ `##1` S'

– Case: J non-empty. By Proposition 6, J is right closed and a clock of S is satisfied at its right endpoint. Therefore, L is a strict prefix of J. By induction, $W, J - L \models S$, and so $W, I - L \models S$ `##1` S'.

Suppose now that $W, I - L \models S$ `##1` S'. Then $\exists K, K'$:

1. $I - L = K \cup K'$
2. $K < K'$
3. $W, K \models S$ and $W, K' \models S'$

– Case: K empty. Then $K' = I - L$ and S admits empty match.

$W, I - L \models S'$
iff [induction]
$W, I \models S'$
$\Rightarrow$ [S admits empty match]
$W, I \models S$ `##1` S'

– Case: K non-empty. Then L is a strict prefix of $L \cup K$. By induction, $W, L \cup K \models S$, and so $W, I \models S$ `##1` S'.

- $R = S$ `##0` S'. Suppose first that $W, I \models S$ `##0` S'. Then $\exists J, J', a$:

  1. $I = J \cup J'$ and $J \cap J' = [a, a]$
  2. $J \leq [a, a]$
  3. $[a, a] \leq J'$
  4. $W, J \models S$ and $W, J' \models S'$

  Then J is non-empty, so, by Proposition 6, a clock of S is satisfied at a. Therefore L is a strict prefix of J. By induction, $W, J - L \models S$. Then $W, I - L \models S$ `##0` S'.
  Suppose now that $W, I - L \models S$ `##0` S'. Then $\exists K, K', a$:

  1. $I - L = K \cup K'$ and $K \cap K' = [a, a]$
  2. $K \leq [a, a]$
  3. $[a, a] \leq K'$
  4. $W, K \models S$ and $W, K' \models S'$

  Then K is non-empty, so L is a strict prefix of $L \cup K$. By induction, $W, L \cup K \models S$, and so $W, I \models S$ `##0` S'.

- $R = S$ `or` S'.

  $W, I \models S$ `or` S'
  iff $W, I \models S$ or $W, I \models S'$
  iff [induction]
  $W, I - L \models S$ or $W, I - L \models S'$
  iff $W, I - L \models S$ `or` S'

- $R = S$ `intersect` S'.

  $W, I \models S$ `intersect` S'
  iff $W, I \models S$ and $W, I \models S'$
  iff [induction]
  $W, I - L \models S$ and $W, I - L \models S'$
  iff $W, I - L \models S$ `intersect` S'

- $R = S$`[*0]`. Since L is a strict prefix of I, I and $I - L$ are both non-empty. Therefore, $W, I \not\models S$`[*0]` and $W, I - L \not\models S$`[*0]`.
- $R = S$`[*1:$]`.
  Suppose first that $W, I \models S$`[*1:$]`. Then $\exists n \geq 1$ and $J_1, ..., J_n$ such that

  1. $\forall 1 \leq i < j \leq n : J_i < J_j$
  2. $I = J_1 \cup ... \cup J_n$
  3. $\forall 1 \leq i \leq n : W, J_i \models S$

Since I is non-empty, WLOG all the J_i are non-empty. Since $W, J_1 \models\!\equiv S$, Proposition 6 implies that J_1 is right closed and a clock of S is satisfied at its right endpoint. Therefore, L is a strict prefix of J_1. By induction, $W, J_1 - L \models\!\equiv S$, and so $W, I - L \models\!\equiv S$`[*1:$]`.

Suppose now that $W, I - L \models\!\equiv S$`[*1:$]`. Then $\exists n \geq 1$ and $K_1, ..., K_n$ such that

1. $\forall 1 \leq i < j \leq n : K_i < K_j$
2. $I - L = K_1 \cup ... \cup K_n$
3. $\forall 1 \leq i \leq n : W, K_i \models\!\equiv S$

Since $I - L$ is non-empty, WLOG all the K_i are non-empty. Then L is a strict prefix of $L \cup K_1$. By induction, $W, L \cup K_1 \models\!\equiv S$. Then $W, I \models\!\equiv S$`[*1:$]`.

□

Definition 8 (Set of leading clocks):

- *lead_clocks*(`@`(c)`(`b`)`) $= \{c\}$
- *If* R_1 *admits empty match, then*

$$lead\_clocks(R_1 \texttt{ ##1 } R_2) = lead\_clocks(R_1) \cup lead\_clocks(R_2)$$

  Otherwise

$$lead\_clocks(R_1 \texttt{ ##1 } R_2) = lead\_clocks(R_1)$$

- $lead\_clocks(R_1 \texttt{ ##0 } R_2) = lead\_clocks(R_1)$
- $lead\_clocks(R_1 \texttt{ or } R_2) = lead\_clocks(R_1) \cup lead\_clocks(R_2)$
- $lead\_clocks(R_1 \texttt{ intersect } R_2) = lead\_clocks(R_1) \cup lead\_clocks(R_2)$
- $lead\_clocks(R\texttt{[*0]}) = \{\}$
- $lead\_clocks(R\texttt{[*1:\$]}) = lead\_clocks(R)$

□

Proposition 9: *Let S be a digital sequence. If $W, I \models\!\equiv S$ and I is non-empty, then there exists a point of I at which a leading clock of S is satisfied in W.*

Proof: By induction.

- $S =$ `@`(c)`(`b`)`. $lead\_clocks(S) = \{c\}$.

  $W, I \models\!\equiv$ `@`(c)`(`b`)`
  $\Rightarrow$ I is right closed and $W(\sup I) \models c$

- $S = R \texttt{ ##1 } R'$.

  $W, I \models\!\equiv R \texttt{ ##1 } R'$
  iff $\exists J, J'$:
  1. $I = J \cup J'$
  2. $J < J'$
  3. $W, J \models\!\equiv R$ and $W, J' \models\!\equiv R'$

Suppose J is empty. Then R admits empty match, so $lead\_clocks(S) = lead\_clocks(R) \cup lead\_clocks(R')$. Also, $I = J'$, so J' must be non-empty. By induction there exists a point of J' at which a leading clock of R' is satisfied in W.

Suppose now that J is non-empty. Then $lead\_clocks(S) \supseteq lead\_clocks(R)$. By induction, there exists a point of J at which a leading clock of R is satisfied in W.

- $S = R$ `##0` R'. $lead\_clocks(S) = lead\_clocks(R)$.

  $W, I \models R$ `##0` R'
  iff $\exists J, J', a$:
  1. $I = J \cup J'$ and $J \cap J' = [a, a]$
  2. $J \leq [a, a]$
  3. $[a, a] \leq J'$
  4. $W, J \models R$ and $W, J' \models R'$

  Since $[a, a] \subseteq J$, J is non-empty. By induction, there exists a point of J at which a leading clock of R is satisfied in W.

- $S = R$ `or` R'. $lead\_clocks(S) = lead\_clocks(R) \cup lead\_clocks(R')$.

  $W, I \models R$ `or` R'
  iff either
  1. $W, I \models R$

  or

  2. $W, I \models R'$

  WLOG, $W, I \models R$. By induction, there exists a point of I at which a leading clock of R is satisfied in W.

