



Background: LF Federated Execution

A Tag is basically like a logical time, so it makes sense to add a logical delay to a Tag.

Throughout, e_1 and e_2 will denote events occurring in a process called the RTI. Every event involves the RTI either sending a message to a federate, or receiving a message from a federate. So, if $\text{Federate}(e_1) = f$, then that means that e_1 is an event in which the RTI either sends a message to f or receives a message from f .

Federates have ports. Different federates are connected to each other via their ports, possibly using multiple connections, and every connection has a nonnegative logical delay associated with it. If a connection goes from federate A to federate B , and the delay associated with that connection is D , then that means that when federate A is executing some tag T , it is possible for federate A to send a signal that logically reaches federate B at tag $T' := T + D$.

The RTI ensures that if a federate A sends a signal to a federate B that logically reaches federate B at time T' , federate B will not execute anything at a tag later than T' before it receives the signal. Furthermore, before federate B receives the signal, federate B can only execute things at the tag T' that are statically guaranteed not to be affected by the signal. When federate B violates these rules by executing something too early, we call that an STP (safe-to-process) violation.

In addition to preventing STP violations, the RTI must prevent deadlocks by allowing federates to proceed forward to logical times when they may have events to process.

The RTI can be described as receiving information from federates, possibly aggregating it into information about the current state of the system, and then propagating the information in such a way that federates can proceed without STP violations.

To ensure that messages arrive in the right order, even messages that conceptually should go from federate to federate go through the RTI. For example, `PORT_ABS` and `TAGGED_MSG` messages are sent from an upstream federate to the RTI, and then from the RTI to the downstream federate. However, most messages simply go from a federate to the RTI or from the RTI to a federate, and do not directly correspond to any federate-to-federate communication. For example, federates do not send `TAG` or `PTAG` messages; instead, they only receive such messages from the RTI, because it is up to the RTI, not the federates, to control federates' time advancement.

The RTI also coordinates program initialization and shutdown.

A given message is of one of the following types:

1. FED_ID (Federate ID): This is sent from a federate to the RTI during program initialization so that the federate can declare its ID to the RTI.
2. ACK (ACKnowledgement): This message acknowledges receipt of a federate ID.
3. TIMESTAMP: This message is used during initialization to help the federates and the RTI agree on a start time for the main part of the program.
4. NET (Next Event Tag): This message is sent from a federate to the RTI to declare the time of the next event that has been scheduled to occur at that federate so far.
5. PORT_ABS (PORT ABSent): This message communicates that an upstream federate is not going to send a signal that will be received at a particular port of a downstream federate at a particular tag. The federate associated with a PORT_ABS message that the RTI receives is the upstream federate, because that is the federate that the RTI is communicating with; and the federate associated with a PORT_ABS message that the RTI sends is the downstream federate, because then the RTI is communicating to the downstream federate. However, the tag associated with a PORT_ABS message always corresponds to a (logical) tag from the perspective of the downstream federate, both when that message is received by the RTI and when it is forwarded from the RTI to the downstream federate.
6. PTAG (Provisional Tag Advance Grant): This message is sent from the RTI to a federate to communicate that the federate is allowed to proceed forward in logical time up to but not including a particular tag.
7. TAGGED_MSG (Tagged MeSsaGe): This is a signal that goes from a federate to another federate via some connection. The only difference between a TAGGED_MSG and a PORT_ABS message is that a TAGGED_MSG indicates presence of a signal on the connection instead of absence; however, otherwise is very similar to a PORT_ABS message. In particular, just like with a PORT_ABS message, the federate associated with a TAGGED_MSG depends on whether the message is being received or sent by the RTI, whereas its associated tag always corresponds to a (logical) tag from the perspective of the downstream federate.
8. TAG (Tag Advance Grant): This message is sent from the RTI to a federate to communicate that the federate is allowed to proceed forward in logical time up to and including a particular tag.
9. STOP_REQ (STOP REQuest)

10. STOP_REQ_REP (STOP REQuest REPlY)
11. STOP_GRN (STOP GRaNted)
12. LTC (Logical Tag Complete): This message is sent from a federate to the RTI to declare that the federate is finished executing a particular tag.

TAGGED_MSG messages that are received from an upstream federate while the upstream federate is executing a tag T are associated with the tag $T + D$ when a federate-to-federate signal on a connection will be received. PORT_ABS messages are similar.

Preliminary Syntax Explanation

$e_1 < e_2$ means that it is not possible, under any physical, real-life execution of the federated program, for e_1 to occur after e_2 in physical time.

Formulas are stated in an S-expression-like format. For example, where we write $(f\ e_1)$, we mean "f of e_1 ".

All events are described from the perspective of the RTI. For example, where we write

$(e_1\ \text{is}\ (\text{Receiving}\ \text{PORT_ABS}))$, we mean that e_1 is an event in which a federate sends a PORT_ABS message, and the RTI receives the PORT_ABS message. Similarly, where we write $((e_1\ \text{is}\ (\text{Sending}\ \text{PORT_ABS})))$, we mean that e_1 is an event in which the RTI sends a PORT_ABS message, and a federate receives the PORT_ABS message.

The above context, which was provided to an LLM, was written by a human. However, most of the remaining text in this document is machine-generated. Human-generated text or commentary that does not come from an LLM will be presented in *italics*. Be warned that some of the content produced by the LLM, *especially* the content labeled as "high-level justification," may contain conceptual mistakes that in a human would indicate a lack of deep understanding.

Formula 1

Formula 1 states:

```

(((e1 is (Receiving LTC)))  $\wedge$  (Federate (e1) = Federate (e2))  $\wedge$  ((Tag e1) <
(Tag e2)))
 $\wedge$  ((e2 is (Receiving LTC))))
 $\Rightarrow$  (e1 < e2)

```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas:

1. Atomic Sub-formulas:

- `(e_1 is (Receiving LTC))` : This atomic sub-formula is true when event e_1 involves the RTI receiving a Logical Tag Complete (LTC) message.
- `(Federate(e_1) = Federate(e_2))` : This atomic sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- `((Tag e_1) < (Tag e_2))` : This atomic sub-formula is true when the logical tag of event e_1 is less than the logical tag of event e_2 .
- `(e_2 is (Receiving LTC))` : This atomic sub-formula is true when event e_2 involves the RTI receiving a Logical Tag Complete (LTC) message.

2. Larger Sub-formulas:

- `((e_1 is (Receiving LTC)) \wedge (Federate(e_1) = Federate(e_2)) \wedge ((Tag e_1) < (Tag e_2)))` : This sub-formula is true when event e_1 involves the RTI receiving an LTC message, the federate associated with e_1 is the same as the federate associated with e_2 , and the logical tag of e_1 is less than the logical tag of e_2 .
- `((e_1 is (Receiving LTC)) \wedge (Federate(e_1) = Federate(e_2)) \wedge ((Tag e_1) < (Tag e_2)) \wedge (e_2 is (Receiving LTC)))` : This sub-formula is true when the previous sub-formula is true and event e_2 also involves the RTI receiving an LTC message.

3. Conclusion:

- The antecedent of the implication is true when the larger sub-formula `((e_1 is (Receiving LTC)) \wedge (Federate(e_1) = Federate(e_2)) \wedge ((Tag e_1) < (Tag e_2)) \wedge (e_2 is (Receiving LTC)))` is true. This means that both events e_1 and e_2 are LTC messages

received by the same federate, and the logical tag of e_1 is less than the logical tag of e_2 .

By understanding when each sub-formula is true, we can see the conditions under which the antecedent of the implication holds, leading to the conclusion that event e_1 must precede event e_2 based on the given formula.

(This explanation was generated in 12 seconds.)

Summary of the meaning of formula 1

The formula states that if event e_1 involves the RTI receiving an LTC message, the federate associated with e_1 is the same as the federate associated with e_2 , and the logical tag of e_1 is less than the logical tag of e_2 , and event e_2 also involves the RTI receiving an LTC message, then event e_1 must precede event e_2 in physical time.

In simpler terms, the formula ensures that if two LTC messages are received by the same federate, with the first LTC message having a lower logical tag than the second LTC message, then the first LTC message must occur before the second LTC message in physical time. This ordering constraint is crucial for maintaining the logical consistency and proper sequencing of events within the federated program.

The formula guarantees this ordering relationship for all possible events e_1 and e_2 , providing a logical rule that must be upheld to prevent inconsistencies in the execution of the federated program.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee that an LTC (Logical Tag Complete) message with a lower logical tag from a federate precedes an LTC message with a higher logical tag from the same federate in physical time is a direct consequence of how federates and the RTI manage logical time. Federates execute events in logical time order, completing all activities associated with a given logical time before moving to a higher logical time. An LTC message signifies the completion of all activities at a specific logical time. Therefore, for a federate to send an LTC message for a higher logical time, it must have already completed and sent an LTC message for any lower logical time. This sequential progression ensures that LTC messages for increasing logical times are sent in order,

reflecting the real-world, physical time ordering of these messages as they are received by the RTI. This ordering is essential for maintaining the consistency of the simulation's logical timeline and preventing temporal anomalies.

(This explanation was generated in 14 seconds.)

