Verifying the Soundness of Resource Analysis for LogicGuard Monitors
Revised Version*

Temur Kutsia Wolfgang Schreiner

RISC, Johannes Kepler University Linz

{kutsia,schreine}@risc.jku.at

September 17, 2014

Abstract

In a companion paper (Wolfgang Schreiner, Temur Kutsia. A Resource Analysis for LogicGuard Monitors. RISC Technical report, December 5, 2013) we described a static analysis to determine whether a specification expressed in the LogicGuard language gives rise to a monitor that can operate with a finite amount of resources, notably with finite histories of the streams that are monitored. Here we prove the soundness of the analysis with respect to a formal operational semantics. The analysis is presented for an abstract core language that monitors a single stream.

Contents

1 Introduction **2**

2 The Core Language and Resource Analysis **2**

3 Operational Semantics **4**

4 Soundness of Resource Analysis **9**

5 Conclusion **14**

A Proofs **16**

A.1 Theorem 1: Soundness Theorem . . . 16
A.2 Proposition 1: The Invariant Statement . . . 26
A.3 Lemma 1: Soundness Lemma for Formulas . . . 36
A.4 Lemma 2: Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions 46
A.5 Lemma 3: History Cut-Off Lemma . . . 53
A.6 Lemma 4: n-Step Reductions to **done** Formulas for TN, TCS, TCP . . . 70
A.7 Lemma 5: Soundness Lemma for Universal Formulas . . . 95
A.8 Lemma 6: Monotonicity of Reduction to **done** . . . 101
A.9 Lemma 7: Shifting Lemma . . . 106
A.10 Lemma 8: Triangular Reduction Lemma . . . 107
A.11 Lemma 9: Soundness of Bound Analysis . . . 126
A.12 Lemma 10: Invariant Lemma for Universal Formulas . . . 130

*The project "LogicGuard: The Efficient Checking of Time-Quantified Logic Formulas with Applications in Computer Security" is sponsored by the FFG BRIDGE program, project No. 832207.

1 Introduction

The goal of the LogicGuard project is to investigate to what extent classical predicate logic formulas are suitable as the basis for the specification and efficient runtime verification of system runs. The specific focus of the project is on computer and network security, concentrating on predicate logic specifications of security properties of network traffic. Properties are expressed by quantified formulas interpreted over sequences of messages; the quantified variable denotes a position in the sequence. Using the ordering of stream positions and nested quantification, complex properties can be formulated. Furthermore, to raise the level of abstraction, a higher-level stream may be constructed from a lower-level stream by a notation analogous to classical set builders. A translator generates from the specification an executable monitor.

The main ideas of these developments have been presented in [4] and [5]; in [1], the syntax and semantics of (an early abstract form of) the specification language are given; in [2], the translation of a specification to an executable monitor is described. A prototype of the translator and of the corresponding runtime system have been implemented and are operational.

The current implementation assumes that the whole "history" of a stream is preserved, i.e., that all received messages are stored in memory; thus the memory requirements of a monitor continuously grow. In practice, however, we are only interested in monitors that operate for an indefinite amount of time within a bounded amount of memory.

In [6], we tried to fill this gap by presenting a static analysis that

- is able to determine whether a given specification can be monitored with a finite amount of history (and that may consequently generate a warning/error message, if not) and that
- generates corresponding information in an easily accessible form such that after each execution step the runtime system of the monitor may appropriately prune the histories of the streams on which it operates.

One part of [6] was devoted to presenting the main ideas of the analysis by an abstract core language, which is only a skeleton of the real language; in particular it only monitors a single stream and does not support the construction of virtual streams. In this report, we use this language to formalize the operational semantics of the monitor execution and prove the soundness of the analysis presented in this report with respect to that semantics.

This paper is organized as follows: In Sect. 2 we briefly recall the definitions of the core language and the resource analysis from [6]. In Sect. 3 the operational semantics of the core language is described. In Sect. 4 the main result is formulated: soundness of the resource analysis with respect to the operational semantics. This section contains also all the lemmas needed for proving the soundness theorem. The proofs can be found in the Appendix.

This paper is an extended and revised version of [3] and subsumes it: We fixed typos, added Lemma 10 and the proof of Lemma 5, and in some places modified the statements and proofs of the other lemmas.

2 The Core Language and Resource Analysis

The core language is depicted in Figure 1.

A specification in the core language describes a single monitor that controls a single stream of Boolean values where the atomic predicate `@`X denotes the value on the stream at the position X, `~`X denotes negation, F_1 `&&` F_2 denotes sequential conjunction (the evaluation of F_2 is delayed until the value of F_1 becomes available), F_1 `/\` F_2 describes parallel evaluation (both formulas are evaluated simultaneously until one of them becomes false or both become true) and `forall` X `in` B_1`..`B_2 `:` F evaluates F at all positions in the range denoted by the interval $B_1 \ldots B_2$ until one instance becomes false or all instances become true; the creation of a new instance $F[n]$ is triggered by the arrival of the message number n on the stream.

This language is interpreted over a single stream of messages carrying truth values. We assume that a monitor M in this language is executed as follows: whenever a new message arrives on the

$$
\begin{aligned}
M &::= \texttt{monitor}\ X\ \texttt{:}\ F \\
F &::= \texttt{@}X \ \mid\ \texttt{~}F \ \mid\ F_1\ \texttt{\&\&}\ F_2 \ \mid F_1\ \texttt{/\textbackslash}\ F_2 \ \mid\ \texttt{forall}\ X\ \texttt{in}\ B_1\texttt{..}B_2\ \texttt{:}\ F \\
B &::= \texttt{0} \ \mid \texttt{infinity} \ \mid\ X \ \mid\ B+N \ \mid\ B-N \\
N &::= \texttt{0} \ \mid\ \texttt{1} \ \mid\ \texttt{2} \ \mid\ \ldots \\
X &::= \texttt{x} \ \mid\ \texttt{y} \ \mid\ \texttt{z} \ \mid\ \ldots
\end{aligned}
$$

Figure 1: The Core Language

stream, an instance $F[p/X]$ of the monitor body F is created where p denotes the position of the message in the stream. All instances are evaluated on every subsequently arriving message which may or may not let the instance evaluate to a definite truth value; whenever an instance evaluates to such a value, this instance is discarded from the set; the positions of instances with negative truth values are reported as "violations" of the monitor.

A formula F in a monitor instance is evaluated as follows:

- the predicate `@`X is immediately evaluated to the truth value of the message at position X of the stream (see below for further explanation);
- `~`F first evaluates F and then negates the result;
- F_1 `&&` F_2 first evaluates F_1 and, if the result is true, then also evaluates F_2;
- F_1 `/\` F_2 evaluates both F_1 and F_2 "in parallel" until the value of one subformula determines the value of the total formula;
- `forall` X `in` B_1`..`B_2 `:` F first determines the bounds of the position interval $[B_1, B_2]$; it then creates for every position p in the interval, as soon as the messages in the stream reach that position, an instance $F[p/X]$ of the formula body. All instances are evaluated on the subsequently arriving messages until all instances have been evaluated to "true" (and no more instances are to be generated) or some instance has been evaluated to "false".

We assume that the monitoring formula M is closed, i.e., every occurrence of a position variable in it is bound by a quantifier `monitor` or `forall`. Since by the evaluation strategies for these quantifiers, a formula instance is created only when the messages have reached the position assigned to the quantified variable, every occurrence of predicate `@`X can be immediately evaluated without delay.

We are interested in determining bounds for the resources used by the monitor, i.e., in particular in the following questions:

1. From the position where a monitor instance is created, how many "look-back" positions are required to evaluate the formula? This value determines the size of the "history" of past messages that have to be preserved in an implementation of the monitor.
2. How many instances can be active at the same time? This value determines the size that has to be reserved for the set of instances in the implementation of the monitor.

The basic idea for the analysis is a sort of "abstract interpretation" of the monitor where in a top-down fashion every position variable X is annotated as $X^{(l,u)}$ where the interval $[p+l, p+u]$ denotes those positions that the variables can have in relation to the position p of the "current" message of the stream; in a bottom up step, we then annotate every formula F with a pair (h, d) where h is (an upper bound of) the size of the "history" (the number of past messages) required for the evaluation of F and d is (an upper bound of) the number of future messages that may be required such that the evaluation of F may be "delayed" by this number of steps.