- $S = R$ `intersect` R'. $lead\_clocks(S) = lead\_clocks(R) \cup lead\_clocks(R')$.

  $W, I \models R$ `intersect` R'
  iff both
  1. $W, I \models R$

  and

  2. $W, I \models R'$

  By induction, there exists a point of I at which a leading clock of R is satisfied in W.

- $S = R$`[*0]`.

  $W, I \models R$`[*0]`
  iff I is empty

  But I is non-empty.

- $S = R$`[*1:$]`. $lead\_clocks(S) = lead\_clocks(R)$.

  $W, I \models R$`[*1:$]`
  iff $\exists n \geq 1$ and $J_1, ..., J_n$ such that
  1. $\forall 1 \leq i < j \leq n : J_i < J_j$
  2. $I = J_1 \cup ... \cup J_n$
  3. $\forall 1 \leq i \leq n : W, J_i \models R$

  Since I is non-empty, WLOG all J_i are non-empty. By induction, there exists a point in J_1 at which a leading clock of R is satisfied in W.

□

Lemma 10: *Let R be a digital sequence. Let L be a strict prefix of I such that no leading clock of R holds in W at any point of L. Then $W, I \models R$ iff $W, I - L \models R$.*

Proof: By induction.

- $R =$ `@(`c`)(`b`)`. c is the leading clock of R.

  $W, I \models$ `@(`c`)(`b`)`
  iff 1. I is right closed
  2. $W(\sup I) \models b$
  3. $\{t \in I : W(t) \models c\} = \{\sup I\}$
  iff [
  L is a strict prefix of I.
  For ($\Leftarrow$), c does not hold in W at any point of L
  ]
  1. $I - L$ is right closed
  2. $W(\sup(I - L)) \models b$
  3. $\{t \in I - L : W(t) \models c\} = \{\sup(I - L)\}$
  iff $W, I - L \models$ `@(`c`)(`b`)`

- $R = S$ `##1` S'. Suppose first that $W, I \models S$ `##1` S'. Then $\exists J, J'$:

  1. $I = J \cup J'$
  2. $J < J'$
  3. $W, J \models S$ and $W, J' \models S'$

  – Case: J empty. Then $J' = I$ and S admits empty match. Therefore, $lead\_clocks(R) = lead\_clocks(S) \cup lead\_clocks(S')$.
  $W, I \models S'$
  iff [induction]
  $W, I - L \models S'$
  $\Rightarrow$ [S admits empty match]
  $W, I - L \models S$ `##1` S'

  – Case: J non-empty. $lead\_clocks(R) \supseteq lead\_clocks(S)$. By Proposition 9, there exists a point of J at which a leading clock of S is satisfied in W. Therefore, L is a strict prefix of J. By induction, $W, J - L \models S$, hence $W, I - L \models S$ `##1` S'.

  Suppose now that $W, I - L \models S$ `##1` S'. Then $\exists K, K'$:

  1. $I - L = K \cup K'$
  2. $K < K'$
  3. $W, K \models S$ and $W, K' \models S'$

  – Case: K empty. Then $K' = I - L$ and S admits empty match. Therefore, $lead\_clocks(R) = lead\_clocks(S) \cup lead\_clocks(S')$.
  $W, I - L \models S'$
  iff [induction]
  $W, I \models S'$
  $\Rightarrow$ [S admits empty match]
  $W, I \models S$ `##1` S'

  – Case: K non-empty. L is a strict prefix of $L \cup K$. $lead\_clocks(R) \supseteq lead\_clocks(S)$. By induction, $W, L \cup K \models S$, and so $W, I \models S$ `##1` S'.

- $R = S$ `##0` S'. $lead\_clocks(R) = lead\_clocks(S)$.
  Suppose first that $W, I \models S$ `##0` S'. Then $\exists J, J', a$:

  1. $I = J \cup J'$ and $J \cap J' = [a, a]$
  2. $J \leq [a, a]$
  3. $[a, a] \leq J'$
  4. $W, J \models S$ and $W, J' \models S'$

Then J is non-empty, so by Proposition 9, there exists a point of J at which a leading clock of S is satisfied in W. Therefore L is a strict prefix of J. By induction, $W, J - L \models S$. Then $W, I - L \models S$ `##0` S'.

Suppose now that $W, I - L \models S$ `##0` S'. Then $\exists K, K', a$:

1. $I - L = K \cup K'$ and $K \cap K' = [a, a]$
2. $K \leq [a, a]$
3. $[a, a] \leq K'$
4. $W, K \models S$ and $W, K' \models S'$

Then K is non-empty, so L is a strict prefix of $L \cup K$. By induction, $W, L \cup K \models S$, and so $W, I \models S$ `##0` S'.

- $R = S$ `or` S'. $lead\_clocks(R) = lead\_clocks(S) \cup lead\_clocks(S')$.

  $W, I \models S$ `or` S'
  iff $W, I \models S$ or $W, I \models S'$
  iff [induction]
  $W, I - L \models S$ or $W, I - L \models S'$
  iff $W, I - L \models S$ `or` S'

- $R = S$ `intersect` S'. $lead\_clocks(R) = lead\_clocks(S) \cup lead\_clocks(S')$.

  $W, I \models S$ `intersect` S'
  iff $W, I \models S$ and $W, I \models S'$
  iff [induction]
  $W, I - L \models S$ and $W, I - L \models S'$
  iff $W, I - L \models S$ `intersect` S'

- $R = S$`[*0]`. Since L is a strict prefix of I, I and $I - L$ are both non-empty. Therefore, $W, I \not\models S$`[*0]` and $W, I - L \not\models S$`[*0]`.

- $R = S$`[*1:$]`. $lead\_clocks(R) = lead\_clocks(S)$.
  Suppose first that $W, I \models S$`[*1:$]`. Then $\exists n \geq 1$ and $J_1, ..., J_n$ such that

  1. $\forall 1 \leq i < j \leq n : J_i < J_j$
  2. $I = J_1 \cup ... \cup J_n$
  3. $\forall 1 \leq i \leq n : W, J_i \models S$

  Since I is non-empty, WLOG all the J_i are non-empty. Since $W, J_1 \models S$, by Proposition 9 there is a point in J_1 at which a leading clock of S is satisfied in W. Therefore, L is a strict prefix of J_1. By induction, $W, J_1 - L \models S$, and so $W, I - L \models S$`[*1:$]`.

  Suppose now that $W, I - L \models S$`[*1:$]`. Then $\exists n \geq 1$ and $K_1, ..., K_n$ such that

  1. $\forall 1 \leq i < j \leq n : K_i < K_j$
  2. $I - L = K_1 \cup ... \cup K_n$
  3. $\forall 1 \leq i \leq n : W, K_i \models S$

  Since $I - L$ is non-empty, WLOG all the K_i are non-empty. Then L is a strict prefix of $L \cup K_1$. By induction, $W, L \cup K_1 \models S$. Then $W, I \models S$`[*1:$]`.