Formula 2

Formula 2 states:

```
((e1 is the first event satisfying ( $\lambda$  e1 .
  ((e1 is (Sending STOP_GRN))
    $\vee$  ((e1 is (Receiving LTC))
         $\vee$  (((e1 is (Receiving NET))  $\vee$  ((e1 is (Sending TAGGED_MSG))))))
    $\wedge$  ((Tag e1) = (Tag e2))
    $\wedge$  (((Tag e1) finite)  $\wedge$  ((Tag e1)  $\neq$  0))))
  $\wedge$  (((e2 is (Sending TAG))  $\vee$  ((e2 is (Sending PTAG)))))
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then move on to larger sub-formulas constructed from them.

1. Atomic Sub-formulas:

- `(e_1 is (Sending STOP_GRN))` : This atomic sub-formula is true when event e_1 corresponds to the RTI sending a STOP_GRN message.
- `(e_1 is (Receiving LTC))` : This atomic sub-formula is true when event e_1 corresponds to the RTI receiving an LTC message.
- `(e_1 is (Receiving NET))` : This atomic sub-formula is true when event e_1 corresponds to the RTI receiving a NET message.
- `(e_1 is (Sending TAGGED_MSG))` : This atomic sub-formula is true when event e_1 corresponds to the RTI sending a TAGGED_MSG message.
- `(Tag e_1) = (Tag e_2)` : This atomic sub-formula is true when the

tags of events e_1 and e_2 are equal.

- $((\text{Tag } e_1) \text{ finite}) \wedge ((\text{Tag } e_1) \neq 0)$: This atomic sub-formula is true when the tag of event e_1 is a finite value and not equal to zero.
- $(e_2 \text{ is (Sending TAG)})$: This atomic sub-formula is true when event e_2 corresponds to the RTI sending a TAG message.
- $(e_2 \text{ is (Sending PTAG)})$: This atomic sub-formula is true when event e_2 corresponds to the RTI sending a PTAG message.

2. Larger Sub-formulas:

- $(((e_1 \text{ is (Sending STOP_GRN)}) \vee (e_1 \text{ is (Receiving LTC)}) \vee (e_1 \text{ is (Receiving NET)})) \wedge ((\text{Tag } e_1) = (\text{Tag } e_2)))$: This sub-formula is true when event e_1 involves either sending a STOP_GRN message, receiving an LTC message, receiving a NET message, or sending a TAGGED_MSG message, and the tags of e_1 and e_2 are equal.
- $(((\text{Tag } e_1) \text{ finite}) \wedge ((\text{Tag } e_1) \neq 0))$: This sub-formula is true when the tag of event e_1 is a finite value and not equal to zero.
- $(((e_2 \text{ is (Sending TAG)}) \vee (e_2 \text{ is (Sending PTAG)})))$: This sub-formula is true when event e_2 involves either sending a TAG message or sending a PTAG message.

3. Analysis of Larger Sub-formulas:

- The conjunction of the first two larger sub-formulas
 $((((e_1 \text{ is (Sending STOP_GRN)}) \vee (e_1 \text{ is (Receiving LTC)}) \vee (e_1 \text{ is (Receiving NET)})) \wedge ((\text{Tag } e_1) = (\text{Tag } e_2))) \wedge (((\text{Tag } e_1) \text{ finite}) \wedge ((\text{Tag } e_1) \neq 0)))$
is true when event e_1 satisfies one of the specified conditions and has the same tag as event e_2 .
- The conjunction of the above sub-formula with the third larger sub-formula
 $((((e_1 \text{ is (Sending STOP_GRN)}) \vee (e_1 \text{ is (Receiving LTC)}) \vee (e_1 \text{ is (Receiving NET)})) \wedge ((\text{Tag } e_1) = (\text{Tag } e_2))) \wedge (((\text{Tag } e_1) \text{ finite}) \wedge ((\text{Tag } e_1) \neq 0))) \wedge (((e_2 \text{ is (Sending TAG)}) \vee (e_2 \text{ is (Sending PTAG)})))$
is true when event e_1 satisfies the specified conditions and has a finite tag not equal to zero, and event e_2 involves sending a TAG message or a PTAG message.

By analyzing the atomic and larger sub-formulas, we can understand the conditions under which the antecedent of the logical implication is true.

(This explanation was generated in 13 seconds.)

Summary of the meaning of formula 2

The given formula states that if event e_1 in the RTI satisfies certain conditions related to sending or receiving specific types of messages (STOP_GRN, LTC, NET, TAGGED_MSG) and shares the same tag with event e_2 , and if the tag of event e_1 is a finite value not equal to zero, then event e_2 in the RTI must involve sending either a TAG message or a PTAG message. This implication holds true for all possible events e_1 and e_2 in the RTI.

In simpler terms, the formula ensures that when the RTI is involved in sending or receiving certain types of messages with specific tag constraints, there is a subsequent event where the RTI sends a message that allows logical time to advance. This logical structure guarantees the proper sequencing and progression of events within the RTI system, preventing any violations or inconsistencies in the flow of messages and time advancement.

(This explanation was generated in 3 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the fundamental operational rules of the RTI and federates regarding message handling and logical time progression. Specifically, when the RTI receives or sends messages related to event completion (LTC), event scheduling (NET), or signal transmission (TAGGED_MSG), it indicates a point at which federates have reached or are planning for a certain logical time. The sending of TAG (Tag Advance Grant) or PTAG (Provisional Tag Advance Grant) messages by the RTI following these events ensures that federates are correctly synchronized and can safely advance to new logical times without violating the causality and consistency required by the simulation. This mechanism prevents STP (safe-to-process) violations and deadlocks by ensuring that federates only proceed to logical times when they are allowed to, based on the information processed by the RTI, thus maintaining the integrity of the distributed simulation.

(This explanation was generated in 7 seconds.)

Formula 3

Formula 3 states:

```
(((((e1 is (Receiving PORT_ABS))) v ((e1 is (Receiving TAGGED_MSG))))  
  ∧ (Federate (e1) = Federate (e2))  
  ∧ ((Tag e1) ≤ (Tag e2)))  
  ∧ (((e2 is (Receiving LTC)))))  
⇒ (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then move on to larger sub-formulas constructed from them.

1. Atomic Sub-formulas:

- $(e_1 \text{ is (Receiving PORT_ABS)})$: This sub-formula is true when event e_1 corresponds to the RTI receiving a PORT_ABS message.
- $(e_1 \text{ is (Receiving TAGGED_MSG)})$: This sub-formula is true when event e_1 corresponds to the RTI receiving a TAGGED_MSG message.
- $(\text{Federate}(e_1) = \text{Federate}(e_2))$: This sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- $((\text{Tag } e_1) \leq (\text{Tag } e_2))$: This sub-formula is true when the logical time tag of event e_1 is less than or equal to the logical time tag of event e_2 .
- $(e_2 \text{ is (Receiving LTC)})$: This sub-formula is true when event e_2 corresponds to the RTI receiving a LTC message.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Receiving PORT_ABS)}) v (e_1 \text{ is (Receiving TAGGED_MSG)}))$: This sub-formula is true when event e_1 corresponds to the RTI receiving either a PORT_ABS message or a TAGGED_MSG message.
- $((((e_1 \text{ is (Receiving PORT_ABS)}) v (e_1 \text{ is (Receiving TAGGED_MSG)})) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)))) \wedge (e_2 \text{ is (Receiving LTC)})$: This sub-formula is true when event e_1 satisfies the conditions of receiving a PORT_ABS or TAGGED_MSG message, having the same

federate as event e_2 , and having a logical time tag less than or equal to event e_2 .

- $((e_1 \text{ is (Receiving PORT_ABS)}) \vee (e_1 \text{ is (Receiving TAGGED_MSG)})) \wedge$
: This sub-formula is true when the previous sub-formula is satisfied, and event e_2 corresponds to the RTI receiving a LTC message.

3. Conclusion:

- The entire antecedent of the logical implication is true when the above sub-formula is satisfied. This implies that if events e_1 and e_2 satisfy the conditions outlined in the antecedent, then event e_1 must occur before event e_2 in physical time, denoted by the implication $e_1 \prec e_2$.

By breaking down the antecedent into its sub-formulas and analyzing when each sub-formula is true, we can understand the conditions under which the logical implication holds true in the context of the RTI events and their relationships.

(This explanation was generated in 11 seconds.)

Summary of the meaning of formula 3

The formula states that if event e_1 in the RTI involves receiving a PORT_ABS or TAGGED_MSG message from a federate, and the federate associated with e_1 is the same as the federate associated with event e_2 , and the logical time tag of e_1 is less than or equal to the logical time tag of e_2 , and event e_2 involves receiving a LTC message, then event e_1 must occur before event e_2 in physical time.

In simpler terms, the formula ensures that when the RTI receives a message (PORT_ABS or TAGGED_MSG) from a federate, and the logical time of that message is before or at the same time as another event where the RTI receives a LTC message, then the first event must occur before the second event in physical time. This condition helps maintain the correct order of events in the federated system to prevent any violations or conflicts that could arise if events were processed out of order.

The formula guarantees that this relationship holds true for all possible events e_1 and e_2 in the RTI, ensuring the proper sequencing and coordination of messages and actions within the federated system.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the fundamental principles of managing logical time in a distributed simulation system. The RTI serves as the coordinator for ensuring that all federates proceed through logical time in a consistent and orderly manner. When a federate sends a PORT_ABS or TAGGED_MSG message, it is communicating about the absence or presence of a signal at a specific logical time. The LTC (Logical Tag Complete) message indicates that a federate has completed all processing for a given logical time. For the system to maintain consistency, any messages about signals (PORT_ABS or TAGGED_MSG) for a given logical time must be processed before the system can acknowledge that the processing for that time is complete (LTC). This ensures that all relevant information for a logical time is considered before moving past that point in logical time, thereby preventing any causality violations or inconsistencies in the simulation's logical timeline.

(This explanation was generated in 11 seconds.)