The basic idea is formalized in Figures 2 and 3 by a rule system with three kinds of judgements:

$$\vdash M : \mathbb{N}^\infty \times \mathbb{N}^\infty \quad \mathit{Environment} \vdash F : \mathbb{N}^\infty \times \mathbb{N}^\infty \quad \mathit{Environment} \vdash B : \mathbb{Z}^\infty \times \mathbb{Z}^\infty$$

$$\frac{[[\![X]\!] \mapsto (0,0)] \vdash F : (h,d)}{\vdash (\texttt{monitor}\ X\ \texttt{:}\ F) : (h,d)}$$

$$e \vdash \texttt{@}X : (0,0) \qquad \frac{e \vdash F : (h,d)}{e \vdash \texttt{\~{}}F : (h,d)}$$

$$\frac{e \vdash F_1 : (h_1,d_1),\ e \vdash F_2 : (h_2,d_2)}{e \vdash F_1\ \texttt{\&\&}\ F_2 : (\max^\infty(h_1, h_2 +^\infty d_1), \max^\infty(d_1,d_2))}$$

$$\frac{e \vdash F_1 : (h_1,d_1),\ e \vdash F_2 : (h_2,d_2)}{e \vdash F_1\ \texttt{/\textbackslash}\ F_2 : (\max^\infty(h_1,h_2), \max^\infty(d_1,d_2))}$$

$$\frac{\begin{array}{l} e \vdash B_1 : (l_1,u_1),\ e \vdash B_2 : (l_2,u_2) \\ e[[\![X]\!] \mapsto (l_1,u_2)] \vdash F : (h',d') \\ h = \max^\infty(h', \mathbb{N}^\infty(-^\infty l_1)) \\ d = \max^\infty(d', \mathbb{N}^\infty(u_2)) \end{array}}{e \vdash \texttt{forall}\ X\ \texttt{in}\ B_1\texttt{..}B_2\ \texttt{:}(h,d)}$$

$$e \vdash \texttt{0} : (-\infty, 0) \quad e \vdash\ \texttt{infinity} : (\infty,\infty) \quad \frac{[\![X]\!] \notin \mathit{domain}(e)}{e \vdash X : (0,0)} \quad \frac{[\![X]\!] \in \mathit{domain}(e)}{e \vdash X : e([\![X]\!])}$$

$$\frac{e \vdash B : (l,u)}{e \vdash B\texttt{+}N : (l +^\infty [\![N]\!], u +^\infty [\![N]\!])} \quad \frac{e \vdash B : (l,u)}{e \vdash B\texttt{-}N : (l -^\infty [\![N]\!], u -^\infty [\![N]\!])}$$

Figure 2: The Analysis of the Core Language

- $\vdash M : (h,d)$ states that the evaluation of the monitor M requires at most h messages from the past of the stream and at most d old monitor instances.
- $e \vdash F : (h,d)$ states that the evaluation of formula F requires at most h messages from the past of the stream and at most d messages from the future of the stream. e denotes a partial mapping of variables to pairs (l,u) denoting the lower bound and upper bound of the interval relative to the position of the "current" message.
- $e \vdash B : (l,u)$ determines the lower bound l and upper bound u for the position denoted by an interval bound B.

We have $(h,d) \in \mathbb{N}^\infty \times \mathbb{N}^\infty$ where $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$; a value of ∞ indicates that the corresponding resource (history/instance set) cannot be bounded by the analysis. We have $e(X) \in \mathbb{Z}^\infty \times \mathbb{Z}^\infty$ where $\mathbb{Z}^\infty = \mathbb{Z} \cup \{\infty, -\infty\}$; a value of ∞, respectively $-\infty$, indicates that the position cannot be bounded from above, respectively from below, by the analysis. We have $(l,u) \in \mathbb{Z}^\infty \times \mathbb{Z}^\infty$; a value of ∞ for u indicates that the corresponding interval has no upper bound; a value of $-\infty$ for l indicates that the interval has no lower bound.

In [6] one can find more detailed illustration of the resource analysis, based on examples.

3 Operational Semantics

In this section we describe formalization of the operational interpretation of a monitor by a translation $T : \mathit{Monitor} \rightarrow \mathit{TMonitor}$ from the abstract syntax domain *Monitor* to a domain *TMonitor* denoting the runtime representation of the monitor. First, we list the domains used in the formal-

$$Environment := Variable \rightarrow \mathbb{Z}^\infty \times \mathbb{Z}^\infty$$
$$\mathbb{N}^\infty := \mathbb{N} \cup \{\infty\}, \mathbb{Z}^\infty := \mathbb{Z} \cup \{\infty, -\infty\}$$

$$<^\infty \subseteq \mathbb{N} \times \mathbb{N}^\infty$$
$$n_1 <^\infty n_2 :\Leftrightarrow n_2 = \infty \vee n_1 < n_2$$

$$\leq^\infty \subseteq \mathbb{N} \times \mathbb{N}^\infty$$
$$n_1 \leq^\infty n_2 :\Leftrightarrow n_2 = \infty \vee n_1 \leq n_2$$

$$>^\infty \subseteq \mathbb{N} \times \mathbb{N}^\infty$$
$$n_1 >^\infty n_2 :\Leftrightarrow n_2 \neq \infty \wedge n_1 > n_2$$

$$\geq^\infty \subseteq \mathbb{N} \times \mathbb{N}^\infty$$
$$n_1 \geq^\infty n_2 :\Leftrightarrow n_2 \neq \infty \wedge n_1 \geq n_2$$

$$max^\infty : \mathbb{N} \times \mathbb{N}^\infty \rightarrow \mathbb{N}^\infty$$
$$max^\infty(n_1, n_2) := \text{ if } n_2 = \infty \text{ then } \infty \text{ else } \max(n_1, n_2)$$

$$+^\infty : \mathbb{N}^\infty \times \mathbb{N}^\infty \rightarrow \mathbb{N}^\infty$$
$$n_1 +^\infty n_2 := \text{ if } n_1 = \infty \vee n_2 = \infty \text{ then } \infty \text{ else } n_1 + n_2$$

$$-^\infty : \mathbb{N}^\infty \times \mathbb{N} \rightarrow \mathbb{N}^\infty$$
$$n_1 -^\infty n_2 := \text{ if } n_1 = \infty \text{ then } \infty \text{ else } max(0, n_1 - n_2)$$

$$-^\infty : \mathbb{Z}^\infty \rightarrow \mathbb{Z}^\infty$$
$$-^\infty i := \text{ if } i = \infty \text{ then } -\infty \text{ else if } i = -\infty \text{ then } \infty \text{ else} - i$$

$$\mathbb{N} : \mathbb{Z}^\infty \rightarrow \mathbb{N}^\infty$$
$$\mathbb{N}(i) := \text{ if } i = -\infty \vee i < 0 \text{ then } 0 \text{ else } i$$

Figure 3: The Semantic Algebras of the Analysis

ization, together with their definitions ($\mathbb{P}$ stands for the powerset and $\stackrel{\textsf{part.}}{\rightarrow}$ for the partial function):

$$\begin{aligned}
TMonitor &:= TM \textbf{ of } Variable \times TFormula \times \mathbb{P}(TInstance)\\
TInstance &:= \mathbb{N} \times TFormula \times Context\\
Context &:= (Variable \stackrel{\textsf{part.}}{\rightarrow} \mathbb{N}) \times (Variable \stackrel{\textsf{part.}}{\rightarrow} Message)\\
TFormula &:= \textbf{done of } Bool \mid \textbf{next of } TFormulaCore\\
TFormulaCore &:=\\
&TV \textbf{ of } Variable \mid\\
&TN \textbf{ of } TFormula \mid\\
&TCS \textbf{ of } TFormula \times TFormula \mid\\
&TCP \textbf{ of } TFormula \times TFormula \mid\\
&TA \textbf{ of } Variable \times BoundValue \times BoundValue \times TFormula \mid\\
&TA0 \textbf{ of } Variable \times \mathbb{N} \times \mathbb{N}^\infty \times TFormula \mid
\end{aligned}$$

$$TA1 \textsf{ of } Variable \times \mathbb{N}^\infty \times TFormula \times \mathbb{P}(TInstance)$$
$$BoundValue := Context \to \mathbb{N}^\infty$$

Translation. The translation is defined for monitors, formulas, and bounds. Monitors are translated into *TMonitor*'s (translated monitors), formulas are translated into *TFormula*'s (translated formulas), and bounds are translated into *BoundValue*'s:

$$\begin{aligned}
T &: Monitor \to TMonitor\\
T(\texttt{monitor } X \texttt{ : } F) &:= TM(X, T(F), \emptyset)
\end{aligned}$$

$$\begin{aligned}
T &: Formula \to TFormula\\
T(\texttt{@}X) &:= \textsf{next}(TV(X))\\
T(\texttt{\~{}}F) &:= \textsf{next}(TN(T(F)))\\
T(F_1 \texttt{ \&\& } F_2) &:= \textsf{next}(TCS(T(F_1), T(F_2)))\\
T(F_1 \texttt{ /\textbackslash{} } F_2) &:= \textsf{next}(TCP(T(F_1), T(F_2)))\\
T(\texttt{forall } X \texttt{ in } B_1 \texttt{..} B_2 \texttt{ : } F) &:= \textsf{next}(TA(X, T(B_1), T(B_2), T(F)))
\end{aligned}$$

$$\begin{aligned}
T &: Bound \to BoundValue\\
T(0)(c) &:= 0\\
T(\infty)(c) &:= \infty\\
T(X)(c) &:= c.1(X) \text{ if } X \in dom(c.1)\\
T(X)(c) &:= 0 \text{ if } X \notin dom(c.1)\\
T(B+N)(c) &:= T(B)(c) + [\![N]\!]\\
T(B+N)(c) &:= T(B)(c) - [\![N]\!]
\end{aligned}$$

One-Step Operational Semantics. Apart from the quantified position variable X and the translation $f = T(F)$ of the body of the monitor, the representation maintains the set fs of instances of f which for certain values of X could not yet be evaluated to a truth value. The execution of the monitor is formalized by an operational semantics with a small step transition relation $\to_{n,ms,m,rs}$ where n is the index of the next message m arriving on the stream, ms denotes the sequence of messages that have previously arrived (the stream history), and rs denotes the set of those positions for which it can be determined by the current step that they violate the specification. In this step, first a new instance mapping X to the pair (p, m) is created and added to the instance set, and all instances in this set are evaluated; rs becomes the set of positions of those instances yielding "false", the new instance set fs_1 preserves all those instances that could not yet be evaluated to a definite truth value:

$$TMonitor \to_{\mathbb{N}, Message^\omega, Message, \mathbb{P}(\mathbb{N})} TMonitor$$

$$\frac{\begin{array}{l}
fs_0 = fs \cup \{(p, f, [X \mapsto (p, m)])\}\\
rs = \{t \in \mathbb{N} \mid \exists g \in TFormula, c \in Context : (t, g, c) \in fs_0 \wedge\\
\qquad\qquad \vdash g \to_{p,ms,m,c} \textsf{done}(\text{false})\}\\
fs_1 = \{(t, \textsf{next}(fc), c) \in TInstance \mid \exists g \in TFormula : (t, g, c) \in fs_0 \wedge\\
\qquad\qquad \vdash g \to_{p,ms,m,c} \textsf{next}(fc)\}
\end{array}}{TM(X, f, fs) \to_{p,ms,m,rs} TM(X, f, fs_1)}$$