$\square$

Claim 11: *Let R be a digital sequence singly clocked by c. Let*

$$I(W, c) = \{t \in I : W(t) \models c\}$$

If $W, I \models R$, then $I(W, c)$ is finite.

Proof: By induction.

- $R =$ `@(`c`)(`b`)`.

  $W, I \models$ `@(`c`)(`b`)`
  iff 1. I is right closed
  2. $W(\sup I) \models b$
  3. $\{t \in I : W(t) \models c\} = \{\sup I\}$
  $\Rightarrow$ $I(W, c)$ is a singleton
  $\Rightarrow$ $I(W, c)$ is finite

- $R = S$ `##1` S'. Suppose that $W, I \models S$ `##1` S'. Then $\exists J, J'$:

  1. $I = J \cup J'$
  2. $J < J'$
  3. $W, J \models S$ and $W, J' \models S'$

  By induction, $J(W, c)$ and $J'(W, c)$ are both finite. Since $I = J \cup J'$, $I(W, c) = J(W, c) \cup J'(W, c)$, which is again finite.

- $R = S$ `##0` S'. Suppose that $W, I \models S$ `##0` S'. Then $\exists J, J', a$:

  1. $I = J \cup J'$ and $J \cap J' = [a, a]$
  2. $J \leq [a, a]$
  3. $[a, a] \leq J'$
  4. $W, J \models S$ and $W, J' \models S'$

  By induction, $J(W, c)$ and $J'(W, c)$ are both finite. Since $I = J \cup J'$, $I(W, c) = J(W, c) \cup J'(W, c)$, which is again finite.

- $R = S$ `or` S'.

  $W, I \models S$ `or` S'
  iff $W, I \models S$ or $W, I \models S'$
  $\Rightarrow$ [induction]
  $I(W, c)$ is finite

- $R = S$ `intersect` S'.

  $W, I \models S$ `intersect` S'
  iff $W, I \models S$ and $W, I \models S'$
  $\Rightarrow$ [induction]
  $I(W, c)$ is finite

- $R = S$`[*0]`.

  $W, I \models S$`[*0]`
  iff I is empty
  $\Rightarrow$ $I(W, c)$ is finite

- $R = S$`[*1:$]`. Suppose that $W, I \models S$`[*1:$]`. Then $\exists n \geq 1$ and $J_1, ..., J_n$ such that

  1. $\forall 1 \leq i < j \leq n : J_i < J_j$
  2. $I = J_1 \cup ... \cup J_n$
  3. $\forall 1 \leq i \leq n : W, J_i \models S$

  By induction, for each i, $J_i(W, c)$ is finite. Since $I = J_1 \cup ... \cup J_n$, $I(W, c) = J_1(W, c) \cup ... \cup J_n(W, c)$, which is again finite.

□

Corollary 12: *Let R be a digital sequence singly clocked by c. Let*

$$I(W, c) = \{t \in I : W(t) \models c\}$$

If $W, I \models R$, then $I(W, c)$ has no limit point in $\mathbb{R}$. □

5 Realtime Sequences

The following is the grammar for realtime sequences:

$$
\begin{aligned}
R ::= & \; b \\
& \mid \texttt{@(}c\texttt{)(}b\texttt{)} \\
& \mid R \;\texttt{\#\#1}\; R \\
& \mid R \;\texttt{\#\#0}\; R \\
& \mid R \;\texttt{or}\; R \\
& \mid R \;\texttt{intersect}\; R \\
& \mid R\texttt{[*0]} \\
& \mid R\texttt{[*1:\$]} \\
& \mid R \;\texttt{\#0}\; R \\
& \mid R \;\texttt{without @(}c\texttt{)} \\
& \mid b\texttt{[*}\alpha\,[\texttt{+}] : \beta\,[\texttt{-}]\texttt{]}
\end{aligned}
$$

It is understood that `$` may stand for β, with meaning ∞.

The substrings "[+]" and "[-]" in the grammar above represent optional parts of the syntax. The "[" and "]" in these substrings are not terminals. Every other bracket in the grammar is typeset as "`[`" or "`]`" and is a terminal. Thus, "$b\texttt{[*}\alpha\,[\texttt{+}] : \beta\,[\texttt{-}]\texttt{]}$" abbreviates the following four variants:

$b\texttt{[*}\alpha : \beta\texttt{]}$
$b\texttt{[*}\alpha\texttt{+} : \beta\texttt{]}$
$b\texttt{[*}\alpha : \beta\texttt{-]}$
$b\texttt{[*}\alpha\texttt{+} : \beta\texttt{-]}$

Definition 13 (Realtime Sequence Semantics): *The semantics for the digital operators has already been given in Definition 4.*

- $W, I \models b$
  iff $\exists t$:
  1. $I = [t, t]$
  2. $W(t) \models b$

- $W, I \models R \;\texttt{\#0}\; R'$
  iff $\exists J, J'$:
  1. $J \leq J'$
  2. $I = J \cup J'$
  3. $W, J \models R$
  4. $W, J' \models R'$

- $W, I \models R \;\texttt{without @(}c\texttt{)}$
  iff
  1. $W, I \models R$
  2. $\forall t \in I : W(t) \not\models c$

- $W, I \models b\texttt{[*}\alpha\,[\texttt{+}] : \beta\,[\texttt{-}]\texttt{]}$
  iff
  1. $\alpha \; \leq [<] \; \mathit{length}(I) \; \leq [<] \; \beta$
  2. $\forall t \in I : W(t) \models b$

□

Derived Forms:

- b`[*`α`]` $\equiv b$`[*`$\alpha : \alpha$`]`
- R `#[`α`[+]:`β`[-]]` $R' \equiv R$ `#0 1[*`α`[+]:`β`[-]] #0` R'. Claim 20 below shows that `#0` is associative. Proposition 22 gives direct semantics for `#[`α`[+]:`β`[-]]` .
- R `#[`α`]` $R' \equiv R$ `#[`$\alpha : \alpha$`]` R'
- R `and` R'

  $\equiv$ `((`R `#0 1[*0.0:$]) intersect` R'`)`
  `or`
  `(`R `intersect (`R' `#0 1[*0.0:$]))`
- b`[->1]` $\equiv$ `!`b`[*0.0:$] #0` b. Proposition 23 gives direct semantics for `[->1]`.