Formula 4

Formula 4 states:

```
(((((e1 is (Receiving NET))) ∧ (Federate (e1) = Federate (e2))) ∧ ((Tag e1) ≤ (Tag e2)))  
  ∧ (((e2 is (Receiving LTC))) ∧ ((Tag e2) ≠ 0)))  
⇒ (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Receiving NET))` : This atomic sub-formula is true when event e_1 in the RTI involves receiving a NET message.
- `(Federate(e_1) = Federate(e_2))` : This atomic sub-formula is true when the federate associated with event e_1 is the same as the federate

associated with event e_2 .

- $((\text{Tag } e_1) \leq (\text{Tag } e_2))$: This atomic sub-formula is true when the tag of event e_1 is less than or equal to the tag of event e_2 .
- $(e_2 \text{ is (Receiving LTC)})$: This atomic sub-formula is true when event e_2 in the RTI involves receiving a LTC message.
- $((\text{Tag } e_2) \neq 0)$: This atomic sub-formula is true when the tag of event e_2 is not equal to 0.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Receiving NET)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)))$: This sub-formula is true when event e_1 involves receiving a NET message, the federate associated with e_1 is the same as the federate associated with e_2 , and the tag of e_1 is less than or equal to the tag of e_2 .
- $((e_2 \text{ is (Receiving LTC)}) \wedge ((\text{Tag } e_2) \neq 0))$: This sub-formula is true when event e_2 involves receiving a LTC message and the tag of e_2 is not equal to 0.

3. Largest Sub-formula:

- $((e_1 \text{ is (Receiving NET)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)) \wedge (e_2 \text{ is (Receiving LTC)}) \wedge ((\text{Tag } e_2) \neq 0))$: This largest sub-formula is true when both the sub-formula related to event e_1 and the sub-formula related to event e_2 are true simultaneously.

Therefore, the breakdown of the antecedent of the logical implication into its sub-formulas allows us to understand the conditions under which each sub-formula is true, leading to the overall truth of the logical implication.

(This explanation was generated in 12 seconds.)

Summary of the meaning of formula 4

The formula states that if event e_1 in the RTI involves receiving a NET message from a federate, and the federate associated with e_1 is the same as the federate associated with event e_2 , and the tag of e_1 is less than or equal to the tag of e_2 , and event e_2 in the RTI involves receiving a LTC message with a tag not equal to 0, then event e_1 cannot occur after event e_2 in physical time.

In simpler terms, the formula ensures that when a federate declares the time of its next event to

the RTI (NET message) and the RTI receives this information, and then the federate declares that it has finished executing a particular tag (LTC message), the order of these events in logical time must be preserved such that the NET message is received before the LTC message. This guarantees the correct sequencing of events within the federated program to prevent inconsistencies or violations.

The formula captures a specific constraint within the RTI's communication and coordination process, ensuring that the logical order of events related to time declarations and completion notifications is maintained, ultimately contributing to the overall integrity and reliability of the federated system's operation.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the fundamental rules governing the interactions between federates and the RTI in a distributed simulation system. When a federate sends a NET message, it is declaring the next scheduled event's logical time, indicating its future state. The LTC message, on the other hand, signifies the completion of processing for a specific logical time. For the system to maintain consistency and prevent logical time violations, the RTI must receive and process these messages in the order they logically occur: the declaration of a future event (NET) must logically precede the completion of an event (LTC). This ordering ensures that the federate's state transitions and event processing adhere to the logical time progression, maintaining the integrity of the simulation's temporal logic.

(This explanation was generated in 8 seconds.)

Formula 5

Formula 5 states:

```

((((e1 is (Receiving LTC)))
  ∧ (Federate (e1) = Federate (e2))
  ∧ ((Tag e1) + (largest delay of a connection going out of the federate of
    e1) < (Tag e2)))
  ∧ (((e2 is (Receiving PORT_ABS))) ∨ ((e2 is (Receiving TAGGED_MSG)))))
⇒ (e1 < e2)

```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Receiving LTC))` : This sub-formula is true when event e_1 corresponds to the RTI receiving a Logical Tag Complete (LTC) message from a federate.
- `(Federate(e_1) = Federate(e_2))` : This sub-formula is true when events e_1 and e_2 occur in the same federate.
- `((Tag e_1) + (largest delay of a connection going out of the federate of e_1) < (Tag e_2))` : This sub-formula is true when the tag of event e_1 plus the largest delay of a connection going out of the federate of event e_1 is less than the tag of event e_2 .
- `(e_2 is (Receiving PORT_ABS))` : This sub-formula is true when event e_2 corresponds to the RTI receiving a PORT_ABS message.
- `(e_2 is (Receiving TAGGED_MSG))` : This sub-formula is true when event e_2 corresponds to the RTI receiving a TAGGED_MSG message.

2. Larger Sub-formulas:

- `((e_1 is (Receiving LTC)) ∧ (Federate(e_1) = Federate(e_2)) ∧ ((Tag e_1) + (largest delay of a connection going out of the federate of e_1) < (Tag e_2)))` : This sub-formula is true when event e_1 involves the RTI receiving an LTC message, events e_1 and e_2 occur in the same federate, and the tag of event e_1 plus the largest delay of a connection going out of the federate of event e_1 is less than the tag of event e_2 .
- `((e_2 is (Receiving PORT_ABS)) ∨ (e_2 is (Receiving TAGGED_MSG)))` : This sub-formula is true when event e_2 corresponds to the RTI receiving either a PORT_ABS message or a TAGGED_MSG message.

3. Largest Sub-formula:

- The antecedent of the implication is the conjunction of the two larger sub-formulas above:

$$(((e_1 \text{ is (Receiving LTC)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2))) \wedge ((\text{Tag } e_1 + \text{largest_delay} < \text{Tag } e_2)))$$

This breakdown helps in understanding the conditions under which the antecedent of the implication is true, leading to the conclusion that event e_1 must precede event e_2 in physical time.

(This explanation was generated in 10 seconds.)

Summary of the meaning of formula 5

The formula states that if event e_1 in the RTI involves receiving a Logical Tag Complete (LTC) message from a federate, and the tag of e_1 plus the largest delay of a connection going out of the federate of e_1 is less than the tag of event e_2 , and both e_1 and e_2 occur in the same federate, then event e_2 in the RTI must involve either receiving a PORT_ABS message or a TAGGED_MSG message.

This implies that when the RTI receives an LTC message from a federate and the conditions regarding tags and delays are met, the subsequent event in the same federate must involve receiving a message that indicates either the absence of a signal (PORT_ABS) or the presence of a signal (TAGGED_MSG) from another federate. This ensures proper synchronization and communication between federates to prevent violations such as executing something too early (STP violation) and to allow for the orderly progression of events in the federated system.

The formula guarantees that this logical implication holds true for all possible events e_1 and e_2 in the RTI, ensuring the correct sequencing and coordination of actions within the federated system without the risk of violating safety or deadlock constraints.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the operational rules and objectives of the RTI and federates in managing logical time and ensuring message synchronization. When the RTI receives a Logical Tag Complete (LTC) message, it signifies that a federate has completed all actions up to a certain logical time. The condition involving the tag of e_1 plus the largest delay indicates the earliest logical time a future message could affect another

federate. Therefore, for the RTI to receive either a PORT_ABS or a TAGGED_MSG from the same federate after e_1 under these conditions ensures that federates only proceed to new logical times when it's safe—i.e., they have accounted for all potential incoming messages up to that point. This mechanism prevents STP violations and deadlocks by ensuring federates do not advance to a logical time where they could be affected by unprocessed messages, thus maintaining the integrity and order of the distributed simulation.

(This explanation was generated in 9 seconds.)

Formula 6

Formula 6 states:

```
((e1 is the first event satisfying ( $\lambda$  e1 . (((e1 is (Sending TAG)))  $\vee$  ((e1 is (Sending PORT_ABS))
 $\wedge$  (Federate (e1) = Federate (e2))
 $\wedge$  ((Tag e1) + (largest delay of a connection going out of the federate
of e1)  $\geq$  (Tag e2)))))
 $\wedge$ 
(((e2 is (Receiving PORT_ABS)))  $\vee$  ((e2 is (Receiving TAGGED_MSG))))
 $\wedge$  ( $\neg$  ((Fed e2) has no upstream with delay  $\leq$  (Tag e2)))))
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Sending TAG))` : This atomic sub-formula is true when event e_1 involves the RTI sending a TAG message.
- `(e_1 is (Sending PORT_ABS))` : This atomic sub-formula is true when event e_1 involves the RTI sending a PORT_ABS message.
- `Federate(e_1) = Federate(e_2)` : This atomic sub-formula is true when events e_1 and e_2 occur in the same federate.
- `(Tag e_1) + (largest delay of a connection going out of the federate of e_1) \geq (Tag e_2)` : This atomic sub-formula is true when the logical time of e_1 plus the largest delay of a connection going out of the federate of e_1 is greater than or equal to the logical time of e_2 .

: This atomic sub-formula is true when the sum of the tag of event e_1 and the largest delay of a connection going out of the federate of e_1 is greater than or equal to the tag of event e_2 .

- $(e_2 \text{ is (Receiving PORT_ABS)})$: This atomic sub-formula is true when event e_2 involves the RTI receiving a PORT_ABS message.
- $(e_2 \text{ is (Receiving TAGGED_MSG)})$: This atomic sub-formula is true when event e_2 involves the RTI receiving a TAGGED_MSG message.
- $(\text{Fed } e_2) \text{ has no upstream with delay } \leq (\text{Tag } e_2)$: This atomic sub-formula is true when the federate associated with event e_2 has no upstream connection with a delay less than or equal to the tag of event e_2 .

2. Larger Sub-formulas:

- $((e_1 \text{ is (Sending TAG)}) \vee (e_1 \text{ is (Sending PTAG)}))$: This sub-formula is true when event e_1 involves the RTI sending either a TAG or a PTAG message.
- $((e_1 \text{ is the first event satisfying } (\dots) \wedge (\dots) \wedge (\dots)))$: This sub-formula is true when event e_1 satisfies the conditions specified within the parentheses, including sending a TAG or PTAG message, being in the same federate as event e_2 , and having a tag that allows for the necessary delay.
- $((e_2 \text{ is (Receiving PORT_ABS)}) \vee (e_2 \text{ is (Receiving TAGGED_MSG)}))$: This sub-formula is true when event e_2 involves the RTI receiving either a PORT_ABS or a TAGGED_MSG message.
- $(\neg((\text{Fed } e_2) \text{ has no upstream with delay } \leq (\text{Tag } e_2)))$: This sub-formula is true when the federate associated with event e_2 has at least one upstream connection with a delay less than or equal to the tag of event e_2 .

3. Largest Sub-formula:

- The antecedent of the logical implication is the largest sub-formula, combining all the atomic and larger sub-formulas using logical operators like \wedge and \vee .

By breaking down the antecedent into its constituent sub-formulas and understanding when each sub-formula is true, we can analyze the conditions under which the entire antecedent holds true, leading to the implication that e_1 precedes e_2 .

(This explanation was generated in 10 seconds.)

Summary of the meaning of formula 6

The formula states that if event e_1 in the RTI involves sending either a TAG or a PTAG message to the same federate as event e_2 , with a tag that accounts for the largest delay of a connection going out of the federate of e_1 , and event e_2 in the RTI involves receiving a PORT_ABS or a TAGGED_MSG message, and the federate associated with e_2 has at least one upstream connection with a delay less than or equal to the tag of e_2 , then event e_1 must precede event e_2 in logical time.