As one can see from this definition, the monitor operation is based on an operational semantics of formula evaluation. The rules for the latter are given below:

$$TFormula \to_{\mathbb{N}, Message^\omega, Message, Context} TFormula$$

Atomic formula:

$$\frac{X \in dom(c.2)}{\textbf{next}(TV(X)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(c.2(X))}$$

$$\frac{X \notin dom(c.2)}{\textbf{next}(TV(X)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}$$

Negation:

$$\frac{f \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f')}{\textbf{next}(TN(f)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(TN(\textbf{next}(f')))}$$

$$\frac{f \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{true})}{\textbf{next}(TN(f)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}$$

$$\frac{f \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}{\textbf{next}(TN(f)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{true})}$$

Sequential Conjunction:

$$\frac{f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f_1')}{\textbf{next}(TCS(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(TCS(\textbf{next}(f_1'), f_2)}$$

$$\frac{f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}{\textbf{next}(TCS(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}$$

$$\frac{\begin{array}{l} f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{true}) \\ f_2 \rightarrow_{(p,\, ms,\, m,\, c)} f_2' \end{array}}{\textbf{next}(TCS(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} f_2'}$$

Parallel Conjunction:

$$\frac{\begin{array}{l} f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f_1') \\ f_2 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f_2') \end{array}}{\textbf{next}(TCP(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(TCP(\textbf{next}(f_1'), \textbf{next}(f_2'))}$$

$$\frac{\begin{array}{l} f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f_1') \\ f_2 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{true}) \end{array}}{\textbf{next}(TCP(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f_1')}$$

$$\frac{\begin{array}{l} f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{next}(f_1') \\ f_2 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false}) \end{array}}{\textbf{next}(TCP(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}$$

$$\frac{f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}{\textbf{next}(TCP(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{false})}$$

$$\frac{\begin{array}{l} f_1 \rightarrow_{(p,\, ms,\, m,\, c)} \textbf{done}(\text{true}) \\ f_2 \rightarrow_{(p,\, ms,\, m,\, c)} f_2' \end{array}}{\textbf{next}(TCP(f_1, f_2)) \rightarrow_{(p,\, ms,\, m,\, c)} f_2'}$$

Universal Quantification:

$$\frac{\begin{array}{l} p_1 = b_1(c) \\ p_p = b_2(c) \\ p_1 = \infty \vee p_1 >^\infty p_2 \end{array}}{\textsf{next}(\mathit{TA}(X, b_1, b_2, f)) \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{done}(\text{true})}$$

$$\frac{\begin{array}{l} p_1 = b_1(c) \\ p_2 = b_2(c) \\ p_1 \neq \infty \wedge p_1 \leq^\infty p_2 \\ \textsf{next}(\mathit{TA0}(X, p_1, p_2, f)) \rightarrow_{(p,\, ms,\, m,\, c)} \mathit{TA0}' \end{array}}{\textsf{next}(\mathit{TA}(X, b_1, b_2, f)) \rightarrow_{(p,\, ms,\, m,\, c)} \mathit{TA0}'}$$

$$\frac{p < p_1}{\textsf{next}(\mathit{TA0}(X, p_1, p_2, f)) \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{next}(\mathit{TA0}(X, p_1, p_2, f))}$$

$$\frac{\begin{array}{l} p \geq p_1 \\ \mathit{fs} = \{(p_0, f, (c.1[X \mapsto p_0], c.2[X \mapsto ms(p_0 + p - |ms|)])) \mid p_1 \leq p_0 <^\infty \min^\infty(p, p_2 +^\infty 1)\} \\ \textsf{next}(\mathit{TA1}(X, p_2, f, \mathit{fs})) \rightarrow_{(p,\, ms,\, m,\, c)} \mathit{TA1}' \end{array}}{\textsf{next}(\mathit{TA0}(X, p_2, f, \mathit{fs})) \rightarrow_{(p,\, ms,\, m,\, c)} \mathit{TA1}'}$$

$$\frac{\begin{array}{l} \mathit{fs}_0 = \textsf{if } p >^\infty p_2 \textsf{ then } \mathit{fs} \textsf{ else } \mathit{fs} \cup \{(p, f, (c.1[X \mapsto p], c.2[X \mapsto m]))\} \\ \exists t \in \mathbb{N}, g \in \mathit{TFormula}, c \in \mathit{Context} : (t, g, c) \in \mathit{fs}_0 \wedge \vdash g \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{done}(\text{false}) \end{array}}{\textsf{next}(\mathit{TA1}(X, p_2, f, \mathit{fs})) \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{done}(\text{false})}$$

$$\frac{\begin{array}{l} \mathit{fs}_0 = \textsf{if } p >^\infty p_2 \textsf{ then } \mathit{fs} \textsf{ else } \mathit{fs} \cup \{(p, f, (c.1[X \mapsto p], c.2[X \mapsto m]))\} \\ \neg\exists t \in \mathbb{N}, g \in \mathit{TFormula}, c \in \mathit{Context} : (t, g, c) \in \mathit{fs}_0 \wedge \vdash g \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{done}(\text{false}) \\ \mathit{fs}_1 = \{(t, \textsf{next}(\mathit{fc}), c) \in \mathit{TInstance} \mid \\ \qquad\qquad \exists g \in \mathit{TFormula} : (t, g, c) \in \mathit{fs}_0 \wedge \vdash g \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{next}(\mathit{fc})\} \\ \mathit{fs}_1 = \emptyset \wedge p \geq^\infty p_2 \end{array}}{\textsf{next}(\mathit{TA1}(X, p_2, f, \mathit{fs})) \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{done}(\text{true})}$$

$$\frac{\begin{array}{l} \mathit{fs}_0 = \textsf{if } p >^\infty p_2 \textsf{ then } \mathit{fs} \textsf{ else } \mathit{fs} \cup \{(p, f, (c.1[X \mapsto p], c.2[X \mapsto m]))\} \\ \neg\exists t \in \mathbb{N}, g \in \mathit{TFormula}, c \in \mathit{Context} : (t, g, c) \in \mathit{fs}_0 \wedge \vdash g \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{done}(\text{false}) \\ \mathit{fs}_1 = \{(t, \textsf{next}(\mathit{fc}), c) \in \mathit{TInstance} \mid \\ \qquad\qquad \exists g \in \mathit{TFormula} : (t, g, c) \in \mathit{fs}_0 \wedge \vdash g \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{next}(\mathit{fc})\} \\ \neg(\mathit{fs}_1 = \emptyset \wedge p \geq^\infty p_2) \end{array}}{\textsf{next}(\mathit{TA1}(X, p_2, f, \mathit{fs})) \rightarrow_{(p,\, ms,\, m,\, c)} \textsf{next}(\mathit{TA1}(X, p_2, f, \mathit{fs}_1))}$$

Finally, we give definitions of n-step reduction. There are for versions: right- and left-recursive with and without history.

Definition 1 (Right-Recursive n-Step Reduction)**.**

Without history. $\mathit{TFormula} \rightarrow^*_{(\mathbb{N}, \mathbb{N}, \mathit{Stream}, \mathit{Environment})} \mathit{TFormula}$, where the first $\mathbb{N}$ is the number of steps and the second $\mathbb{N}$ is the current position.

$$\mathit{Ft} \rightarrow^*_{(0,p,s,e)} \mathit{Ft} \qquad \frac{\begin{array}{l} n > 0 \\ c = (e, \{(X, s(e(X))) \mid X \in \mathit{dom}(e)\}) \\ \mathit{Ft} \rightarrow_{(p,\, s{\downarrow}p,\, s(p),\, c)} \mathit{Ft}' \\ \mathit{Ft}' \rightarrow^*_{(n-1,p+1,s,e)} \mathit{Ft}'' \end{array}}{\mathit{Ft}' \rightarrow^*_{(n,p,s,e)} \mathit{Ft}''}$$

With history. $\mathit{TFormula} \rightarrow^*_{(\mathbb{N}, \mathbb{N}, \mathit{Stream}, \mathit{Environment}, \mathit{Message}^*)} \mathit{TFormula}$, where the first $\mathbb{N}$ is the number of steps, the second $\mathbb{N}$ is the current position, and $Message^*$ is the history.

$$Ft \to^*_{(0,p,s,e,h)} Ft \qquad \frac{\begin{array}{l} n > 0 \\ c = (e, \{(X, s(e(X))) \mid X \in dom(e)\}) \\ Ft \to_{(p,\, s\uparrow(\max(0,p-h),\min(p,h)),\, s(p),\, c)} Ft' \\ Ft' \to^*_{(n-1,p+1,s,e,h)} Ft'' \end{array}}{Ft' \to^*_{(n,p,s,e,h)} Ft''}$$

Definition 2 (Left-Recursive n-Step Reduction)**.**

Without history. $TFormula \to^{l*}_{(\mathbb{N},\mathbb{N},Stream,Environment)} TFormula$, where the first $\mathbb{N}$ is the number of steps and the second $\mathbb{N}$ is the current position.