□

6 Relating Semantics of Digital and Realtime Operators

Proposition 14: *Assume that R, R' are digital sequences, each singly clocked by c, and that neither admits empty match. Then $W, I \models R$ `##1` R' iff $W, I \models R$ `#0 (1[*0.0+:$] without @(`c`)) #0` R'.*

Proof: Note that

$W, I \models R$ `##1` R'
iff (A):
 $\exists J, J'$:
 1. $I = J \cup J'$
 2. $J < J'$
 3. $W, J \models R$ and $W, J' \models R'$

Also note that

$W, I \models R$ `#0 ((1[*0.0+:$] without @(`c`)) #0` R'`)`
iff $\exists J, J'$:
 1. $J \leq J'$
 2. $I = J \cup J'$
 3. $W, J \models R$
 4. $W, J' \models$ `((1[*0.0+:$] without @(`c`)) #0` R'`)`
iff (B):
 $\exists J, J'$:
 1. $J \leq J'$
 2. $I = J \cup J'$
 3. $W, J \models R$
 4. $\exists L, L'$:
  a. $L \leq L'$
  b. $J' = L \cup L'$
  c. $W, L \models$ `(1[*0.0+:$] without @(`c`))`
  d. $W, L' \models R'$

(⇒) Assume (A). Since neither R nor R' admits empty match, J and J' are non-empty. By Proposition 6, J is right closed and c is satisfied in W at the right endpoint of J. From $J < J'$ and $J \cup J' = I$ it then follows that J' is left-open. By Corollary 12, $J'(W, c)$ has no limit point in $\mathbb{R}$. Therefore, there is a strict prefix L of J' of positive length such that c is not satisfied in W at any point of L. Let $L' = J' - L$. By Lemma 7, $W, L' \models R'$. This shows that (B) holds.

(⇐) Assume (B). L has positive length and is a prefix of J'. J and L' are non-empty because R and R' do not admit empty match. By Proposition 6, J and L' are right closed and c is satisfied in W at their right endpoints. If J' were left closed, then the left endpoint would be in L and in J, which cannot be since c is not satisfied in W at any point of L. Therefore J' is left open. This shows that $J < J'$. Similarly, the right endpoint of L' cannot be in L, and so L is a strict prefix of J'. By Lemma 7, $W, J' \models R'$. This shows that (A) holds. □

Remark 15: Changing `#0` to `#0.0` in Proposition 14 above is problematic because occurrences of c can be skipped in the stitching together that is specified by `#0.0`. □

Notation 16: Let *empty* = `1[*0]`, *positive* = `1[*0.0+:$]`, *nonempty* = *positive* `or 1`. □

Claim 17:

- *$W, I \models$ empty iff I is empty.*
- *$W, I \models$ positive iff $length(I) > 0$.*
- *$W, I \models$ nonempty iff I is non-empty.*

Proof:

$W, I \models$ *empty*
iff $W, I \models$ `1[*0]`
iff I is empty

$W, I \models$ *positive*
iff $W, I \models$ `1[*0.0+:$]`
iff 1. $0 < length(I) \leq \infty$
2. $\forall t \in I : W(t) \models$ `1`
iff $0 < length(I)$

$W, I \models$ *nonempty*
iff $W, I \models$ *positive* `or 1`
iff either
$W, I \models$ *positive*
or

$W, I \models$ `1`
iff either
$length(I) > 0$
or
I is a singleton
iff I is non-empty

$\square$

Proposition 18: *Assume that R, R' are digital sequences, each singly clocked by c. Then $W, I \models R$* `##1` *R' iff*

$W, I \models$ R `and` `(`R' `intersect` *empty*`)`
`or`
`(`R `intersect` *empty*`)` `and` R'
`or`
`(`R `intersect` *nonempty*`)`
`#0 (`*positive* `without @(`c`))`
`#0 (`R' `intersect` *nonempty*`)`

Proof: ($\Rightarrow$) Assume $W, I \models R$ `##1` R'. Then $\exists J, J'$:

1. $I = J \cup J'$
2. $J < J'$
3. $W, J \models R$ and $W, J' \models R'$

- Case: J' empty. Then $W, I \models R$ and $W, \{\} \models R'$, hence

  $W, \{\} \models R'$ `intersect` *empty*

  Therefore

  $W, I \models$ `(`R' `intersect` *empty*`) #0 1[*0.0:$]`

  hence

  $W, I \models R$ `intersect ((`R' `intersect` *empty*`) #0 1[*0.0:$])`

  hence

  $W, I \models R$ `and (`R' `intersect` *empty*`)`
- Case: J empty. Similar.
- Case: J and J' both non-empty. Then

  $W, J \models R$ `intersect` *nonempty*

  and

  $W, J' \models R'$ `intersect` *nonempty*

  so

  $W, I \models$ `(`R `intersect` *nonempty*`) ##1 (`R' `intersect` *nonempty*`)`

  Note that

$W, I \models$ R `intersect` *nonempty*
iff [Claim 17]
I is non-empty and $W, I \models$ R
iff [Proposition 6, R is singly clocked by c]
$W, I \models$ R `and @(`c`)(1)`

Therefore, we may consider "R `intersect` *nonempty*" (and similarly with R' in place of R) to be singly clocked by c and not to admit empty match. By Proposition 14,

$W, I \models$ R `and (`R' `intersect` *empty*`)`
`or`
`(`R `intersect` *empty*`) and` R'
`or`
`(`R `intersect` *nonempty*`)`
`#0 (`*positive* `without @(`c`))`
`#0 (`R' `intersect` *nonempty*`)`

($\Leftarrow$) Assume that

$W, I \models$ R `and (`R' `intersect` *empty*`)`
`or`
`(`R `intersect` *empty*`) and` R'
`or`
`(`R `intersect` *nonempty*`)`
`#0 (`*positive* `without @(`c`))`
`#0 (`R' `intersect` *nonempty*`)`

- Case: $W, I \models R$ and `(`R' `intersect` *empty*`)`.
  - Subcase: $W, I \models$ `(`R `#0 1[*0.0:$]) intersect (`R' `intersect` *empty*`)`. Then $I = \{\}$. Therefore,

    $$W, \{\} \models R \quad \text{and} \quad W, \{\} \models R' \ ,$$

    hence $W, \{\} \models R$ `##1` R', hence $W, I \models R$ `##1` R'.
  - Subcase: $W, I \models R$ `intersect ((`R' `intersect` *empty*`) #0 1[*0.0:$])`. Then $W, I \models R$ and $W, I \models$ `(`R' `intersect` *empty*`) #0 1[*0.0:$]`. Therefore, $\exists K$ such that $W, K \models R'$ `intersect` *empty*. K must be empty, so $W, \{\} \models R'$. Therefore $W, I \models R$ `##1` R'.
- Case: $W, I \models$ `(`R `intersect` *empty*`)` and R'. Similar.
- Case:

  $W, I \models$ `(`R `intersect` *nonempty*`)`
  `#0 (`*positive* `without @(`c`))`
  `#0 (`R' `intersect` *nonempty*`)`

  "R `intersect` *nonempty*" (and similarly with R' in place of R) may be considered, as above, to be singly clocked by c and not to admit empty match. By Proposition 14,

  $W, I \models$ `(`R `intersect` *nonempty*`) ##1 (`R' `intersect` *nonempty*`)` ,

  which implies that $W, I \models R$ `##1` R'.