In simpler terms, this formula ensures that when the RTI sends a message to a federate that needs to be received before a certain tag, and that federate has the necessary information to process the message without violating any timing constraints, the sending event must occur before the receiving event to maintain the logical order of events in the system. This logical implication holds true for all possible events e_1 and e_2 in the RTI, guaranteeing the correct sequencing of message exchanges between the RTI and federates.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the fundamental principles of maintaining logical consistency and causality within a distributed simulation system. When the RTI sends a TAG or PTAG message to a federate, it is essentially granting permission for the federate to advance to a new logical time, taking into account the maximum delay from its connections. For the system to remain consistent, any incoming messages (PORT_ABS or TAGGED_MSG) that could affect the state of the federate at this new logical time must be processed before the federate advances. This ensures that the federate's state reflects all relevant events up to that point in logical time. The ordering guarantee ensures that messages are processed in a sequence that respects the logical time constraints, thereby preventing causality violations and ensuring the integrity of the simulation.

(This explanation was generated in 16 seconds.)

Formula 7

Formula 7 states:

```
(((((e1 is (Sending PTAG))) v ((e1 is (Sending TAG))))  
  ∧ (Federate (e1) = Federate (e2))  
  ∧ ((Tag e1) < (Tag e2)))  
  ∧ (((e2 is (Sending PTAG))) v ((e2 is (Sending TAG)))))  
⇒ (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas:

1. Atomic Sub-formulas:

- `(e1 is (Sending PTAG))` : This atomic sub-formula is true when event e_1 corresponds to the RTI sending a PTAG message.
- `(e1 is (Sending TAG))` : This atomic sub-formula is true when event e_1 corresponds to the RTI sending a TAG message.
- `Federate(e1) = Federate(e2)` : This atomic sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- `(Tag e1) < (Tag e2)` : This atomic sub-formula is true when the tag associated with event e_1 is less than the tag associated with event e_2 .

2. Larger Sub-formulas:

- `((e1 is (Sending PTAG)) v (e1 is (Sending TAG)))` : This sub-formula is true when event e_1 corresponds to the RTI sending either a PTAG or a TAG message.
- `((((e1 is (Sending PTAG)) v (e1 is (Sending TAG))) ∧ (Federate(e1) = Federate(e2))) ∧ ((Tag e1) < (Tag e2)))` : This sub-formula is true when event e_1 corresponds to the RTI sending a PTAG or TAG message, the federate associated with e_1 is the same as the federate associated with e_2 , and the tag of e_1 is less than the tag of e_2 .

3. Largest Sub-formula:

- $((((e_1 \text{ is (Sending PTAG)}) \vee (e_1 \text{ is (Sending TAG)})) \wedge (\text{Federate}(e_1)))$
: This largest sub-formula is true when both event e_1 and event e_2 satisfy the conditions specified in the inner sub-formula, i.e., e_1 involves sending a PTAG or TAG message, e_1 and e_2 are associated with the same federate, the tag of e_1 is less than the tag of e_2 , and e_2 involves sending a PTAG or TAG message.

By breaking down the antecedent of the logical implication into its sub-formulas and analyzing when each sub-formula is true, we can understand the conditions under which the entire antecedent is true, leading to the implication $e_1 \prec e_2$.

(This explanation was generated in 9 seconds.)

Summary of the meaning of formula 7

The formula provided states that if event e_1 in the RTI involves sending a PTAG or TAG message to the same federate as event e_2 , and the tag associated with e_1 is less than the tag associated with e_2 , and event e_2 also involves sending a PTAG or TAG message, then event e_1 must occur before event e_2 in logical time.

In simpler terms, the formula ensures that if the RTI sends a PTAG or TAG message to a federate, and then later sends another PTAG or TAG message to the same federate with a higher tag value, the first message must have been sent before the second message. This logical relationship is crucial for maintaining the order of message transmissions and ensuring the correct progression of events within the federated system.

The formula captures a specific sequencing constraint within the communication protocol of the federated system, emphasizing the importance of maintaining the order of message transmissions between the RTI and federates. By guaranteeing the truth of this formula for all possible events e_1 and e_2 , the system ensures the integrity and consistency of the communication and event processing mechanisms.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee is expected to be correct because the RTI manages the logical time progression

within the federated system, ensuring that federates only proceed to execute events at or before the granted logical times to maintain causality and prevent STP (safe-to-process) violations. PTAG (Provisional Tag Advance Grant) and TAG (Tag Advance Grant) messages are critical for this, as they inform federates of the logical times up to which they are allowed to advance. Since these messages dictate the permissible logical time advancements for federates, the RTI must send them in ascending order of their associated tags to each federate. This ensures that a federate does not receive permission to advance to a later logical time before being granted permission for an earlier time, maintaining the integrity of the simulation's temporal ordering and preventing any logical inconsistencies or causality errors within the distributed simulation environment.

(This explanation was generated in 14 seconds.)

Formula 8

Formula 8 states:

```
((((e1 is (Sending PTAG))) ∧ (Federate (e1) = Federate (e2)) ∧ ((Tag e1) ≤ (Tag e2)))
 ∧ ((e2 is (Sending TAG))))
⇒ (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Sending PTAG))` : This sub-formula is true when event e_1 is an event in which the RTI sends a PTAG message.
- `(Federate(e_1) = Federate(e_2))` : This sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- `((Tag e_1) ≤ (Tag e_2))` : This sub-formula is true when the tag of event e_1 is less than or equal to the tag of event e_2 .
- `(e_2 is (Sending TAG))` : This sub-formula is true when event e_2 is

an event in which the RTI sends a TAG message.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Sending PTAG)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)))$
: This sub-formula is true when all three atomic sub-formulas are true simultaneously. It represents the conjunction of the RTI sending a PTAG message, both events occurring in the same federate, and the tag of e_1 being less than or equal to the tag of e_2 .
- $((e_1 \text{ is (Sending PTAG)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)) \wedge (e_2 \text{ is (Sending TAG)}))$
: This sub-formula is true when the previous sub-formula is true and event e_2 is an event in which the RTI sends a TAG message.

3. Analysis:

- The larger sub-formula
 $((e_1 \text{ is (Sending PTAG)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)))$
is true when the RTI sends a PTAG message, both events occur in the same federate, and the tag of e_1 is less than or equal to the tag of e_2 . This condition ensures that the PTAG message is sent before the TAG message.
- The sub-formula
 $((e_1 \text{ is (Sending PTAG)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)) \wedge (e_2 \text{ is (Sending TAG)}))$
further refines the condition by requiring that the TAG message is sent after the PTAG message.

Therefore, the antecedent of the implication is true when the RTI sends a PTAG message, both events occur in the same federate, the tag of e_1 is less than or equal to the tag of e_2 , and then the RTI sends a TAG message. This sequence of events ensures that the PTAG message is sent before the TAG message, leading to the implication that event e_1 precedes event e_2 in physical time.

(This explanation was generated in 14 seconds.)

Summary of the meaning of formula 8

The formula states that if event e_1 in the RTI sends a PTAG message to a federate and the tag of e_1 is less than or equal to the tag of another event e_2 in the same federate where e_2 sends a TAG message, then event e_1 precedes event e_2 in physical time. This implies that the PTAG message must be sent before the TAG message in the same federate to maintain the temporal order of

events.

In essence, the formula ensures a specific ordering constraint between the sending of a PTAG message and a TAG message within the same federate. It guarantees that the PTAG message, which grants permission to advance in logical time, is always sent before the TAG message, which actually advances the logical time. This sequential ordering is crucial to maintain the logical consistency and temporal integrity of the events within the federate.

By proving that this formula holds true for all possible combinations of events e_1 and e_2 , we establish a fundamental rule governing the temporal relationship between these specific types of messages sent by the RTI within a federate.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is correct because it aligns with the operational logic of the RTI and federates concerning time management and message processing. A PTAG (Provisional Tag Advance Grant) message is sent by the RTI to a federate to indicate that the federate may proceed up to, but not including, a specified logical time. This message is provisional and does not commit the federate to a new logical time. The TAG (Tag Advance Grant) message, on the other hand, authorizes the federate to advance to and include a specific logical time, finalizing the time advancement. The ordering guarantee ensures that a federate receives permission to tentatively advance in logical time (PTAG) before receiving final authorization to commit to that advancement (TAG). This sequence is necessary to maintain the integrity of logical time progression within the simulation, ensuring that federates process events in an order that respects both the logical temporal order and the dependencies between events.

(This explanation was generated in 10 seconds.)

Formula 9

Formula 9 states:

```

(((e1 is the first event in a given federate satisfying ( $\lambda$  e1 .
  (((e1 is (Receiving LTC)))
     $\wedge$  ((Federate of e1 is upstream of federate of e2 via a zero-delay
      connection))
     $\wedge$  ((Tag e1)  $\geq$  (Tag e2)))
  v
  (((e1 is (Sending TAG)))
    v ((e1 is (Receiving NET)))
    v ((e1 is (Sending STOP_GRN))))
     $\wedge$  ((Federate of e1 is upstream of federate of e2 via a zero-delay
      connection))
     $\wedge$  ((Tag e1)  $\geq$  (Tag e2))))))
 $\wedge$  (((e2 is (Sending TAG)))  $\wedge$  ((Tag e2)  $\neq$  0)))
 $\Rightarrow$  (e1 < e2)

```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- **(e₁ is the first event in a given federate satisfying (...))**
: This sub-formula is true when event e_1 is the first event in a given federate that satisfies the condition specified within the parentheses. This condition involves a conjunction of sub-formulas related to the type of event e_1 and its properties.
- **(e₁ is (Receiving LTC))** : This sub-formula is true when event e_1 involves the RTI receiving a Logical Tag Complete message.
- **(Federate of e₁ is upstream of federate of e₂ via a zero-delay connection)**
: This sub-formula is true when the federate associated with event e_1 is upstream of the federate associated with event e_2 and there is a zero-delay connection between them.
- **(Tag e₁) \geq (Tag e₂)** : This sub-formula is true when the logical tag associated with event e_1 is greater than or equal to the logical tag associated with event e_2 .
- **(e₁ is (Sending TAG))** : This sub-formula is true when event e_1

involves the RTI sending a TAG message.