$$Ft \to^{l*}_{(0,p,s,e)} Ft \qquad \frac{\begin{array}{l} n > 0 \\ Ft \to^{l*}_{(n-1,p,s,e)} Ft' \\ c = (e, \{(X, s(e(X))) \mid X \in dom(e)\}) \\ Ft' \to_{(p+n-1,\, s\downarrow(p+n-1),\, s(p+n-1),\, c)} Ft'' \end{array}}{Ft' \to^{l*}_{(n,p,s,e)} Ft''}$$

With history. $TFormula \to^{l*}_{(\mathbb{N},\mathbb{N},Stream,Environment,Message^*)} TFormula$, where the first $\mathbb{N}$ is the number of steps, the second $\mathbb{N}$ is the current position, and $Message^*$ is the history.

$$Ft \to^{l*}_{(0,p,s,e,h)} Ft \qquad \frac{\begin{array}{l} n > 0 \\ Ft \to^{l*}_{(n-1,p,s,e,h)} Ft' \\ c = (e, \{(X, s(e(X))) \mid X \in dom(e)\}) \\ Ft' \to_{(p+n-1,\, s\uparrow(\max(0,p+n-1-h),\min(p+n-1,h)),\, s(p+n-1),\, c)} Ft'' \end{array}}{Ft \to^{l*}_{(n,p,s,e,h)} Ft''}$$

4 Soundness of Resource Analysis

In this section we formulate the main result:

Theorem 1 (Soundness of Resource Analysis for Monitors)**.** *The resource analysis of the core monitor language is sound with respect to its operational semantics, i.e., if the analysis yields for monitor M natural numbers h and d, then the execution does not maintain more than d monitor instances and does not require more than the last h messages from the stream. Formally:*

$$\begin{array}{l} \forall X, Y \in Variable,\ F \in Formula,\ Ft \in TFormula,\ It \in \mathbb{P}(Instance),\ n \in \mathbb{N},\ s \in Stream, \\ \quad rs \in \mathbb{P}(\mathbb{N}),\ h, d \in \mathbb{N}^\infty : \\ \quad \textbf{let } M = \texttt{monitor } X : F,\ Mt = TM(Y, Ft, It) : \\ \qquad \vdash M : (h, d) \Rightarrow \\ \qquad\quad (d \in \mathbb{N} \Rightarrow (\vdash T(M) \to^*_{n,s,rs} Mt \Rightarrow |It| \le d)) \wedge \\ \qquad\quad (h \in \mathbb{N} \Rightarrow (\vdash T(M) \to^*_{n,s,rs} Mt \Leftrightarrow \vdash T(M) \to^*_{n,s,rs,h} Mt)). \end{array}$$

The proof of this theorem uses three lemmas and a statement about an invariant of n-step reductions of translated monitors. These propositions, for their part, rely on additional lemmas. Dependencies between these statements, which give an idea of the high-level proof structure, are shown in Fig. 4. Below we formulate these lemmas with some informal explanations. The complete proofs can be found in the appendix.

The Invariant Statement asserts essentially the following: For a monitor M (with the monitoring variable X and the monitored formula F), if the analysis yields natural numbers h and d, and the translated version of M reduces to another translated monitor $TM(Y, Ft, It)$ in n steps, then the following invariant holds:

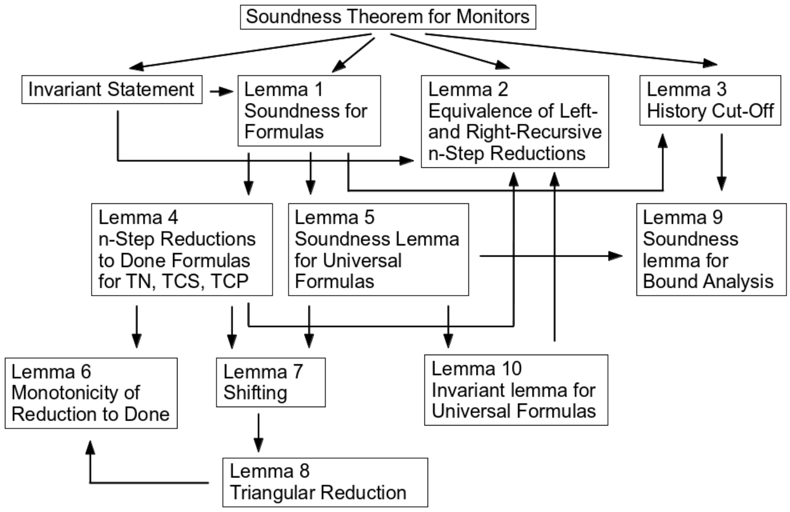

Figure 4: Lemma dependencies in the proof of the Soundness Theorem.

- X and Y are the same and Ft is the translation of F,
- all elements in the set of instances It contain **next** formulas, which have been generated at different steps in the past, but not earlier than d units before from the current step,
- the formulas in the elements of It are obtained by reductions of $T(F)$, and they themselves will reduce to a **done** formula in at most d steps from the moment of their creation.

More formally, the invariant definition looks as follows:

Definition 3 (Invariant)**.**

$$
\begin{array}{l}
\forall X, Y \in \mathit{Variable}, F \in \mathit{Formula}, Ft \in \mathit{TFormula}, It \in \mathbb{P}(\mathit{TInstance}),\\
\quad n \in \mathbb{N}, s \in \mathit{Stream}, d \in \mathbb{N}^{\infty} :\\
\quad\quad \mathit{invariant}(X, Y, F, Ft, It, n, s, d) :\Leftrightarrow\\
\quad\quad\quad X = Y \wedge Ft = T(F) \wedge \mathit{alldiff}(It) \wedge \mathit{allnext}(It) \wedge\\
\quad\quad\quad \forall t \in \mathbb{N}, Ft' \in \mathit{TFormula}, c \in \mathit{Context} :\\
\quad\quad\quad\quad (t, Ft', c) \in It \wedge d \in \mathbb{N} \Rightarrow\\
\quad\quad\quad\quad\quad c.1 = \{(X, t)\} \wedge c.2 = \{(X, s(t))\} \wedge\\
\quad\quad\quad\quad\quad n - d \le t \le n - 1 \wedge\\
\quad\quad\quad\quad\quad T(F) \rightarrow^{*}_{n-t,t,s,c.1} Ft' \wedge\\
\quad\quad\quad\quad\quad \exists b \in \mathit{Bool}, d' \in \mathbb{N} :\\
\quad\quad\quad\quad\quad\quad d' \le d \wedge \vdash Ft' \rightarrow^{*}_{\max(0,t+d'-n),n,s,c.1} \textbf{done}(b),
\end{array}
$$

where $\mathit{alldiff}(It)$ means that $t_1 \neq t_2$ for all distinct elements (t_1, Ft_1, c_1), (t_2, Ft_2, c_2) of It, and $\mathit{allnext}(It)$ denotes the fact that for all $(t, Ft, c) \in It$, Ft is a **next** formula.

Then the Invariant Statement is formulated in the following way:

Proposition 1 (Invariant Statement)**.**

$$
\begin{array}{l}
\forall X \in \mathit{Variable}, F \in \mathit{Formula}, h \in \mathbb{N}^{\infty}, d \in \mathbb{N}^{\infty}, n \in \mathbb{N}, s \in \mathit{Stream}, \\
\quad rs \in \mathbb{P}(\mathbb{N}), Y \in \mathit{Variable}, \mathit{Ft} \in \mathit{TFormula}, \mathit{It} \in \mathbb{P}(\mathit{TInstance}) : \\
\quad\quad \vdash (\texttt{monitor}\ X : F) : (h, d) \wedge \\
\quad\quad \vdash T(\texttt{monitor}\ X : F) \rightarrow^{*}_{n,s,rs} \mathit{TM}(Y, \mathit{Ft}, \mathit{It}) \Rightarrow \\
\quad\quad\quad \mathit{invariant}(X, Y, F, \mathit{Ft}, \mathit{It}, n, s, d)
\end{array}
$$

In the course of proving the Soundness Statement, the reasoning moves from the monitor level to the formula level. Therefore, we need a counterpart of the Soundness Theorem (which is formulated for monitors) for formulas. This is the first Lemma.

Lemma 1 (Soundness Lemma for Formulas)**.**

$$
\begin{array}{l}
\forall F, F' \in \mathit{Formula}, \mathit{re} \in \mathit{RangeEnv}, e \in \mathit{Environment}, \mathit{Ft} \in \mathit{TFormula}, n, p \in \mathbb{N}, \\
\quad s \in \mathit{Stream}, d \in \mathbb{N}^{\infty}, h \in \mathbb{N} : \\
\quad\quad \vdash (\mathit{re} \vdash F : (h, d)) \wedge \mathit{dom}(e) = \mathit{dom}(\mathit{re}) \wedge \\
\quad\quad \forall Y \in \mathit{dom}(e) : \mathit{re}(Y).1 + p \le e(Y) \le \mathit{re}(Y).2 + p \Rightarrow \\
\quad\quad\quad (d \in \mathbb{N} \Rightarrow \\
\quad\quad\quad\quad \exists b \in \mathit{Bool}, d' \in \mathbb{N} : \\
\quad\quad\quad\quad\quad d' \le d + 1 \wedge \vdash T(F) \rightarrow^{*}_{d',p,s,e} \textsf{done}(b)) \wedge \\
\quad\quad\quad (\forall h' \in \mathbb{N} : h' \ge h \Rightarrow \\
\quad\quad\quad\quad (T(F) \rightarrow^{*}_{n,p,s,e} \mathit{Ft} \Leftrightarrow T(F) \rightarrow^{*}_{n,p,s,e,h'} \mathit{Ft})).
\end{array}
$$

The second lemma states equivalence of left- and right-recursive definitions of n-step reductions. This is a technical result which helps to simplify proofs of the Soundness Theorem, Invariant Statement, and Lemma 4 and Lemma 10 below.