□

Proposition 19: *Assume that R, R' are digital, each singly clocked by c. $W, I \models R$ `##0` R' iff*

$W, I \models$ `(`R `intersect` *nonempty*`)`
`#0 (((1 intersect @(`c`)(1)) #0 1[*0.0:$]) intersect` R'`)`

Proof:

$W, I \models$ `(`R `intersect` *nonempty*`)`
`#0 (((1 intersect @(`c`)(1)) #0 1[*0.0:$]) intersect` R'`)`
iff $\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
3. J is non-empty and $W, J \models R$
4. $W, J' \models$ `((1 intersect @(`c`)(1)) #0 1[*0.0:$]) intersect` R'

iff $\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
3. J is non-empty and $W, J \models R$
4. $W, J' \models$ `(1 intersect @(`c`)(1)) #0 1[*0.0:$]`
5. $W, J' \models R'$

iff $\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
3. J is non-empty and $W, J \models R$
4. J' is left closed and $W(\inf J') \models c$
5. $W, J' \models R'$

iff [Proposition 6]
$\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
3. $W, J \models R$ and J is right closed and $W(\sup J) \models c$
4. J' is left closed and $W(\inf(J')) \models c$
5. $W, J' \models R'$

iff [
(⇒) $\sup J = \inf J'$, and let this point be a.
(⇐) $\sup J = \inf J' = a$. By Proposition 6, $W(a) \models c$.
]
$\exists J, J', a$:
1. $I = J \cup J'$ and $J \cap J' = [a, a]$
2. $J \leq [a, a]$
3. $[a, a] \leq J'$
4. $W, J \models R$ and $W, J' \models R'$

iff $W, I \models R$ `##0` R'

□

7 Recovering Digital Semantics

This section proves that the realtime semantics of digital sequences is consistent with the discrete sampled semantics of the same sequences as already standardized in SVA, provided that the set of points where clocking events occur has no limit point in $\mathbb{R}$. The primary result is the following:

Proposition 0: *Let R be a digital sequence singly clocked by c. Let T be a sampling such that $T(\mathbb{N})$ contains the set $\{t \in \mathbb{R} : W(t) \models c\}$. Let $w = W \circ T$. Let I be a bounded interval. If I is empty, let v be the empty word. Otherwise, assume that I is right-closed with* $\sup I \in T(\mathbb{N})$*, and define $v = w^{i..j}$, where i is the minimum and j the maximum natural numbers that are mapped by T into I. Then $W, I \models R$ in the realtime semantics iff $v \models R$ in the SVA semantics.*

Proof: If I is non-empty, then $0 \le i \le j$, $T(j) = \sup I$ (hence $w^j = W(\sup I)$), and

$$\{t \in I : W(t) \models c\} \subseteq \{T(i), \ldots, T(j)\} .$$

If I is empty, then let $0 \le j$ and $i = j + 1$, so that $w^{i..j} = v$ (i.e., the empty word). In any case,

$$\{T(i), \ldots, T(j)\} = T(\mathbb{N}) \cap I$$

and

$$\{i, \ldots, j\} = \{n \in \mathbb{N} : T(n) \in I\} .$$

Let U be the unclocked sequence that results from R by applying the SVA clock rewrite rules. By the SVA semantics, $v \models R$ iff $v \models U$ in the unclocked semantics. Therefore, it suffices to show that $W, I \models R$ in the realtime semantics iff $v \models U$ in the unclocked SVA semantics. This is done by induction. R`[+]` abbreviates R`[*1:$]`, and R`[*]` abbreviates R`[*0] or` R`[*1:$]`.

- $R =$ `@(`c`)(`b`)`. By clock rewrite rules, $U =$ `!`c`[*] ##1` c `&&` b. Then

  $v \models U$
  iff $v \models$ `!`c`[*] ##1` c `&&` b
  iff [v has no special letters $\top$, $\bot$]
  1. $|v| > 0$
  2. $v^{|v|-1} \models c$ `&&` b
  3. $v^k \not\models c$ for all $0 \le k < |v| - 1$

  iff [$|v| > 0$ iff I is non-empty, right-closed, and $v = w^{i..j}$]
  1. I is non-empty and $\sup I \in I$
  2. $w^j \models c$ `&&` b
  3. $w^k \not\models c$ for all $i \le k < j$

  iff [
  I non-empty implies $T(j) = \sup I$, hence $w^j = W(\sup I)$.
  ($\Rightarrow$) Suppose $t \in I - \{\sup I\}$ and $W(t) \models c$. Then $t \in \{T(i), \ldots, T(j-1)\}$, hence $W(t) \in \{w^i, \ldots, w^{j-1}\}$, and so by 3 above, $W(t) \not\models c$, a contradiction.
  ($\Leftarrow$) Suppose $i \le k < j$ and $w^k \models c$. Then $T(k) \in I - \sup I$ and $W(T(k)) = w^k \models c$, a contradiction of 3 below.
  ]
  1. I is non-empty and $\sup I \in I$
  2. $W(\sup I) \models c$ `&&` b

3. $W(t) \not\models c$ for all $t \in I - \{\sup I\}$
iff 1. I is non-empty and $\sup I \in I$
2. $W(\sup I) \models b$
3. $\{t \in I : W(t) \models c\} = \{\sup I\}$
iff $W, I \models$ `@(`c`)(`b`)`

- $R = R_1$ `##1` R_2. Then $U = U_1$ `##1` U_2, where U_e is the clock rewrite of R_e for $e = 1, 2$.

$v \models U_1$ `##1` U_2
iff $\exists\, v_1, v_2 : v = v_1 v_2$ and $v_1 \models U_1$ and $v_2 \models U_2$
iff $[v = w^{i..j}]$
$\exists\, i - 1 \leq k \leq j : w^{i..k} \models U_1$ and $w^{k+1..j} \models U_2$
iff [
($\Rightarrow$)
Let $I_1 = I \cap [0, T(k)]$, $I_2 = I \cap (T(k), \infty)$.
If $I_1 = \{\}$, then $T(k) \notin I$, so $k < i$ and $w^{i..k}$ is empty.
If $I_1 \neq \{\}$, then $I \neq \{\}$, hence $\sup I = T(j) \geq T(k)$. Since $I_1 \neq \{\}$, we must have $T(k) \in I$, hence $T(k) \in I_1$ and $T(k) = \sup I_1$. Also, $T(k) \geq T(i)$. Therefore, $\{i, \ldots, k\} = \{n \in \mathbb{N} : T(n) \in I_1\}$.
If $I_2 = \{\}$, then $T(k+1) \notin I$, so $k + 1 > j$ and $w^{k+1..j}$ is empty.
If $I_2 \neq \{\}$, then $I \neq \{\}$, hence $T(j) = \sup I$. $k = j$ implies $I_2 = \{\}$, so $k < j$. Therefore, $T(k+1) \in I$ and $\{k+1, \ldots, j\} = \{n \in \mathbb{N} : T(n) \in I_2\}$. Then $\sup I_2 = T(j)$.
By induction, $W, I_1 \models R_1$ and $W, I_2 \models R_2$.
Clearly, $I_1 < I_2$.
($\Leftarrow$)
If $I_1 = \{\}$, let $k = i - 1$. Then $w^{i..k}$ is empty. Then $I_2 = I$, $w^{k+1..j} = w^{i..j}$, and $\{T(i), \ldots, T(j)\} = \{n \in \mathbb{N} : T(n) \in I\}$.
If $I_1 \neq \{\}$, then by Proposition 6, I_1 is right-closed and $W(\sup I_1) \models c$. Therefore, $\sup I_1 \in \{T(i), \ldots, T(j)\}$. Let $\sup I_1 = T(k)$, where $i \leq k \leq j$. Then $\{i, \ldots, k\} = \{n \in \mathbb{N} : T(n) \in I_1\}$, and $\{k + 1, \ldots, j\} = \{n \in \mathbb{N} : T(n) \in I_2\}$. If $k = j$, then $I_2 = \{\}$ since $I_1 < I_2$. In this case, also $w^{k+1..j}$ is empty.
By induction, $w^{i..k} \models U_1$ and $w^{k+1..j} \models U_2$.
]
$\exists\, I_1, I_2$:
1. $I = I_1 \cup I_2$
2. $I_1 < I_2$
3. $W, I_1 \models R_1$ and $W, I_2 \models R_2$
iff $W, I \models R_1$ `##1` R_2