- $(e_1 \text{ is (Receiving NET)})$: This sub-formula is true when event e_1 involves the RTI receiving a NET message.
- $(e_1 \text{ is (Sending STOP_GRN)})$: This sub-formula is true when event e_1 involves the RTI sending a STOP_GRN message.
- $(e_2 \text{ is (Sending TAG)})$: This sub-formula is true when event e_2 involves the RTI sending a TAG message.
- $(\text{Tag } e_2) \neq 0$: This sub-formula is true when the logical tag associated with event e_2 is not equal to zero.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Receiving LTC)}) \wedge (\dots) \vee (\dots) \vee (\dots)) \wedge (\dots)$: This sub-formula is true when event e_1 satisfies one of the specified conditions related to LTC, TAG, NET, or STOP_GRN messages, and the federate of event e_1 is upstream of the federate of event e_2 via a zero-delay connection, and the logical tag of event e_1 is greater than or equal to the logical tag of event e_2 .
- $((e_1 \text{ is (Receiving LTC)}) \wedge (\dots) \vee (\dots) \vee (\dots)) \wedge (\dots) \wedge (e_2 \text{ is (Sending TAG)})$: This sub-formula is true when the previous sub-formula holds for event e_1 , and event e_2 involves the RTI sending a TAG message with a non-zero logical tag.

3. Conclusion:

- The antecedent of the logical implication is true when the larger sub-formula described above holds for events e_1 and e_2 . This indicates specific conditions related to the types of events, their order, and the properties of the federates and connections involved.

By breaking down the antecedent into its constituent sub-formulas and analyzing when each sub-formula is true, we can understand the conditions under which the entire antecedent is satisfied.

(This explanation was generated in 11 seconds.)

Summary of the meaning of formula 9

The formula given states a logical implication that must hold true for all events e_1 and e_2 in the system. The implication can be broken down into the following key components:

1. The first event e_1 in a given federate that satisfies a specific condition must either:
 - Involve the RTI receiving a Logical Tag Complete (LTC) message from an upstream federate via a zero-delay connection with a tag greater than or equal to the tag associated with event e_2 .
 - Or, it must involve the RTI sending a TAG message, receiving a NET message, or sending a STOP_GRN message to an upstream federate via a zero-delay connection with a tag greater than or equal to the tag associated with event e_2 .
2. The second event e_2 must involve the RTI sending a TAG message with a non-zero tag.

The formula essentially ensures that if event e_1 satisfies certain conditions related to LTC, TAG, NET, or STOP_GRN messages and their associated tags concerning upstream federates and zero-delay connections, and event e_2 involves the RTI sending a TAG message with a non-zero tag, then event e_1 must logically precede event e_2 in the system.

This logical implication guarantees the correct order of events and message exchanges between the RTI and federates, ensuring proper coordination and synchronization within the federated system to prevent violations such as STP violations and deadlocks.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the fundamental principles governing the interactions between the RTI and federates, particularly in ensuring logical consistency and preventing STP violations. The RTI's role in managing logical time advancement and ensuring messages are processed in the correct order necessitates that any event involving the RTI sending a TAG message with a non-zero tag (indicating permission for a federate to advance to a specific logical time) must logically follow any relevant upstream events that could affect the state or decisions at that logical time. This includes the processing of LTC messages (indicating completion of activities at a certain tag), handling of NET messages (indicating the next scheduled event), or STOP_GRN messages (indicating shutdown procedures), especially over zero-delay connections that imply immediate logical effect. This ordering ensures that all dependencies and potential impacts are accounted for before a federate is allowed to proceed, thus maintaining the integrity of the simulation's logical timeline and preventing premature progression that could lead to inconsistencies or errors in the simulation's execution.

(This explanation was generated in 14 seconds.)

Formula 10

Formula 10 states:

```
((e1 is the first event satisfying ( $\lambda$  e1 .
  ((e1 is (Sending PTAG)))
   $\wedge$  ((Federate of e1 is upstream of federate of e2 via a zero-delay
    connection))
   $\wedge$  ((Tag e1) = (Tag e2)))
 $\vee$ 
  (((e1 is (Receiving NET)))  $\vee$  ((e1 is (Sending STOP_GRN))))
   $\wedge$ 
  ((Federate (e1) = Federate (e2))
     $\vee$  ((Federate of e1 is directly upstream of federate of e2)))
   $\wedge$  ((Tag e1) = (Tag e2))))
 $\wedge$  ((e2 is (Sending PTAG)))  $\wedge$  ((Tag e2)  $\neq$  0))
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Sending PTAG))` : This sub-formula is true when event e_1 corresponds to the RTI sending a PTAG message.
- `(Federate of e_1 is upstream of federate of e_2 via a zero-delay connection)` : This sub-formula is true when the federate of event e_1 is upstream of the federate of event e_2 via a zero-delay connection.
- `(Tag e_1) = (Tag e_2)` : This sub-formula is true when the tags of events e_1 and e_2 are equal.
- `(e_1 is (Receiving NET))` : This sub-formula is true when event e_1 corresponds to the RTI receiving a NET message.

- $(e_1 \text{ is (Sending STOP_GRN)})$: This sub-formula is true when event e_1 corresponds to the RTI sending a STOP_GRN message.
- $(\text{Federate}(e_1) = \text{Federate}(e_2))$: This sub-formula is true when the federates of events e_1 and e_2 are the same.
- $(\text{Federate of } e_1 \text{ is directly upstream of federate of } e_2)$: This sub-formula is true when the federate of event e_1 is directly upstream of the federate of event e_2 .
- $(\text{Tag } e_1) = (\text{Tag } e_2)$: This sub-formula is true when the tags of events e_1 and e_2 are equal and non-zero.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Sending PTAG)}) \wedge ((\text{Federate of } e_1 \text{ is upstream of federate of } e_2)))$: This sub-formula is true when event e_1 is the RTI sending a PTAG message to a federate upstream of the federate of event e_2 via a zero-delay connection, and the tags of events e_1 and e_2 are equal.
- $((e_1 \text{ is (Receiving NET)}) \vee (e_1 \text{ is (Sending STOP_GRN)})) \wedge ((\text{Federate of } e_1 = \text{Federate of } e_2) \vee (\text{Federate of } e_1 \text{ is directly upstream of federate of } e_2))$: This sub-formula is true when event e_1 is either the RTI receiving a NET message or sending a STOP_GRN message, the federates of events e_1 and e_2 are the same or the federate of event e_1 is directly upstream of the federate of event e_2 , and the tags of events e_1 and e_2 are equal.

3. Largest Sub-formula:

- $((\text{((} e_1 \text{ is the first event satisfying (...))} \wedge (e_2 \text{ is (Sending PTAG)})) \wedge ((\text{Tag } e_1) = (\text{Tag } e_2)))$: This sub-formula is true when event e_1 is the first event satisfying the conditions described in the antecedent, event e_2 is the RTI sending a PTAG message with a non-zero tag.

By analyzing the truth conditions of each sub-formula, we can understand when the antecedent of the logical implication is true, leading to the conclusion that $e_1 < e_2$.

(This explanation was generated in 15 seconds.)

Summary of the meaning of formula 10

The given formula states a logical implication that must hold true for all events e_1 and e_2 in the RTI. The formula specifies conditions under which event e_1 precedes event e_2 in physical time.

The antecedent of the formula describes a complex set of conditions that must be satisfied for event e_1 to precede event e_2 . These conditions involve scenarios where the RTI sends PTAG messages, receives NET messages, sends STOP_GRN messages, and the relationships between federates and their connections. The formula ensures that if event e_1 meets certain criteria related to message exchanges and federate relationships, and event e_2 involves the RTI sending a PTAG message with a non-zero tag, then event e_1 must occur before event e_2 .

The use of operators like `first e1 satisfying` and conditions such as the direction of message flow (sending vs. receiving) and the relationships between federates via connections are crucial in determining the order of events. The formula guarantees that the specified conditions are met consistently for all pairs of events e_1 and e_2 , ensuring a well-defined ordering of events within the RTI system.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the operational rules and message semantics within the RTI and federate system, specifically regarding logical time progression and message handling. The RTI's role in controlling the logical time advancement of federates through PTAG and TAG messages ensures that federates only proceed to new logical times when it is safe to do so, preventing STP violations. The ordering of events, as described by the guarantee, reflects the necessary sequence of operations where the RTI receives and processes information (e.g., NET messages indicating the next scheduled event for a federate) before it can safely allow federates to advance to new logical times. This sequence ensures that all dependencies and conditions are met before a federate proceeds, thereby maintaining the consistency and correctness of the simulation's logical time progression.

(This explanation was generated in 7 seconds.)

Formula 11

Formula 11 states:

```

(((e1 is the first event satisfying ( $\lambda$  e1 . ((e1 is (Receiving PORT_ABS)))
   $\wedge$  ((Federate of e1 is directly upstream of federate of e2))
   $\wedge$  ((Tag e1) = (Tag e2)))))
 $\wedge$  ((e2 is (Sending PORT_ABS))))
 $\Rightarrow$  (e1 < e2)

```

In-depth syntactic explanation

To break down the antecedent of the given implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Receiving PORT_ABS))` : This atomic sub-formula is true when event e_1 corresponds to the RTI receiving a PORT_ABS message.
- `(Federate of e_1 is directly upstream of federate of e_2)` : This atomic sub-formula is true when the federate associated with event e_1 is directly upstream of the federate associated with event e_2 .
- `(Tag e_1) = (Tag e_2)` : This atomic sub-formula is true when the tags of events e_1 and e_2 are equal.
- `(e_2 is (Sending PORT_ABS))` : This atomic sub-formula is true when event e_2 corresponds to the RTI sending a PORT_ABS message.

2. Larger Sub-formulas:

- `((e_1 is (Receiving PORT_ABS)) \wedge (Federate of e_1 is directly upstream of federate of e_2) \wedge (Tag e_1) = (Tag e_2))` : This sub-formula is true when event e_1 involves the RTI receiving a PORT_ABS message, the federate of e_1 is directly upstream of the federate of e_2 , and the tags of e_1 and e_2 are equal.
- `((e_1 is the first event satisfying (...)) \wedge (e_2 is (Sending PORT_ABS)))` : This sub-formula is true when event e_1 is the first event satisfying the conditions specified in the parentheses, and event e_2 corresponds to the RTI sending a PORT_ABS message.

3. Largest Sub-formula:

- `((e_1 is the first event satisfying (...)) \wedge (e_2 is (Sending PORT_ABS))) \Rightarrow (e1 < e2)` : This is the largest sub-formula, representing the logical implication that

if event e_1 satisfies certain conditions and is the first event to do so, and event e_2 corresponds to the RTI sending a PORT_ABS message, then event e_1 must occur before event e_2 in physical time.

In summary, the breakdown of the antecedent into its sub-formulas helps us understand the conditions under which the implication holds true. By analyzing the truth values of the atomic and larger sub-formulas, we can determine the validity of the logical implication in the context of events occurring in the RTI.

(This explanation was generated in 8 seconds.)

Summary of the meaning of formula 11

The formula states that if event e_1 is the first event where the RTI receives a PORT_ABS message from a federate directly upstream of the federate associated with event e_2 at the same tag, and event e_2 is when the RTI sends a PORT_ABS message, then event e_1 must occur before event e_2 in physical time.

In simpler terms, the formula ensures that when the RTI receives a PORT_ABS message from an upstream federate at a specific tag, and then subsequently sends a PORT_ABS message, the receiving event must happen before the sending event in physical time. This ordering constraint is crucial for maintaining the logical flow and synchronization of messages within the federated system, preventing any violations or conflicts that could arise if the sending event were to occur before the receiving event.

The use of the `first event satisfying` operator emphasizes that this ordering relationship between the receiving and sending events must be the first instance of such a pattern occurring in the sequence of events. This requirement adds an additional layer of control to ensure that the specified conditions hold true for the initial occurrence of the events meeting the defined criteria.

Overall, the formula establishes a strict temporal relationship between the reception of a PORT_ABS message from an upstream federate and the subsequent transmission of a PORT_ABS message by the RTI, enforcing a specific order of events to maintain the integrity and consistency of message propagation within the federated system.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is correct because it aligns with the operational protocol between the RTI and federates, ensuring logical consistency and temporal ordering of events. When the RTI receives a PORT_ABS message from an upstream federate, it indicates that no signal will be sent to a downstream federate at a specific logical time. Before the RTI can inform the downstream federate of this (by sending a PORT_ABS message), it must first process the received information to maintain the integrity of the logical timeline. This processing ensures that the downstream federate only proceeds with actions that are unaffected by potential signals, adhering to the system's rules to prevent STP violations. The real-world event ordering (receiving before sending) reflects the necessary sequence of actions to uphold the logical time model and ensure the system's correct operation.

(This explanation was generated in 8 seconds.)

Formula 12

Formula 12 states:

```
((e1 is the first event satisfying ( $\lambda$  e1 . ((e1 is (Receiving TAGGED_MSG)))  
   $\wedge$  ((Federate of e1 is directly upstream of federate of e2))  
   $\wedge$  ((Tag e1) = (Tag e2)))))  
 $\wedge$  ((e2 is (Sending TAGGED_MSG)))  
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas:

1. Atomic Sub-formulas:

- `(e1 is (Receiving TAGGED_MSG))` : This atomic sub-formula is true when event e_1 corresponds to the RTI receiving a TAGGED_MSG message from a federate.
- `(Federate of e1 is directly upstream of federate of e2)` :

This atomic sub-formula is true when the federate associated with event e_1 is directly upstream of the federate associated with event e_2 .

- $(\text{Tag } e_1) = (\text{Tag } e_2)$: This atomic sub-formula is true when the tags of events e_1 and e_2 are equal.
- $(e_2 \text{ is (Sending TAGGED_MSG)})$: This atomic sub-formula is true when event e_2 corresponds to the RTI sending a TAGGED_MSG message to a federate.

2. Larger Sub-formulas:

- $((e_1 \text{ is the first event satisfying } (...)))$: This sub-formula is true when event e_1 is the first event satisfying the conditions specified within the parentheses. This implies that there may be multiple events, but e_1 is the earliest one that satisfies the conditions.
- $((...) \wedge (...) \wedge (...))$: This sub-formula represents the conjunction of multiple atomic sub-formulas. It is true only when all the atomic sub-formulas within it are true simultaneously.
- \Rightarrow : This sub-formula represents the logical implication between the antecedent and the consequent. It is true unless the antecedent is true and the consequent is false.

3. Analysis:

- When $(e_1 \text{ is (Receiving TAGGED_MSG)})$ and $(\text{Federate of } e_1 \text{ is directly upstream of federate of } e_2)$ and $(\text{Tag } e_1) = (\text{Tag } e_2)$ are all true, and $(e_2 \text{ is (Sending TAGGED_MSG)})$ is also true, then the antecedent of the implication is true.
- This implies that event e_1 is the first event where the RTI receives a TAGGED_MSG message from a directly upstream federate at a specific tag, and event e_2 is the subsequent event where the RTI sends a TAGGED_MSG message.
- The logical implication states that if such a sequence of events occurs, where the RTI receives a TAGGED_MSG message before sending one, then event e_1 must precede event e_2 in physical time, denoted by $\$e_1 < e_2$.

By understanding the truth conditions of each sub-formula and how they combine to form the antecedent of the logical implication, we can determine when the antecedent is true and what it

implies about the ordering of events in the system.

(This explanation was generated in 13 seconds.)

Summary of the meaning of formula 12

The formula states that if event e_1 is the first event where the RTI receives a TAGGED_MSG message from a directly upstream federate at a specific tag, and event e_2 is the subsequent event where the RTI sends a TAGGED_MSG message, then event e_1 must occur before event e_2 in physical time. This implies that when the RTI receives a TAGGED_MSG message from an upstream federate before sending a TAGGED_MSG message to another federate, the receiving event must precede the sending event in physical time.

The use of `first e1 satisfying` ensures that we are considering the earliest event that satisfies the specified conditions, indicating that there may be multiple events but we are focusing on the first occurrence. The formula guarantees that this ordering constraint holds true for all possible events e_1 and e_2 , emphasizing the importance of maintaining the chronological sequence of receiving and sending TAGGED_MSG messages in the system to ensure proper message flow and synchronization between federates.

(This explanation was generated in 3 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the operational protocol between the RTI and federates regarding message handling and logical time progression. In a distributed simulation system, the RTI acts as a coordinator to ensure that messages are processed in a logically consistent order, respecting the logical delays associated with connections between federates. When the RTI receives a TAGGED_MSG from an upstream federate, it signifies an event that must be logically processed before it can be forwarded to the downstream federate. This ensures that events are handled in a sequence that respects the logical time, preventing any causality violations or STP (safe-to-process) violations. The physical time ordering of receiving before sending is crucial for maintaining the integrity of the simulation, ensuring that all federates operate on a consistent view of the simulated world's state.

(This explanation was generated in 11 seconds.)

Formula 13

Formula 13 states:

```
(((((e1 is (Receiving PORT_ABS))) v ((e1 is (Receiving TAGGED_MSG))))  
  ∧ (Federate (e1) = Federate (e2))  
  ∧ ((Tag e1) ≤ (Tag e2)))  
  ∧ (((e2 is (Receiving LTC)))))  
⇒ (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the logical implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- $(e_1 \text{ is (Receiving PORT_ABS)})$: This sub-formula is true when event e_1 corresponds to the RTI receiving a PORT_ABS message.
- $(e_1 \text{ is (Receiving TAGGED_MSG)})$: This sub-formula is true when event e_1 corresponds to the RTI receiving a TAGGED_MSG message.
- $(\text{Federate}(e_1) = \text{Federate}(e_2))$: This sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- $((\text{Tag } e_1) \leq (\text{Tag } e_2))$: This sub-formula is true when the logical time tag of event e_1 is less than or equal to the logical time tag of event e_2 .
- $(e_2 \text{ is (Receiving LTC)})$: This sub-formula is true when event e_2 corresponds to the RTI receiving a LTC message.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Receiving PORT_ABS)}) v (e_1 \text{ is (Receiving TAGGED_MSG)}))$: This sub-formula is true when event e_1 corresponds to the RTI receiving either a PORT_ABS message or a TAGGED_MSG message.
- $((e_1 \text{ is (Receiving PORT_ABS)}) v (e_1 \text{ is (Receiving TAGGED_MSG)})) \wedge ((\text{Federate}(e_1) = \text{Federate}(e_2)) \wedge ((\text{Tag } e_1) \leq (\text{Tag } e_2)))$: This sub-formula is true when event e_1 satisfies the conditions of receiving a PORT_ABS or TAGGED_MSG message, having the same

federate as event e_2 , and occurring at a logical time tag less than or equal to that of event e_2 .

- $((e_1 \text{ is (Receiving PORT_ABS)}) \vee (e_1 \text{ is (Receiving TAGGED_MSG)})) \wedge$
: This sub-formula is true when the previous sub-formula is satisfied, and event e_2 corresponds to the RTI receiving a LTC message.

3. Conclusion:

- The entire antecedent of the logical implication is true when the above largest sub-formula is satisfied, indicating that if event e_1 involves receiving a PORT_ABS or TAGGED_MSG message from the same federate as event e_2 at a logical time tag less than or equal to that of event e_2 , and event e_2 involves receiving a LTC message, then event e_1 must precede event e_2 in the logical time ordering.

By breaking down the antecedent into its sub-formulas and analyzing their truth conditions, we can understand the implications of the logical formula in the context of events occurring in the RTI system.

(This explanation was generated in 9 seconds.)

Summary of the meaning of formula 13

The formula provided states that if event e_1 in the RTI involves receiving a PORT_ABS or TAGGED_MSG message from the same federate as event e_2 , at a logical time tag less than or equal to that of event e_2 , and event e_2 involves receiving a LTC message, then event e_1 must precede event e_2 in the logical time ordering.

In simpler terms, the formula ensures that when the RTI receives a message (PORT_ABS or TAGGED_MSG) from a federate and the logical time tag of that event is before or at the same time as another event where the RTI receives a LTC message, then the event of receiving the initial message must happen before the event of receiving the LTC message. This logical implication guarantees a specific ordering constraint between these types of events in the RTI system, ensuring proper synchronization and sequencing of actions within the federated program.

(This explanation was generated in 5 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the operational rules governing the RTI and federates within a distributed simulation system. Specifically, PORT_ABS and TAGGED_MSG messages indicate communication about the absence or presence of signals at certain logical times between federates, while a LTC (Logical Tag Complete) message signifies a federate's completion of all activities at a specific logical time. The ordering ensures that the RTI processes information about potential interactions (via PORT_ABS and TAGGED_MSG) before acknowledging a federate's completion of a logical time (via LTC). This sequence is critical for maintaining logical consistency and preventing causality violations, as it ensures that all relevant communications are accounted for before moving past a logical time point, thus upholding the integrity of the simulation's temporal logic.

(This explanation was generated in 12 seconds.)

Formula 14

Formula 14 states:

```
((((e1 is (Receiving FED_ID))) ∧ (Federate (e1) = Federate (e2))) ∧ ((e2 is (Sending ACK))))  
⇒ (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas:

1. Atomic Sub-formulas:

- `(e_1 is (Receiving FED_ID))` : This atomic sub-formula is true when event e_1 corresponds to the RTI receiving a FED_ID message from a federate.
- `(Federate(e_1) = Federate(e_2))` : This atomic sub-formula is true when events e_1 and e_2 involve the same federate.
- `(e_2 is (Sending ACK))` : This atomic sub-formula is true when event e_2 corresponds to the RTI sending an ACK message.