Lemma 2 (Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions)**.**

(a) $\forall n, p \in \mathbb{N}, s \in \mathit{Stream}, e \in \mathit{Environment}, \mathit{Ft}_1, \mathit{Ft}_2 \in \mathit{TFormula} :$
$\quad \mathit{Ft}_1 \rightarrow^{*}_{n,p,s,e} \mathit{Ft}_2 \Leftrightarrow \mathit{Ft}_1 \rightarrow^{l*}_{n,p,s,e} \mathit{Ft}_2.$

(b) $\forall n, p \in \mathbb{N}, s \in \mathit{Stream}, e \in \mathit{Environment}, \mathit{Ft}_1, \mathit{Ft}_2 \in \mathit{TFormula}, h \in \mathbb{N} :$
$\quad \mathit{Ft}_1 \rightarrow^{*}_{n,p,s,e,h} \mathit{Ft}_2 \Leftrightarrow \mathit{Ft}_1 \rightarrow^{l*}_{n,p,s,e,h} \mathit{Ft}_2.$

The next lemma establishes the limit on the number of past messages needed for a single monitoring step to be equivalent to such a step performed with the full history. Both the Soundness Theorem and the Soundness Lemma use it.

Lemma 3 (History Cut-Off Lemma)**.**

$$
\begin{array}{l}
\forall F \in \mathit{Formula}, \mathit{Ft} \in \mathit{TFormula}, p \in \mathbb{N}, s \in \mathit{Stream}, h \in \mathbb{N}, d \in \mathbb{N}^{\infty}, \\
\quad e \in \mathit{Environment}, \mathit{re} \in \mathit{RangeEnv} : \\
\quad\quad \vdash (\mathit{re} \vdash F : (h, d)) \wedge \mathit{dom}(e) = \mathit{dom}(\mathit{re}) \wedge \\
\quad\quad \forall Y \in \mathit{dom}(e) : \mathit{re}(Y).1 + p \le e(Y) \le \mathit{re}(Y).2 + p \Rightarrow \\
\quad\quad \textsf{let}\ c := (e, \{(X, s(e(X))) \mid X \in \mathit{dom}(e)\}) : \\
\quad\quad\quad \forall h' \in \mathbb{N} : h' \ge h \Rightarrow \\
\quad\quad\quad\quad T(F) \rightarrow_{p,s\downarrow p,s(p),c} \mathit{Ft} \\
\quad\quad\quad\quad \Leftrightarrow \\
\quad\quad\quad\quad T(F) \rightarrow_{p,s\uparrow(\max(0,p-h'),\min(p,h')),s(p),c} \mathit{Ft}
\end{array}
$$

The Soundness Lemma for Formulas requires yet two auxiliary propositions. The first of them, Lemma 4 below, establishes the conditions of reduction of translated TN (negation), TCS (sequential conjunction), and TCP (parallel conjunction) formulas into **done** formulas:

Lemma 4 (n-Step Reductions to **done** Formulas for TN, TCS, TCP)**.**

Statement 1. ***TN* Formulas:**

$$\forall F \in \mathit{Formula}, n, p \in \mathbb{N}, s \in \mathit{Stream}, e \in \mathit{Environment}, \mathit{Ft} \in \mathit{TFormula} :$$

$$\begin{aligned}
&T(F) \to^*_{n,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \Rightarrow \textsf{\textbf{next}}(\mathit{TN}(T(F))) \to^*_{n,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \wedge \\
&T(F) \to^*_{n,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \Rightarrow \textsf{\textbf{next}}(\mathit{TN}(T(F))) \to^*_{n,p,s,e} \textsf{\textbf{done}}(\textsf{false})
\end{aligned}$$

Statement 2. ***TCS* Formulas:**

$$\forall p \in \mathbb{N}, s \in \mathit{Stream}, e \in \mathit{Environment} :$$

$$\begin{aligned}
&\forall \mathit{Ft}_1, \mathit{Ft}_2 \in \mathit{TFormula}, n \in \mathbb{N} : \\
&\quad n > 0 \wedge \mathit{Ft}_1 \to^*_{n,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \Rightarrow \\
&\quad\quad \textsf{\textbf{next}}(\mathit{TCS}(\mathit{Ft}_1, \mathit{Ft}_2)) \to^*_{n,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \wedge \\
&\forall \mathit{Ft}_1, \mathit{Ft}_2 \in \mathit{TFormula}, n_1, n_2 \in \mathbb{N}, b \in \mathit{Bool} : \\
&\quad n_1 > 0 \wedge n_2 > 0 \wedge \mathit{Ft}_1 \to^*_{n_1,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \wedge \mathit{Ft}_2 \to^*_{n_2,p,s,e} \textsf{\textbf{done}}(b) \Rightarrow \\
&\quad\quad \textsf{\textbf{next}}(\mathit{TCS}(\mathit{Ft}_1, \mathit{Ft}_2)) \to^*_{\max(n_1,n_2),p,s,e} \textsf{\textbf{done}}(b)
\end{aligned}$$

Statement 3. ***TCP* Formulas:**

$$\forall p \in \mathbb{N}, s \in \mathit{Stream}, e \in \mathit{Environment}, \mathit{Ft}_1, \mathit{Ft}_2 \in \mathit{TFormula}, n_1, n_2 \in \mathbb{N} :$$

$$\begin{aligned}
&n_1 > 0 \wedge \mathit{Ft}_1 \to^*_{n_1,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \wedge \mathit{Ft}_2 \to^*_{n_2,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \Rightarrow \\
&\quad \textsf{\textbf{next}}(\mathit{TCP}(\mathit{Ft}_1, \mathit{Ft}_2)) \to^*_{n_1,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \\
&\wedge \\
&n_1 > 0 \wedge n_2 > 0 \wedge \mathit{Ft}_1 \to^*_{n_1,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \wedge \mathit{Ft}_2 \to^*_{n_2,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \Rightarrow \\
&\quad \textsf{\textbf{next}}(\mathit{TCP}(\mathit{Ft}_1, \mathit{Ft}_2)) \to^*_{\min(n_1,n_2),p,s,e} \textsf{\textbf{done}}(\textsf{false}) \\
&\wedge \\
&n_1 > 0 \wedge n_2 > 0 \wedge \mathit{Ft}_1 \to^*_{n_1,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \wedge \mathit{Ft}_2 \to^*_{n_2,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \Rightarrow \\
&\quad \textsf{\textbf{next}}(\mathit{TCP}(\mathit{Ft}_1, \mathit{Ft}_2)) \to^*_{\max(n_1,n_2),p,s,e} \textsf{\textbf{done}}(\textsf{true}) \\
&\wedge \\
&n_1 > 0 \wedge n_2 > 0 \wedge \mathit{Ft}_1 \to^*_{n_1,p,s,e} \textsf{\textbf{done}}(\textsf{true}) \wedge \mathit{Ft}_2 \to^*_{n_2,p,s,e} \textsf{\textbf{done}}(\textsf{false}) \Rightarrow \\
&\quad \textsf{\textbf{next}}(\mathit{TCP}(\mathit{Ft}_1, \mathit{Ft}_2)) \to^*_{n_2,p,s,e} \textsf{\textbf{done}}(\textsf{false})
\end{aligned}$$

The other auxiliary statement needed in the proof of Lemma 1 is Lemma 5 below, which formulates a special case of the soundness statement for universally quantified formulas.

Lemma 5 (Soundness Lemma for Universal Formulas)**.**

$$\forall F \in \mathit{Formula},\ X \in \mathit{Variable},\ B_1, B_2 \in \mathit{Bound} :$$

$$R(F) \Rightarrow R(\texttt{forall}\ X\ \texttt{in}\ B_1\,\texttt{..}\,B_2\ \texttt{:}\ F)$$

where

$$\begin{aligned}
&R(F) :\Leftrightarrow \\
&\quad \forall \mathit{re} \in \mathit{RangeEnv},\ e \in \mathit{Environment}, s \in \mathit{Stream},\ d \in \mathbb{N}^\infty,\ h \in \mathbb{N}\, p \in \mathbb{N} : \\
&\quad\quad \vdash (\mathit{re} \vdash F : (h, d)) \wedge d \in \mathbb{N} \wedge \mathit{dom}(e) = \mathit{dom}(\mathit{re}) \wedge \\
&\quad\quad \forall Y \in \mathit{dom}(e) : \mathit{re}(Y).1 + p \le e(Y) \le \mathit{re}(Y).2 + p \Rightarrow \\
&\quad\quad\quad (\exists b \in \mathit{Bool}, d' \in \mathbb{N} : d' \le d + 1 \wedge \vdash T(F) \to^*_{d',p.s.e} \textsf{\textbf{done}}(b))
\end{aligned}$$

Proving of Lemma 4 requires a couple of other statements. Besides Lemma 2 above, there are two other lemmas: for monotonicity (Lemma 6) and for shifting (Lemma 7). The Monotonicity Lemma states that if a translated formula reduces to a **done** formula, then starting from that moment on it will always reduce to the same **done** formula:

Lemma 6 (Monotonicity of Reduction to **done**).

$$\forall Ft \in \mathit{TFormula},\ p, k \in \mathbb{N},\ s \in \mathit{Stream},\ c \in \mathit{Context},\ b \in \mathit{Bool}:$$
$$k \geq p \Rightarrow Ft \rightarrow_{p,s\downarrow p,s(p),c} \textsf{done}(b) \Rightarrow Ft \rightarrow_{k,s\downarrow(k),s(k),c} \textsf{done}(b).$$

The Shifting Lemma expresses a simple fact: If a **next** formula reduced to a **done** formula in $n+1$ steps starting from the stream position p, then the same reduction will take n steps if it starts at position $p+1$:

Lemma 7 (Shifting Lemma).

$$\forall f \in \mathit{TFormulaCore},\ n, p \in \mathbb{N},\ s \in \mathit{Stream},\ e \in \mathit{Environment},\ b \in \mathit{Bool}:$$
$$n > 0 \Rightarrow \textsf{next}(f) \rightarrow^{*}_{n+1,p,s,e} \textsf{done}(b) \Rightarrow \textsf{next}(f) \rightarrow^{*}_{n,p+1,s,e} \textsf{done}(b).$$

Lemma 7 requires a so called Triangular Reduction Lemma, shown below. The latter, for itself, relies on Lemma 6.