- $R = R_1$ `##0` R_2. Then $U = U_1$ `##0` U_2, where U_e is the clock rewrite of R_e for $e = 1, 2$.

$v \models U_1$ `##0` U_2
iff $\exists\, v_1, x, v_2 : v = v_1 x v_2$ and $|x| = 1$ and $v_1 x \models U_1$ and $x v_2 \models U_2$
iff $[v = w^{i..j}]$
$\exists\, i \leq k \leq j : w^{i..k} \models U_1$ and $w^{k..j} \models U_2$
iff [

(⇒)
Since $i \le k \le j$, I is non-empty and $T(k) \in I$.
Let $I_1 = I \cap [0, T(k)]$, $I_2 = I \cap [T(k), \infty)$. Then
– $I_1 \cup I_2 = I$ and $I_1 \cap I_2 = \{T(k)\}$
– $I_1 \le \{T(k)\}$
– $\{T(k)\} \le I_2$
Also, $T(k) = \sup I_1$, $T(j) = \sup I_2$,

$$\{i, \ldots, k\} = \{n \in \mathbb{N} : T(n) \in I_1\}$$

and

$$\{k, \ldots, j\} = \{n \in \mathbb{N} : T(n) \in I_2\}$$

By induction, $W, I_1 \models R_1$ and $W, I_2 \models R_2$.
(⇐)
$I_1 \neq \{\}$, so by Proposition 6, I_1 is right-closed and $W(\sup I_1) \models c$. Therefore, $t = \sup I_1 = \inf I_2 \in \{T(i), \ldots, T(j)\}$. Let $\sup I_1 = T(k)$, where $i \le k \le j$. Then

$$\{i, \ldots, k\} = \{n \in \mathbb{N} : T(n) \in I_1\}$$

and

$$\{k, \ldots, j\} = \{n \in \mathbb{N} : T(n) \in I_2\}$$

Also, $\sup I_2 = T(j)$. By induction, $w^{i..k} \models U_1$ and $w^{k..j} \models U_2$.
]
$\exists\ I_1, I_2, t :$
1. $I = I_1 \cup I_2$ and $I_1 \cap I_2 = \{t\}$
2. $I_1 \le \{t\}$
3. $\{t\} \le I_2$
4. $W, I_1 \models R_1$ and $W, I_2 \models R_2$

iff $W, I \models R_1$ `##0` R_2

- $R = R_1$ `or` R_2. Then $U = U_1$ `or` U_2, where U_e is the clock rewrite of R_e for $e = 1, 2$.

  $v \models U_1$ `or` U_2
  iff $v \models U_1$ or $v \models U_2$
  iff [induction]
  $W, I \models R_1$ or $W, I \models R_2$
  iff $W, I \models R_1$ `or` R_2

- $R = R_1$ `intersect` R_2. Then $U = U_1$ `intersect` U_2, where U_e is the clock rewrite of R_e for $e = 1, 2$.

  $v \models U_1$ `intersect` U_2
  iff $v \models U_1$ and $v \models U_2$
  iff [induction]
  $W, I \models R_1$ and $W, I \models R_2$
  iff $W, I \models R_1$ `intersect` R_2

- $R = R_1$`[*0]`. Then $U = U_1$`[*0]`, where U_1 is the clock rewrite of R_1.

  $v \models U_1$`[*0]`
  iff $|v| = 0$
  iff I is empty
  iff $W, I \models R_1$`[*0]`

- $R = R_1$`[*]`. Then $U = U_1$`[*]`, where U_1 is the clock rewrite of R_1.
  Case: v empty. Then $I = \{\}$.

$v \models U_1$ [*]
iff $\exists\, n \geq 1$ and $v_1, \ldots, v_n$ such that
1. $v = v_1 \cdots v_n$
2. $v_k \models U_1$ for all $1 \leq k \leq n$

iff [($\Rightarrow$) v empty implies all v_k are empty]
$v \models U_1$
iff [induction]
$W, I \models R_1$
iff [($\Leftarrow$) I empty implies all J_k are empty]
$\exists\, n \geq 1$ and $J_1, \ldots, J_n$ such that
1. $\forall\, 1 \leq k < l \leq n : J_k < J_l$
2. $I = J_1 \cup \cdots \cup J_n$
3. $\forall\, 1 \leq k \leq n : W, J_k \models R_1$

iff $W, I \models R_1$ [*]

Case: v non-empty. Then I is right-closed.

$v \models U_1$ [*]
iff [($\Rightarrow$) v non-empty implies, WLOG, all v_k are non-empty]
$\exists\, n \geq 1$ and non-empty $v_1, \ldots, v_n$ such that
1. $v = v_1 \cdots v_n$
2. $v_k \models U_1$ for all $1 \leq k \leq n$

iff [

($\Rightarrow$) $w^{i..j} = v = v_1...v_n$. All v_k are non-empty, so we can write $v_k = w^{i_k..j_k}$, where $i_1 = i$, $j_n = j$, and $i_{k+1} = (j_k) + 1$ for $1 \leq k < n$. Let

$$I_1 = I \cap [0, T(j_1)]$$

and

$$I_k = I \cap (T(j_{k-1}), T(j_k)]$$

for $1 < k \leq n$. Then for all $1 \leq k \leq n$, $T(j_k) = \sup I_k$ and

$$\{i_k, \ldots, j_k\} = \{n \in \mathbb{N} : T(n) \in I_k\}$$

We have $I = I_1 \cup \cdots \cup I_n$ and $I_k < I_l$ for $1 \leq k < l \leq n$. By induction, for all $1 \leq k \leq n$, $W, I_k \models R_1$.
($\Leftarrow$) Since I_k is non-empty, Proposition 6 implies that I_k is right-closed and $W(\sup I_k) \models c$. Therefore, $\sup I_k \in T(\mathbb{N})$. Write $\sup I_k = T(j_k)$. Let $i_k = \min\{n \in \mathbb{N} : T(n) \in I_k\}$. Then

$$\{i_k, \ldots, j_k\} = \{n \in \mathbb{N} : T(n) \in I_k\}$$

Let $v_k = w^{i_k..j_k}$. Then $v = v_1...v_n$. By induction, $v_k \models U_1$.