2. Larger Sub-formulas:

- $((e_1 \text{ is (Receiving FED_ID)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)))$
: This sub-formula is true when event e_1 is the RTI receiving a FED_ID message and events e_1 and e_2 involve the same federate.
- $((e_1 \text{ is (Receiving FED_ID)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2))) \wedge (e_2 \text{ is (Sending ACK)}))$
: This sub-formula is true when the above conditions hold, and event e_2 corresponds to the RTI sending an ACK message.

3. Analysis of Sub-formulas:

- $(e_1 \text{ is (Receiving FED_ID)})$: This sub-formula is true when event e_1 involves the RTI receiving a FED_ID message. This implies that a federate is declaring its ID to the RTI during program initialization.
- $(\text{Federate}(e_1) = \text{Federate}(e_2))$: This sub-formula is true when events e_1 and e_2 involve the same federate. This ensures that the events are related to the same federate.
- $(e_2 \text{ is (Sending ACK)})$: This sub-formula is true when event e_2 corresponds to the RTI sending an ACK message. This implies that the RTI is acknowledging the receipt of a federate ID.

4. Conclusion:

- The larger sub-formula $((e_1 \text{ is (Receiving FED_ID)}) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)))$ ensures that event e_1 involves the RTI receiving a federate ID message from a federate, and events e_1 and e_2 pertain to the same federate.
- When this condition is met along with the sub-formula $(e_2 \text{ is (Sending ACK)})$, it implies that after the RTI receives a federate ID from a federate, it sends an acknowledgment (ACK) message back to the federate.
- The implication as a whole states that if the RTI receives a federate ID and then sends an ACK message to the same federate, the event of receiving the ID must precede the event of sending the ACK message in logical time.

By breaking down the antecedent into its sub-formulas and analyzing their truth conditions, we can understand the logical implications and requirements specified by the formula in the context of the RTI's interactions with federates during program initialization.

(This explanation was generated in 11 seconds.)

Summary of the meaning of formula 14

The formula states that if the RTI receives a FED_ID message from a federate (e_1 is the event of receiving the FED_ID) and then sends an ACK message (e_2 is the event of sending the ACK) to the same federate, then the event of receiving the FED_ID must logically precede the event of sending the ACK.

In simpler terms, when a federate declares its ID to the RTI during program initialization (FED_ID message), the RTI must acknowledge the receipt of this ID (ACK message) in a logical order where the acknowledgment follows the reception of the ID.

This formula ensures a specific sequence of events during the initialization phase of the federated program, where the RTI and federates establish communication and synchronization. The logical implication guarantees that the RTI handles federate ID declarations and acknowledgments in a well-defined order, maintaining the integrity and consistency of the initialization process.

(This explanation was generated in 3 seconds.)

High-level justification

The guarantee is correct because it aligns with the fundamental protocol of initialization and communication between federates and the RTI. During initialization, a federate must first declare its presence and identity to the RTI by sending a FED_ID message. The RTI, upon receiving this declaration, acknowledges it by sending back an ACK message. This sequence is essential for establishing a reliable communication channel between the federate and the RTI. The logical ordering (\prec) reflects the necessary real-world sequence of these events to ensure that the federate is recognized by the RTI before any further interaction, such as time synchronization or event processing, takes place. This order is a basic requirement for the coherent initialization and operation of the federated system, ensuring that each federate is properly registered and acknowledged before proceeding with the simulation activities.

(This explanation was generated in 10 seconds.)

Formula 15

Formula 15 states:

```
((((e1 is (Sending ACK)))  $\wedge$  (Federate (e1) = Federate (e2))))  
   $\wedge$  ((e2 is (Receiving TIMESTAMP))))  
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas:

1. Atomic Sub-formulas:

- `(e_1 is (Sending ACK))` : This atomic sub-formula is true when event e_1 corresponds to the RTI sending an ACK message. This means that the RTI acknowledges receipt of a federate ID.
- `(Federate(e_1) = Federate(e_2))` : This atomic sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- `(e_2 is (Receiving TIMESTAMP))` : This atomic sub-formula is true when event e_2 corresponds to the RTI receiving a TIMESTAMP message.

2. Larger Sub-formulas:

- `((e_1 is (Sending ACK)) \wedge (Federate(e_1) = Federate(e_2)))` : This sub-formula is true when event e_1 is the RTI sending an ACK message and the federate associated with e_1 is the same as the federate associated with e_2 .
- `((((e_1 is (Sending ACK)) \wedge (Federate(e_1) = Federate(e_2))) \wedge (e_2 is (Receiving TIMESTAMP)))` : This sub-formula is true when the previous sub-formula is true and event e_2 corresponds to the RTI receiving a TIMESTAMP message.

3. Implication Sub-formula:

- `((((e_1 is (Sending ACK)) \wedge (Federate(e_1) = Federate(e_2))) \wedge (e_2 is (Receiving TIMESTAMP))) \Rightarrow (e_1 < e_2))` : This is the original implication formula. It is true when the antecedent is true, i.e., when event e_1 is the RTI sending an ACK message to the same federate as event e_2 , and event e_2 is the RTI receiving a TIMESTAMP message, then event e_1 logically precedes event e_2 .

In summary, the breakdown of the antecedent of the implication into its sub-formulas helps us understand the conditions under which the implication as a whole is true. By analyzing the truth

values of the atomic and larger sub-formulas, we can determine when the entire implication holds based on the logical relationships between the events described in the formula.

(This explanation was generated in 11 seconds.)

Summary of the meaning of formula 15

The formula states that if event e_1 represents the RTI sending an ACK message to a federate and event e_2 represents the RTI receiving a TIMESTAMP message, and both events involve the same federate, then event e_1 must logically precede event e_2 .

In simpler terms, the formula ensures that when the RTI acknowledges receipt of a federate ID by sending an ACK message to a federate, and then receives a TIMESTAMP message from the same federate, the ACK message must have been sent before the TIMESTAMP message is received. This logical ordering constraint guarantees that the acknowledgment precedes the receipt of the timestamp, maintaining the expected sequence of events between the RTI and the federate.

The formula captures a specific temporal relationship between the sending of an acknowledgment and the subsequent receipt of a timestamp, ensuring a consistent and ordered flow of communication between the RTI and federates in the system.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee is correct because it aligns with the initialization sequence of a federated system, where a federate first identifies itself to the RTI by sending its Federate ID (FED_ID), and in response, the RTI sends back an acknowledgment (ACK) to confirm the receipt of the ID. Only after this exchange can the federate proceed to the next step of synchronization, which involves sending a TIMESTAMP message to establish a common start time. This sequence is critical for ensuring that all federates are correctly registered and acknowledged by the RTI before any logical time synchronization occurs. The ordering of these events—ACK before TIMESTAMP—reflects the necessary procedural steps for initializing communication and synchronization between federates and the RTI, adhering to the rules governing their interaction.

(This explanation was generated in 10 seconds.)

Formula 16

Formula 16 states:

```
((((e1 is (Receiving TIMESTAMP)))  $\wedge$  (Federate (e1) = Federate (e2))))  
   $\wedge$  ((e2 is (Sending TIMESTAMP))))  
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e_1 is (Receiving TIMESTAMP))` : This atomic sub-formula is true when event e_1 corresponds to the RTI receiving a TIMESTAMP message.
- `(Federate(e_1) = Federate(e_2))` : This atomic sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- `(e_2 is (Sending TIMESTAMP))` : This atomic sub-formula is true when event e_2 corresponds to the RTI sending a TIMESTAMP message.

2. Larger Sub-formulas:

- `((e_1 is (Receiving TIMESTAMP)) \wedge (Federate(e_1) = Federate(e_2)))` : This sub-formula is true when event e_1 is the RTI receiving a TIMESTAMP message and the federate associated with e_1 is the same as the federate associated with e_2 .
- `((e_2 is (Sending TIMESTAMP)))` : This sub-formula is true when event e_2 corresponds to the RTI sending a TIMESTAMP message.

3. Complete Antecedent:

- `((((e_1 is (Receiving TIMESTAMP)) \wedge (Federate(e_1) = Federate(e_2)))) \wedge ((e_2 is (Sending TIMESTAMP))))` : This is the complete antecedent of the implication. It is true when event e_1 is the RTI receiving a TIMESTAMP message, the federate associated with e_1 is the same as the federate associated with e_2 , and

event e_2 is the RTI sending a TIMESTAMP message.

In summary, the complete antecedent is true when the RTI receives a TIMESTAMP message from a federate, the federate associated with that event is the same as the federate sending a TIMESTAMP message, and then the RTI sends a TIMESTAMP message. This sequence of events ensures that the sending of the TIMESTAMP message follows the receiving of the TIMESTAMP message from the same federate, which is captured by the implication $e_1 \prec e_2$.

(This explanation was generated in 17 seconds.)

Summary of the meaning of formula 16

The formula states that if the RTI receives a TIMESTAMP message from a federate (e_1) and then subsequently sends a TIMESTAMP message (e_2), both events involving the same federate, then it must be the case that the event of receiving the TIMESTAMP message precedes the event of sending the TIMESTAMP message in logical time.

In other words, the formula ensures that when a federate communicates its timestamp to the RTI and the RTI acknowledges this timestamp by sending it back to the federate, the acknowledgment message is sent after the receipt of the timestamp message. This logical ordering is crucial for maintaining the synchronization and proper flow of information between the federates and the RTI in the system.