Lemma 8 (Triangular Reduction Lemma).

$$\forall f_1, f_2 \in \mathit{TFormulaCore},\ Ft \in \mathit{TFormula},\ p \in \mathbb{N},\ s \in \mathit{Stream},\ c \in \mathit{Context}:$$
$$\textsf{next}(f_1) \rightarrow_{p,s\downarrow p,s(p),c} \textsf{next}(f_2) \wedge \textsf{next}(f_2) \rightarrow_{p+1,s\downarrow(p+1),s(p+1),c} Ft \Rightarrow$$
$$\quad \textsf{next}(f_1) \rightarrow_{p+1,s\downarrow(p+1),s(p+1),c} Ft.$$

Proving Lemma 5 is more involved. It relies on three statements: the already mentioned Shifting Lemma (Lemma 7), Soundness Lemma for Bound Analysis (Lemma 9), and the Invariant Lemma for Universal Formulas (Lemma 10). The proof of Lemma 3 also use Lemma 9.

Lemma 9 (Soundness Lemma for Bound Analysis).

$$\forall re \in \mathit{RangeEnv},\ e \in \mathit{Environment},\ p \in \mathbb{N},\ s \in \mathit{Stream},\ B \in \mathit{Bound},\ l, u \in \mathbb{Z}^{\infty}:$$
$$\begin{array}{l} re \vdash B : (l, u) \wedge dom(e) = dom(re) \wedge \\ \forall Y \in dom(e) : re(Y).1 + p \leq e(Y) \leq re(Y).2 + p \Rightarrow \\ \quad \textsf{let}\ c := (e, \{(X, s(e(X))) \mid X \in dom(e)\}) : \\ \quad l + p \leq T(B)(c) \leq u + p. \end{array}$$

Finally, the Invariant Lemma for Universal Formulas has the following form:

Lemma 10 (Invariant Lemma for Universal Formulas).

$$\forall X \in \mathit{Variable},\ b_1, b_2 \in \mathit{BoundValue},\ f \in \mathit{TFormulaCore}:$$
$$\forall n \in \mathbb{N} : n \geq 1 \Rightarrow \mathit{forall}(n, X, b_1, b_2, \textsf{next}(f))$$

The predicate *forall* in this lemma is defined below:

$$\begin{array}{l} \mathit{forall} \subseteq \mathbb{N} \times \mathit{Variable} \times \mathit{BoundValue} \times \mathit{BoundValue} \times \mathit{TFormula}: \\ \mathit{forall}(n, X, b_1, b_2, f) :\Leftrightarrow \\ \quad \forall p \in \mathbb{N},\ s \in \mathit{Stream},\ e \in \mathit{Environment},\ g \in \mathit{TFormula}: \\ \quad\quad \vdash \textsf{next}(\mathit{TA}(X, b_1, b_2, f)) \rightarrow^{*}_{n,p,s,e} g \Rightarrow \\ \quad\quad\quad \textsf{let}\ c = (e, \{(Y, s(e(Y))) \mid Y \in dom(e)\}),\ p_0 = p + n,\ p_1 = b_1(c),\ p_2 = b_2(c): \\ \quad\quad\quad \big(n = 1 \wedge (p_1 = \infty \vee p_1 >^{\infty} p_2) \wedge g = \textsf{done}(\mathsf{true})\big) \bigvee \\ \quad\quad\quad \big(n \geq 1 \wedge p_1 \neq \infty \wedge p_1 \leq^{\infty} p_2 \wedge p_0 \leq p_1 \wedge g = \textsf{next}(\mathit{TA0}(X, p_1, p_2, f))\big) \bigvee \\ \quad\quad\quad \big(n \geq 1 \wedge p_1 \neq \infty \wedge p_1 \leq^{\infty} p_2 \wedge p_0 > p_1 \wedge \end{array}$$

$$
\begin{aligned}
&(\exists b \in \mathit{Bool} : g = \textsf{\textbf{done}}(b)) \lor \\
&\quad (\exists gs \in \mathbb{P}(\mathit{TInstance}) : (gs \neq \emptyset \lor p + n \leq^{\infty} p_2) \land \\
&\quad \mathit{forallInstances}(X, p, p_0, p_1, p_2, f, s, e, gs) \land g = \textsf{\textbf{next}}(\mathit{TA1}(X, p_2, f, gs)))\big),
\end{aligned}
$$

where the predicate *forallInstances* is defined as follows:

$$
\begin{aligned}
&\mathit{forallInstances} \subseteq \\
&\quad \mathit{Variable} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}^{\infty} \times \mathit{TFormula} \times \mathit{Stream} \times \mathit{Environment} \times \mathbb{P}(\mathit{TInstance}) : \\
&\mathit{forallInstances}(X, p, p_0, p_1, p_2, f, s, e, gs) :\Leftrightarrow \\
&\quad \forall t \in \mathbb{N},\ g \in \mathit{TFormula},\ c_0 \in \mathit{Context} : (t, g, c_0) \in gs \Rightarrow \\
&\qquad \big(\forall t_1 \in \mathbb{N},\ g_1 \in \mathit{TFormula},\ c_1 \in \mathit{Context} : (t_1, g_1, c_1) \in gs \land t = t_1 \Rightarrow \\
&\qquad\quad (t, g, c_0) = (t_1, g_1, c_1)\big) \land \\
&\qquad (\exists gc \in TFormulaCore : g = \mathit{next}(gc)) \land \\
&\qquad\quad c_0.1 = e[X \mapsto t] \land c_0.2 = \{(Y, s(c_0.1(Y))) \mid Y \in \mathit{dom}(e) \lor Y = X\} \land \\
&\qquad\quad p_1 \leq t \leq^{\infty} \mathit{min}^{\infty}(p_0 - 1, p_2) \land \vdash f \rightarrow^{*}_{p_0 - \max(p,t), \max(p,t), s, c_0.1} g
\end{aligned}
$$

5 Conclusion

The goal of resource analysis of the core LogicGuard language is two-fold: To determine the maximal size of the stream history required to decide a given instance of the monitor formula, and to determine the maximal delay in deciding a given instance. Ultimately, it determines whether a specification expressed in this language gives rise to a monitor that can operate with a finite amount of resources. This report presents propositions needed to prove soundness of resource analysis of the core LogicGuard language with respect to the operational semantics.

Acknowledgments

The authors thank the project partner companies: SecureGuard GmbH and RISC Software GmbH.

References

[1] Temur Kutsia and Wolfgang Schreiner. LogicGuard Abstract Language. RISC Report Series 12-08, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, 2012.

[2] Temur Kutsia and Wolfgang Schreiner. Translation Mechanism for the LogicGuard Abstract Language. RISC Report Series 12-11, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, 2012.

[3] Temur Kutsia and Wolfgang Schreiner. Verifying the Soundness of Resource Analysis for LogicGuard Monitors, Part 1. Technical report, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, December 16 2013.

[4] Wolfgang Schreiner. Generating network monitors from logic specifications. Invited Talk at FIT 2012, 10th International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 2012.

[5] Wolfgang Schreiner. Applying predicate logic to monitoring network traffic. Invited talk at PAS 2013 - Second International Seminar on Program Verification, Automated Debugging and Symbolic Computation, Beijing, China, October 23–25, 2013.

[6] Wolfgang Schreiner and Temur Kutsia. A Resource Analysis for LogicGuard Monitors. Technical report, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, December 17, 2013.

A Proofs

A.1 Theorem 1: Soundness Theorem

```
∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, n∈ℕ, s∈Stream, rs∈ℙ(ℕ),
  Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
  let M = monitor X : F, Mt = TM(Y,Ft,It) :
  ⊢ M: (h,d) ⇒
    (d∈ℕ ⇒ (⊢ T(M) →*(n,s,rs) Mt ⇒ |It| ≤  d)) ∧
    (h∈ℕ ⇒ (⊢ T(M) →*(n,s,rs) Mt ⇔ ⊢ T(M) →*(n,s,rs,h) Mt))
```

PROOF:

We split the soundness statement into two formulas:

```
(a) ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, n∈ℕ, s∈Stream, rs∈ℙ(ℕ),
       Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
      let M = monitor X : F, Mt = TM(Y,Ft,It) :
      ⊢ M: (h,d) ⇒
        (d∈ℕ ⇒ (⊢ T(M) →*(n,s,rs) Mt ⇒ |It| ≤  d))
```

and

```
(b) ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, n∈ℕ, s∈Stream, rs∈ℙ(ℕ),
      Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
      let M = monitor X : F, Mt = TM(Y,Ft,It) :
      ⊢ M: (h,d) ⇒
           (h∈ℕ ⇒ (⊢ T(M) →*(n,s,rs) Mt ⇔ ⊢ T(M) →*(n,s,rs,h) Mt))
```

Proof of (a)

We take Xf, Ff, Yf, Ftf, Itf, hf, df, nf, sf, rsf arbitrary buf fixed.