]
$\exists\, n \geq 1$ and non-empty $I_1, ..., I_n$ such that
1. $\forall\, 1 \leq k < l \leq n : I_k < I_l$
2. $I = I_1 \cup \cdots \cup I_n$
3. $\forall\, 1 \leq k \leq n : W, I_k \models R_1$

iff [($\Leftarrow$) I non-empty implies, WLOG, all I_k are non-empty]
$W, I \models R_1$ [*]

□

Corollary: *Let R be a digital sequence that is admissible in SVA as a multiply clocked sequence. Let T be a sampling such that for every clock c of R, $T(\mathbb{N})$ contains the set $\{t \in \mathbb{R} : W(t) \models c\}$. Let $w = W \circ T$. Let I be* *a non-empty bounded interval that is right-closed with* $\sup I \in T(\mathbb{N})$*. Define* $v = w^{i..j}$*, where* i *is the minimum and* j *the maximum natural numbers that are mapped by* T *into* I*. Then* $W, I \models R$ *in the realtime semantics iff* $v \models R$ *in the SVA semantics.*

Proof: Since R is admissible in SVA as a multiply clocked sequence, there exists $m \geq 0$ such that

$$R = R_0 \ \texttt{\#\#}n_1 \ \cdots \ \texttt{\#\#}n_m \ R_m \ ,$$

where each $n_j \in \{\texttt{0}, \texttt{1}\}$ and each R_j is a singly clocked sequence that does not admit empty match.

The proof is by induction on m. If $m = 0$, then the result follows from Proposition 0. The arguments for the cases `##1` and `##0` in the proof of Proposition 0 can be adapted to prove the inductive step, splitting on the cases $n_1 = \texttt{1}$ and $n_1 = \texttt{0}$.

□

8 Sanity Checks

Claim 20: `#0` *is associative.*

Proof:

$W, I \models R \ \texttt{\#0}\ (R' \ \texttt{\#0}\ R'')$
iff $\exists J, K$:
  A1. $J \leq K$
  A2. $I = J \cup K$
  A3. $W, J \models R$
  A4. $W, K \models R' \ \texttt{\#0}\ R''$
iff $\exists J, K$:
  A1. $J \leq K$
  A2. $I = J \cup K$
  A3. $W, J \models R$
  A4. $\exists J', J''$:
    a. $J' \leq J''$
    b. $K = J' \cup J''$
    c. $W, J' \models R'$
    d. $W, J'' \models R''$
iff [
  (⇒) Let $L = J \cup J'$.
    1. $J \leq K$ [A1]
    2. $K = J' \cup J''$ [A4.b]
    3. $J \leq J' \cup J''$ [1,2]
    4. $J \leq J'$ and $J \leq J''$ [3, Claim 3]
    5. $J \leq J''$ [4]
    6. $J \leq J''$ and $J' \leq J''$ [5, A4.a]
    7. $J \cup J' \leq J''$ [6, Claim 3]
    8. $L \leq J''$
  $L \cup J'' = J \cup J' \cup J''$
    $= J \cup K$ [A4.b]
    $= I$ [A2]
  4 implies B3.a.

A3 iff B3.c.
A4.c iff B3.d.
A4.d iff B4.

($\Leftarrow$) Let $K = J' \cup J''$.

1. $L \leq J''$	[B1]
2. $L = J \cup J'$	[B3.b]
3. $J \cup J' \leq J''$	[1,2]
4. $J \leq J''$ and $J' \leq J''$	[3, Claim 3]
5. $J \leq J''$	[4]
6. $J \leq J'$ and $J \leq J''$	[5, B3.a]
7. $J \leq J' \cup J''$	[6, Claim 3]
8. $J \leq K$	
$J \cup K = J \cup J' \cup J''$	
$= L \cup J''$	[B3.b]
$= I$	[B2]

A3 iff B3.c.
4 implies A4.a
A4.c iff B3.d.
A4.d iff B4.
]
$\exists L, J''$:
B1. $L \leq J''$
B2. $I = L \cup J''$
B3. $\exists J, J'$:
a. $J \leq J'$
b. $L = J \cup J'$
c. $W, J \models R$
d. $W, J' \models R'$
B4. $W, J'' \models R''$
iff $\exists L, J''$:
B1. $L \leq J''$
B2. $I = L \cup J''$
B3. $W, L \models R$ `#0` R'
B4. $W, J'' \models R''$
iff $W, I \models$ `(`R `#0` R'`)` `#0` R''

□

Corollary 21: `#[`α`[+]` : β`[-]]` *is associative.*

Proof:

$W, I \models R$ `#[`α`[+]` : β`[-]]` `(`R' `#[`γ`[+]` : δ`[-]]` R''`)`
iff $W, I \models R$ `#0 1[*`α`[+]` : β`[-]]` `#0` `(`R' `#0 1[*`γ`[+]` : δ`[-]]` `#0` R''`)`
iff [`#0` is associative by Claim 20]
$W, I \models$ `(`R `#0 1[*`α`[+]` : β`[-]]` `#0` R'`)` `#0 1[*`γ`[+]` : δ`[-]]` `#0` R''
iff $W, I \models$ `(`R `#[`α`[+]` : β`[-]]` R'`)` `#[`γ`[+]` : δ`[-]]` R''

□

Proposition 22 (Direct Semantics of `#[`α`[+]` : β`[-]]` **):**
$W, I \models R$ `#[`α`[+]` : β`[-]]` R' *iff*

$\exists J, K, J'$:
1. $J \leq K$ *and* $K \leq J'$ *and* $J \leq J'$
2. $I = J \cup K \cup J'$
3. $\alpha\ \leq [<]\ \mathit{length}(K) \leq [<]\ \beta$
4. $W, J \models R$
5. $W, J' \models R'$

Proof:

$W, I \models R$ `#[`α`[+]`$:\beta$`[-]]` R'
iff $W, I \models R$ `#0 1[*`α`[+]`$:\beta$`[-]] #0` R'
iff $\exists J, J_1$:
1. $J \leq J_1$
2. $I = J \cup J_1$
3. $W, J \models R$
4. $W, J_1 \models$ `1[*`α`[+]`$:\beta$`[-]] #0` R'

iff $\exists J, J_1$:
1. $J \leq J_1$
2. $I = J \cup J_1$
3. $W, J \models R$
4. $\exists K, J'$:
   a. $K \leq J'$
   b. $J_1 = K \cup J'$
   c. $W, K \models$ `1[*`α`[+]`$:\beta$`[-]]`
   d. $W, J' \models R'$

iff $[J_1 = K \cup J']$
$\exists J, K, J'$:
1. $J \leq K \cup J'$
2. $I = J \cup K \cup J'$
3. $W, J \models R$
4.a. $K \leq J'$
4.c. $W, K \models$ `1[*`α`[+]`$:\beta$`[-]]`
4.d. $W, J' \models R'$

iff $[\forall t \in K : W(t) \models$ `1`$]$
$\exists J, K, J'$:
1. $J \leq K \cup J'$
2. $I = J \cup K \cup J'$
3. $W, J \models R$
4.a. $K \leq J'$
4.c. $\alpha \leq [<]\ \mathit{length}(K) \leq [<]\ \beta$
4.d. $W, J' \models R'$

iff [Claim 3]
$\exists J, K, J'$:
1. $J \leq K$ and $K \leq J'$ and $J \leq J'$
2. $I = J \cup K \cup J'$
3. $\alpha\ \leq [<]\ \mathit{length}(K) \leq [<]\ \beta$
4. $W, J \models R$
5. $W, J' \models R'$