The formula guarantees this temporal relationship between the two events for all possible events e_1 and e_2 , ensuring that the communication protocol between federates and the RTI is correctly followed without violating the logical time constraints set by the system.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee is expected to be correct because it aligns with the fundamental protocol of communication between federates and the RTI, particularly during initialization. When a federate sends a TIMESTAMP message to the RTI, it is declaring its start time for the main part of the program. The RTI, upon receiving this TIMESTAMP, processes it to ensure synchronization among all federates. The RTI then sends a TIMESTAMP message back to the federate as an acknowledgment or part of the synchronization process. This sequence—receiving the TIMESTAMP from the federate and then sending an acknowledgment or synchronized

TIMESTAMP back—naturally follows the logical flow of initialization and time synchronization in distributed simulations. The real-world event ordering (the RTI receiving a TIMESTAMP before sending one back) reflects the necessary procedural steps to establish a common logical time frame among all participating entities, ensuring that all federates start from a synchronized state.

(This explanation was generated in 12 seconds.)

Formula 17

Formula 17 states:

```
((((e1 is (Sending TIMESTAMP)))  $\wedge$  (Federate (e1) = Federate (e2)))  
  $\wedge$  (((e2 is (Receiving NET)))  $\wedge$  ( $\neg$  ((Tag e2)  $\neq$  0))))  
 $\Rightarrow$  (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas:

1. Atomic Sub-formulas:

- `(e_1 is (Sending TIMESTAMP))` : This sub-formula is true when event e_1 corresponds to the RTI sending a TIMESTAMP message.
- `(Federate(e_1) = Federate(e_2))` : This sub-formula is true when the federate associated with event e_1 is the same as the federate associated with event e_2 .
- `(e_2 is (Receiving NET))` : This sub-formula is true when event e_2 corresponds to the RTI receiving a NET message.
- `(\neg ((Tag e_2) \neq 0))` : This sub-formula is true when the tag associated with event e_2 is not equal to 0.

2. Larger Sub-formulas:

- `((e_1 is (Sending TIMESTAMP)) \wedge (Federate(e_1) = Federate(e_2)))` : This sub-formula is true when event e_1 involves the RTI sending a TIMESTAMP message and the federate associated with e_1 is the same as the federate associated with e_2 .
- `((e_2 is (Receiving NET)) \wedge (\neg ((Tag e_2) \neq 0)))` : This sub-

formula is true when event e_2 involves the RTI receiving a NET message and the tag associated with e_2 is not equal to 0.

3. Largest Sub-formula:

- $(((((e_1 \text{ is (Sending TIMESTAMP)})) \wedge (\text{Federate}(e_1) = \text{Federate}(e_2)))) \wedge$
: This is the largest sub-formula in the antecedent of the implication. It is true when both the sub-formulas related to event e_1 and event e_2 are true simultaneously.

In summary, the antecedent of the implication is true when event e_1 corresponds to the RTI sending a TIMESTAMP message to the same federate as event e_2 , and event e_2 corresponds to the RTI receiving a NET message with a tag not equal to 0.

(This explanation was generated in 10 seconds.)

Summary of the meaning of formula 17

The formula states that if event e_1 in the RTI involves sending a TIMESTAMP message to a federate, and event e_2 in the RTI involves receiving a NET message with a tag equal to 0 from the same federate, then event e_1 must occur before event e_2 in the logical order of events.

In other words, the formula ensures that when the RTI sends a TIMESTAMP message to a federate and that federate subsequently sends a NET message with a tag of 0 back to the RTI, the sending of the TIMESTAMP message must logically precede the receiving of the NET message with a tag of 0.

This logical implication guarantees the ordering of events related to the exchange of specific types of messages between the RTI and federates, ensuring that the sending and receiving of these messages occur in the correct sequence as defined by the logical time of the events.

(This explanation was generated in 4 seconds.)

High-level justification

The guarantee is correct because it aligns with the initialization and logical time progression protocols between the RTI and federates. Initially, the RTI sends a TIMESTAMP message to a federate to establish a start time for the simulation. Only after this initialization step can a federate schedule events and thus send a NET message indicating the time of its next scheduled event, starting logically at tag 0. This sequence ensures that the federate is aware of the simulation start

time before it can declare its readiness to process or schedule events at a specific logical time, maintaining consistency in the logical timeline and ensuring that events proceed in an orderly and predictable manner within the distributed simulation system.

(This explanation was generated in 5 seconds.)

Formula 18

Formula 18 states:

```
((((e1 is (Receiving TIMESTAMP))))
  ^
  (((e2 is (Receiving LTC)))
    v ((e2 is (Receiving PORT_ABS)))
    v ((e2 is (Receiving TAGGED_MSG)))
    v ((e2 is (Sending TAG)))
    v ((e2 is (Sending PTAG)))
    v ((e2 is (Sending PORT_ABS)))
    v ((e2 is (Sending TAGGED_MSG)))
    v ((e2 is (Sending STOP_GRN)))
    v ((e2 is (Sending STOP_REQ)))
    v ((e2 is (Receiving STOP_REQ)))
    v ((e2 is (Receiving STOP_REQ_REP)))))
  => (e1 < e2)
```

In-depth syntactic explanation

To break down the antecedent of the implication into its sub-formulas and analyze when they are true, we will start by examining the atomic sub-formulas and then build up to larger sub-formulas.

1. Atomic Sub-formulas:

- `(e1 is (Receiving TIMESTAMP))` : This sub-formula is true when event e_1 is an event in which the RTI is receiving a TIMESTAMP message.
- `(e2 is (Receiving LTC))` : This sub-formula is true when event e_2 is an event in which the RTI is receiving a LTC message.
- `(e2 is (Receiving PORT_ABS))` : This sub-formula is true when

event e_2 is an event in which the RTI is receiving a PORT_ABS message.

- $(e_2 \text{ is (Receiving TAGGED_MSG)})$: This sub-formula is true when event e_2 is an event in which the RTI is receiving a TAGGED_MSG message.
- $(e_2 \text{ is (Sending TAG)})$: This sub-formula is true when event e_2 is an event in which the RTI is sending a TAG message.
- $(e_2 \text{ is (Sending PTAG)})$: This sub-formula is true when event e_2 is an event in which the RTI is sending a PTAG message.
- $(e_2 \text{ is (Sending PORT_ABS)})$: This sub-formula is true when event e_2 is an event in which the RTI is sending a PORT_ABS message.
- $(e_2 \text{ is (Sending TAGGED_MSG)})$: This sub-formula is true when event e_2 is an event in which the RTI is sending a TAGGED_MSG message.
- $(e_2 \text{ is (Sending STOP_GRN)})$: This sub-formula is true when event e_2 is an event in which the RTI is sending a STOP_GRN message.
- $(e_2 \text{ is (Sending STOP_REQ)})$: This sub-formula is true when event e_2 is an event in which the RTI is sending a STOP_REQ message.
- $(e_2 \text{ is (Receiving STOP_REQ)})$: This sub-formula is true when event e_2 is an event in which the RTI is receiving a STOP_REQ message.
- $(e_2 \text{ is (Receiving STOP_REQ_REP)})$: This sub-formula is true when event e_2 is an event in which the RTI is receiving a STOP_REQ_REP message.

2. Larger Sub-formulas:

- $(e_2 \text{ is (Receiving LTC)}) \vee (e_2 \text{ is (Receiving PORT_ABS)}) \vee (e_2 \text{ is (Receiving TAG)})$: This sub-formula is true when event e_2 is any of the specified types of events related to message sending and receiving.

3. Implication Sub-formula:

- $((e_1 \text{ is (Receiving TIMESTAMP)}) \wedge ((e_2 \text{ is (Receiving LTC)}) \vee \dots)) \Rightarrow$: This is the complete antecedent of the implication, stating that if event e_1 is receiving a TIMESTAMP message and event e_2 is related to message sending or receiving, then e_1 must occur before e_2 in logical time.

By breaking down the antecedent into its sub-formulas and analyzing when each sub-formula is true, we can understand the conditions under which the implication holds true in the context of the RTI events and message handling described in the system.

(This explanation was generated in 15 seconds.)

Summary of the meaning of formula 18

The formula states that if the RTI receives a `TIMESTAMP` message (e_1) and subsequently encounters an event involving the receipt or transmission of various types of messages (e_2), then e_1 must logically occur before e_2 in the sequence of events.

This implies that the receipt of a `TIMESTAMP` message triggers a constraint on the subsequent events related to message handling. The formula ensures that events involving the RTI receiving or sending messages such as `LTC`, `PORT_ABS`, `TAGGED_MSG`, `TAG`, `PTAG`, `STOP_GRN`, `STOP_REQ`, or `STOP_REQ_REP` must occur after the receipt of the `TIMESTAMP` message. This ordering constraint is crucial for maintaining the logical sequence of events within the system.

The formula guarantees that the logical time ordering between the receipt of a `TIMESTAMP` message and subsequent message-related events is preserved, preventing any violations where events related to message handling occur before the receipt of the `TIMESTAMP` message. This strict ordering ensures the proper synchronization and coordination of message exchanges within the federated system, as dictated by the rules and protocols described in the context of the RTI's functionality.

(This explanation was generated in 9 seconds.)

High-level justification

The guarantee is expected to be correct because the `TIMESTAMP` message, used during initialization to synchronize the start time between federates and the RTI, logically precedes other operational messages (such as `LTC`, `PORT_ABS`, `TAGGED_MSG`, `TAG`, `PTAG`, `STOP_GRN`, `STOP_REQ`, or `STOP_REQ_REP`) that are part of the system's runtime behavior. This initial synchronization step ensures that all entities within the system agree on a common logical start time, which is foundational for the orderly progression of logical time and the correct sequencing of events. Subsequent messages, which dictate the flow of events, logical time advancement,

and signal handling between federates and the RTI, rely on this initial time agreement to maintain consistency and prevent logical time violations. Thus, the `TIMESTAMP` message's receipt by the RTI naturally precedes the handling of operational messages, reflecting the system's need to establish a coherent temporal framework before engaging in the complex interactions that govern its behavior.

(This explanation was generated in 9 seconds.)