Assume

```
(1) ⊢ (monitor Xf : Ff): (hf,df)
(2) df∈ℕ
(3) T(monitor Xf : Ff) →*(nf,sf,rsf) TM(Yf,Ftf,Itf)
```

Prove

```
[4] |Itf| ≤ df
```

From (1,2,3), we know that

```
(5) invariant(Xf,Yf,Ff,Ftf,Itf,nf,sf,df)
```

holds. That means, we know

```
(6) Xf = Yf
(7) Ftf = T(Ff)
```

```
(8) alldiffs(Itf)
(9) allnext(Itf)
(10) ∀ t∈ℕ, Ft∈TFormula, c∈Context:
     (t,Ft,c) ∈ Itf ⇒
      c.1={(Xf,t)} ∧ c.2={(Xf,sf(t))} ∧
      T(Ff) →* (n-t,t,s,c.1) Ft1 ∧
      nf-df ≤ t ≤ nf-1 ∧
      ∃b∈Bool ∃d'∈ℕ :
         d'≤df ∧ ⊢ Ft →*(max(0,t+df'-nf),nf,sf,c.1) done(b)
```

From (10), we know that the tags of the elements of Itf are between nf-df and nf-1 inclusive. From (8), we know that no two elements of Itf have the same tag. Hence, Itf can contain at most (nf-1)-(nf-df)+1 = df elements. Hence, (5) holds.

Proof of (b)

Parametrization:

```
Q(n) :⇔
  ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
     Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
     let M = monitor X : F, Mt = TM(Y,Ft,It) :
     ⊢ M: (h,d) ⇒
           (h∈ℕ ⇒ (⊢ T(M) →*(n,s,rs) Mt ⇔ ⊢ T(M) →*(n,s,rs,h) Mt))
```

We want to show

∀n∈ℕ: Q(n).

For this is suffices to show

1. Q(0)
2. ∀n∈ℕ: Q(n) ⇒ Q(n+1)

Proof of 1

Q(0)

```
∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
   Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
   let M = monitor X : F, Mt = TM(Y,Ft,It) :
     ⊢ M: (h,d) ⇒
           (h∈ℕ ⇒ (⊢ T(M) →*(0,s,rs) Mt ⇔ ⊢ T(M) →*(0,s,rs,h) Mt))
```

We take Xf, Ff, Yf, Ftf, cf, Itf, df, hf, sf, rsf arbitrary buf fixed.

Assume

(1) ⊢ (monitor Xf : Ff): (hf,df)

(2) hf∈ℕ

Prove

```
[3] ⊢ T(monitor Xf : Ff) →*(0,sf,rsf) TM(Yf,Ftf,Itf) ⇔
    ⊢ T(monitor Xf : Ff) →*(0,sf,rsf,hf) TM(Yf,Ftf,Itf)
```

Direction (⇒). Assume

(4) ⊢ T(monitor Xf : Ff) →*(0,sf,rsf) TM(Yf,Ftf,Itf)

Prove

[5] ⊢ T(monitor Xf : Ff) →*(0,sf,rsf,hf) TM(Yf,Ftf,Itf)

From (4), by the def. of →*(0,sf,rsf), we get

(6) T(monitor Xf : Ff) = TM(Yf,Ftf,Itf).

and

(7) rsf = ∅.

From (6,7) and the def. of →*(0,sf,rsf,hf) we obtain [5].

Direction (⟸) can be proved analogously.

Hence, Q(0) holds.

========

Proof of 2

Take arbitrary n∈ℕ.

Assume Q(n), i.e.

```
(1)  ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
     Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
     let M = monitor X : F, Mt = TM(Y,Ft,It) :
     ⊢ M: (h,d) ⇒
          (h∈ℕ ⇒ (⊢ T(M) →*(n,s,rs) Mt ⇔ ⊢ T(M) →*(n,s,rs,h) Mt))
```

Prove Q(n+1), i.e.,

```
[2]  ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
     Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
     let M = monitor X : F, Mt = TM(Y,Ft,It) :
     ⊢ M: (h,d) ⇒
          (h∈ℕ ⇒ (⊢ T(M) →*(n+1,s,rs) Mt ⇔ ⊢ T(M) →*(n+1,s,rs,h) Mt))
```

We take Xf, Ff, hf, df, sf, rsf, Yf, Ftf, Itf arbitrary but fixed.

Assume

```
(3) ⊢ (monitor Xf : Ff): (hf, df)
(4) hf∈ℕ
```

and prove

```
[5] ⊢ T(monitor Xf : Ff) →*(n+1,sf,rsf) TM(Yf,Ftf,Itf) ⇔
    ⊢ T(monitor Xf : Ff) →*(n+1,sf,rsf,hf) TM(Yf,Ftf,Itf)
```

To prove (5), we need to prove

```
[5.1]
   ⊢ T(monitor Xf : Ff) →*(n+1,sf,rsf) TM(Yf,Ftf,Itf) ⇒
   ⊢ T(monitor Xf : Ff) →*(n+1,sf,rsf,hf) TM(Yf,Ftf,Itf).
```

and

```
[5.2]
   ⊢ T(monitor Xf : Ff) →*(n+1,sf,rsf,hf) TM(Yf,Ftf,Itf) ⇒
   ⊢ T(monitor Xf : Ff) →*(n+1,sf,rsf) TM(Yf,Ftf,Itf).
```

Proof of [5.1]

Since T(monitor Xf : Ff)=TM(Xf,T(Ff),∅), we assume

```
(6)  ⊢ TM(Xf,T(Ff),∅) →*(n+1,sf,rsf) TM(Yf,Ftf,Itf)
```

and prove

```
[7] ⊢ TM(Xf,T(Ff),∅) →*(n+1,sf,rsf,hf) TM(Yf,Ftf,Itf).
```

From (3) and (6), by the invariant statement, we know

```
(8) Yf=Xf, Ftf=T(Ff)
```

From (6) by the definition of →* we know that there exist Y', Ft', It', rs1' and rs2' such that

```
(9)  rsf=rs1'∪rs2'
(10)  ⊢ TM(Xf,T(Ff),∅) →*(n,sf,rs1') TM(Y',Ft',It')
(11)  ⊢ TM(Y',Ft',It') →(n,sf↓(n),sf(n),rs2') TM(Xf,T(Ff),Itf)
```

From (10), by the definition of →, (and by the invariant) we have

```
(12) Y'=Xf, Ft'=T(Ff).
```

From (10), by (1,3,4), and (12) we get

```
(13) ⊢ TM(Xf,T(Ff),∅) →*(n,sf,rs1',hf) TM(Xf,T(Ff),Itf)
```

From (11) by (12) we have

```
(14) ⊢ TM(Xf,T(Ff),It') →(n,sf↓(n),sf(n),rs2') TM(Xf,T(Ff),Itf)
```

From (14), by definition of → for TMonitors we know

```
(15) rs2' = { t ∈ ℕ | ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
           ⊢ g →(n,sf↓(n),sf(n),c) done(false) }
(16) Itf = { (t,g1,c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈It0 ∧
           ⊢ g →(n,sf↓(n),sf(n),c) next(g1) }
```

where

```
(17) It0 = It' ∪ {(n,T(Ff),({(X,n)},{X,sf(n)}))}
```

To prove (7), by the definition of →* with h-cutoff for TMonitors, and (12), we need to prove that there exist Y*,Ft*, It*, rs1* and rs2* such that

```
(18) rs1*∪rs2*=rsf
(19) ⊢ TM(Xf,T(Ff),∅) →*(n,sf,rs1*,hf) TM(Y*,Ft*,It*)
(20) TM(Y*,Ft*,It*) →(n,s↑(max(0,n-hf),min(n,hf)),s(n),rs2*) TM(Xf,T(Ff),Itf).
```

We can take rs1*=rs1', rs2*=rs2', Y*=Xf, Ft*=Ftf=T(Ff), It*=It'. Then (18) holds due to (9) and (19) holds due to (13). Hence, we need to prove only (20), which after instantiating the variables has the form

```
(21) TM(Xf,T(Ff),It') →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),rs2')
     TM(Xf,T(Ff),Itf).
```

By definition of → for TMonitors, to prove (21), we need to prove

```
[22] rs2' = { t ∈ ℕ |
               ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
               ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) done(false) }
```

and

```
[23] Itf = { (t,g1,c) ∈ TInstance |
               ∃g∈TFormula: (t,g,c)∈It0 ∧
               ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) next(g1) }
```

where Itf0 is defined as in (17).