□

Proposition 23 (Direct Semantics of `[->1]`): $W, I \models b$`[->1]` *iff*

1. I *is non-empty and* $\sup I \in I$

2. $\{t \in I : W(t) \models b\} = \{\sup I\}$

Proof: Note that

$W, I \models$ `!`b`[*0.0 : $]`
iff 1. $0 \leq \mathit{length}(I) \leq \infty$
 2. $\forall t \in I : W(t) \models$ `!`b
iff $\forall t \in I : W(t) \not\models b$

Then

$W, I \models b$`[->1]`
iff $W, I \models$ `!`b`[*0.0 : $] #0` b
iff $\exists J, J'$:
 1. $J \leq J'$
 2. $I = J \cup J'$
 3. $W, J \models$ `!`b`[*0.0 : $]`
 4. $W, J' \models b$
iff $\exists J, t$:
 1. $J \leq [t, t]$
 2. $I = J \cup [t, t]$
 3. $W, J \models$ `!`b`[*0.0 : $]`
 4. $W(t) \models b$
iff [
 ($\Rightarrow$) 1 and 2 above imply that I is non-empty and $t = \sup I$, hence 1 below. 4 above implies $W(\sup I) \models b$. 3 and 4 above imply that $J \cap [t, t] = \{\}$, hence $J = I - \{\sup I\}$. 3 above then implies $\forall t \in I - \{\sup I\} : W(t) \not\models b$, hence 2 below.
 ($\Leftarrow$) Choose $t = \sup I, J = I - \{\sup I\}$.
 ]
 1. I is non-empty and $\sup I \in I$
 2. $\{t \in I : W(t) \models b\} = \{\sup I\}$

□

Test 24:

$W, I \models e$ `#0.0` R'
iff $\exists J, K, J'$:
 1. $J \leq K$ and $K \leq J'$ and $J \leq J'$
 2. $I = J \cup K \cup J'$
 3. $0 \leq \mathit{length}(K) \leq 0$
 4. $W, J \models e$ and
 5. $W, J' \models R'$
iff $\exists J, K, J'$:
 1. $J \leq K$ and $K \leq J'$ and $J \leq J'$
 2. $I = J \cup K \cup J'$
 3. $\mathit{length}(K) = 0$
 4. $\exists t : J = [t, t]$ and $W(t) \models e$
 5. $W, J' \models R'$

iff $[J = [t,t], K \subseteq J]$
$\exists t, J'$:
1. $\{t\} \leq J'$
2. $I = \{t\} \cup J'$
4. $W(t) \models e$
5. $W, J' \models R'$

$\square$

Test 25:

$W, I \models e$`[*0.0] #0.0` R'
iff $\exists J, K, J'$:
1. $J \leq K$ and $K \leq J'$ and $J \leq J'$
2. $I = J \cup K \cup J'$
3. $0 \leq \mathit{length}(K) \leq 0$
4. $W, J \models e$`[*0.0]`
5. $W, J' \models R'$

iff $\exists J, K, J'$:
1. $J \leq K$ and $K \leq J'$ and $J \leq J'$
2. $I = J \cup K \cup J'$
3. $\mathit{length}(K) = 0$
4. $\mathit{length}(J) = 0$ and $\forall t \in J : W(t) \models e$
5. $W, J' \models R'$

iff either $[J = [t,t], K \subseteq J]$
$\exists t, J'$:
1. $\{t\} \leq J'$
2. $I = \{t\} \cup J'$
3. $W(t) \models e$
4. $W, J' \models R'$

or $[J = \{\}]$
$\exists K, J'$:
1. $K \leq J'$
2. $I = K \cup J'$
3. $\mathit{length}(K) = 0$
4. $W, J' \models R'$

$\square$

Claim 26: $W, I \models$ `1[*0.0:$]`

Proof:

$W, I \models$ `1[*0.0:$]`
iff both
1. $0 \leq \mathit{length}(I) \leq \infty$
and
2. $\forall t \in I : W(t) \models$ `1`
iff TRUE

□

Claim 27: $W, I \models e$ `#0 1[*0.0:$]` *iff* $\exists t, J'$:

1. $[t,t] \leq J'$
2. $I = [t,t] \cup J'$
3. $W(t) \models e$

Proof:

$W, I \models e$ `#0 1[*0.0:$]`
iff $\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
3. $W, J \models e$ and
4. $W, J' \models$ `1[*0.0:$]`

iff [Claim 26]
$\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
3. $W, J \models e$

iff $\exists t, J'$:
1. $[t,t] \leq J'$
2. $I = [t,t] \cup J'$
3. $W(t) \models e$

□

Claim 28: $W, I \models R$ `#0.0` b *iff* $W, I \models R$ `#0` b.

Proof:

$W, I \models R$ `#0.0` b
iff $\exists J, K, J'$:
1. $J \leq K$ and $K \leq J'$ and $J \leq J'$
2. $I = J \cup K \cup J'$
3. $length(K) = 0$
4. $W, J \models R$
5. $W, J' \models b$

iff $\exists J, K, t$:
1. $J \leq K$ and $K \leq [t,t]$ and $J \leq [t,t]$
2. $I = J \cup K \cup [t,t]$
3. $length(K) = 0$
4. $W, J \models R$
5. $W(t) \models b$

iff [

(⇒) 1, 4, 5 above imply 1, 4, 5 below. $length(K) = 0$, so either $K = \{\}$ or $K = [a, a]$. If $K = \{\}$, then 2 above implies 2 below. Suppose $K = [a, a]$. If J is empty, then $I = [a, a] \cup [t, t]$, and since I is an interval, $a = t$, hence 2 below follows. Otherwise, J is non-empty and $J \leq K$ implies $\sup J \leq a$. From $K \leq [t, t]$, $a \leq t$. $a < t$ again contradicts the fact that I is an interval, so $a = t$. Therefore $K = [t, t]$, and 2 below follows.
(⇐) Let $K = \{\}$.

]

$\exists J, t$:
1. $J \leq [t, t]$
2. $I = J \cup [t, t]$
4. $W, J \models R$
5. $W(t) \models b$

iff $\exists J, J'$:
1. $J \leq J'$
2. $I = J \cup J'$
4. $W, J \models R$
5. $W, J' \models b$

iff $W, I \models R$ `#0` b

□

References

[1] D. Nickovic. *Checking Timed and Hybrid Properties: Theory and Applications.* PhD thesis, Université Joseph Fourier, 2009.