Hence, by (15) and [22], we need to prove

```
[24] { t ∈ ℕ | ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
               ⊢ g →(n,sf↓(n),sf(n),c) done(false) }
     =
     { t ∈ ℕ |
               ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
               ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) done(false) }
```

By (16) and [23], we need to prove

```
[25] { (t,g1,c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈It0 ∧
          ⊢ g →(n,sf↓(n),sf(n),c) next(g1) }
     =
     { (t,g1,c) ∈ TInstance |
               ∃g∈TFormula: (t,g,c)∈It0 ∧
               ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) next(g1) }
```

To prove [24], we need to show

```
[26] ∀t ∈ ℕ :
       ∃g∈TFormula,c∈Context:
       (t,g,c)∈It0 ∧  ⊢ g →(n,sf↓(n),sf(n),c) done(false)
       ⇔
       ∃g∈TFormula,c∈Context:
       (t,g,c)∈It0 ∧  ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) done(false).
```

To prove (25), we need to show

```
[27] ∀t ∈ ℕ, g1∈TFormula, c∈Context
       ∃g∈TFormula:
       (t,g,c)∈It0 ∧  ⊢ g →(n,sf↓(n),sf(n),c) next(g1)
       ⇔
       ∃g∈TFormula:
       (t,g,c)∈It0 ∧  ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) next(g1).
```

Proof of [26, ⟹].

We take t0 arbitrary but fixed. Let g∈TFormula and c∈Context be such that

(26.1) (t0,g,c)∈It0 and
(26.2) ⊢ g →(n,sf↓(n),sf(n),c) done(false)

hold. We need to find g*∈TFormula and c*∈Context such that

[26.3] (t0,g*,c*)∈It0 and
[26.4] ⊢ g* →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c*) done(false)

hold. We take g*=g and c*=c. Then (26.3) holds because of (26.1). Hence, we only need to prove

[26.4] ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) done(false)

Since (t0,g,c)∈It0, we have either

(26.5) (t0,g,c)∈It', or
(26.6) t0=n, g=T(Ff), c=({(Xf,n)},{Xf,sf(n)}).

Let first consider the case (26.5).

We had

(3) ⊢ (monitor Xf : Ff): (hf, df)

(10) ⊢ TM(Xf,T(Ff),∅) →*(n,sf,rs1’) TM(Y’,Ft’,It’)

From (3) and (10), by the invariant statement, we have

(26.7) invariant(Xf,Y’,Ff,Ft’,It’,n,sf,df)

The invariant (26.7) implies

(12) Y’=Xf, Ft’=T(Ff)

and by (26.5) the following:

(26.8) T(Ff) →* (n-t0,t0,sf,c.1) g.

From (26.8), by Lemma 2 we get

(26.9) T(Ff) →l* (n-t0,t0,sf,c.1) g.

From (26.5) and (26.7) we get

(26.10) c.1={(Xf,t0)}, c.2={(X,sf(t0))}={(X,sf(c.1(Xf)))}

Since by the invariant n-t0+1>0, from (26.9), (26.2), (26.10), by the definition of →l*, we get

(26.11) T(Ff) →l* (n-t0+1,t0,sf,c.1) done(false).

From (26.11), by Lemma 2, we get

(26.12) T(Ff) →* (n-t0+1,t0,sf,c.1) done(false).

From (3) by the definition of ⊢, there exists re0∈RangeEnv such

(26.13) re0 ⊢ Ff: (hf, df) and
(26.14) re0(Xf) = (0,0)

From (26.10) and (26.14) the following is satisfied

(26.15) ∀Y∈dom(c.1): re0(Y).1+t0 ≤ c.1(Y) ≤ re0(Y).2+t0.

Hence, from (26.13), (26.15), (26.12) and the Statement 2 of Lemma 1 (taking F=Ff, re=re0, e=c.1, Ft=g, n=n-t0, p=t0, s=sf, d=df, h=h’=hf) we get

(26.16) T(Ff) →* (n-t0+1,t0,sf,c.1,hf) done(false).

From (26.16), by Lemma 2 we get

(26.17) T(Ff) →l* (n-t0+1,t0,sf,c.1,hf) done(false).

Since by the invariant n-t0+1>0, from (26.17), by the definition of →l* with history, there exists Ft0∈TFormula such that

(26.18) T(Ff) →l* (n-t0,t0,sf,c.1,hf) Ft0, (26.19) Ft0 →(n,s↑(max(0,n-hf),min(n,hf)), s(n), c) done(false).

From (26.18), by Lemma 2, we get

(26.20) T(Ff) →* (n-t0,t0,sf,c.1,hf) Ft0.

From (26.20), by (26.13), (26.15), and Statement 2 of Lemma 1 we get

(26.21) T(Ff) →* (n-t0,t0,sf,c.1) Ft0.

From (26.21) and (26.8), since the rules for → are deterministic and
→* is defined based on →, we conclude

(26.22) Ft0=g.

From (26.22) and (26.19), we get [26.4]

Now we consider the case (26.6):

(26.6) t0=n, g=T(Ff), c=({(Xf,n)},{Xf,sf(n)}).

Under (26.6), the formula (26.2) now looks as

(26.23) ⊢ T(Ff) →(n,sf↓(n),sf(n),c) done(false)

We need to prove [26.4], which, by (26.6) has the form

[26.24] ⊢ T(Ff) →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),({(X,n)},{X,sf(n)})) done(false)

From (3) by the definition of ⊢, there exists re0∈RangeEnv such

(26.25) re0 ⊢ Ff: (hf, df) and
(26.26) re0 = {Xf, (0,0)}

From (26.25) and (26.26) the following is satisfied

(26.27) ∀Y∈dom(c.1): re0(Y).1+n ≤ c.1(Y) ≤ re0(Y).2+n.

From (26.26) and (26.6) we have

(26.28) dom(c.1) = dom(re0).

From (26.25), (26.27), (4), the definition of c in (26.6), (26.28), and Lemma 3
(instantiating F=Ff, Ft=done(false), p=n, s=sf, h=h'=hf, d=df, e=c.1, re=re0)
we get [26.24].

Proof of [26, ⟸].

The direction (⟸) can proved analogously to the direction (⟹). This is easy to
see, because the proof of (⟸) relies on Statement 2 of Lemma 1 and on Lemma 3.
Both of these propositions assert equivalence between a formula expressed in the
version of →* (resp. →) without history and a formula expressed in the
version of →* (resp. →) with history. Hence, for proving [26, ⟹] we can use

Statement 2 of Lemma 1 and Lemma 3 in the direction opposite to the one used in the proof of [26, ⟸].

Proof of [27]

Proof of [27] is analogous to the proof of [26]. This is easy to see, because [27] and [26] differ only with a TFormula in the right hand side of →*, and the proof of [26] does not depend on what stands in that side. Hence, we can replace done(false) in the proof of [26] with next(g1) and we obtain the proof of [27].

Proof of [5.2].

We assume

(28) ⊢ TM(Xf,T(Ff),∅) →*(n+1,sf,rsf,hf) TM(Yf,Ftf,Itf)

and want to prove

[29] ⊢ TM(Xf,T(Ff),∅) →*(n+1,sf,rsf) TM(Yf,Ftf,Itf).

From (28), by the definition of →* with cut-off for TMonitors, we know that there exist Yf', Ftf', Itf', rs1', rs2', such that

(30) rs1'∪rs2'=rsf
(31) ⊢ TM(Xf,T(Ff),∅) →*(n,sf,rs1',hf) TM(Yf',Ftf',Itf') and
(32) TM(Yf',Ftf',Itf') →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),rs2')
      TM(Yf,Ftf,Itf)

From the definitions of →* and → we can see that Yf'=Xf, Ftf'=T(Ff).

To prove [29], by the definition of →* for TMonitors, we need to find such Yf*, Ftf*, Itf*, rs1*, and rs2* that

[33] rs1*∪rs2*=rsf
[34] ⊢ TM(Xf,T(F),∅) →*(n,sf,rs1*) TM(Yf*,Ftf*,Itf*) and
[35] TM(Yf*,Ftf*,Itf*) →(n,sf↓n,sf(n),rs2*) TM(Xf,T(Ff),Itf)

We take Yf*=Xf, Ftf*=T(F), Itf*= Itf', rs1*=rs1', rs2*=rs2'. Then:

- [33] follows from (30).
- [34] follows from (31) by (3,4) and the induction hypothesis (1).

Hence, it is only left to prove the following instance of [35]:

[36)] TM(Xf,T(Ff),Itf') →(n,sf↓n,sf(n),rs2') TM(Xf,T(Ff),Itf)

To show it, by the definition of → for TMonitors,
we need to prove

```
[37] rs2’ = { t ∈ ℕ |
                ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
                ⊢ g →(n,sf↓n,sf(n),c) done(false) }
```

and

```
[38] Itf = { (t,g1,c) ∈ TInstance |
                ∃g∈TFormula: (t,g,c)∈It0 ∧
                ⊢ g →(n,sf↓n,sf(n),c) next(g1) }
```

where It0 = Itf’ ∪ {(n,T(Ff),({(X,n)},{X,sf(n)}))}

On the other hand, from (32) we know that

```
(39) rs2’ = { t ∈ ℕ |
                ∃g∈TFormula,c∈Context: (t,g,c)∈It0’ ∧
                ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) done(false) }
```

and

```
(40) Itf = { (t,g1,c) ∈ TInstance |
                ∃g∈TFormula: (t,g,c)∈It0’ ∧
                ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) next(g1) }
```

where It0’ is defined exactly as It0: It0’=It0.

Hence, by [37] and (39), we need to prove

```
[41]       { t ∈ ℕ |
                ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
                ⊢ g →(n,sf↑n,sf(n),c) done(false) }
           =
           { t ∈ ℕ |
                ∃g∈TFormula,c∈Context: (t,g,c)∈It0 ∧
                ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) done(false) }
```

But this is exactly [24] which we have already proved. Hence, [41] holds.

By (40) and [38], we need to prove

```
[42]       { (t,g1,c) ∈ TInstance |
                ∃g∈TFormula: (t,g,c)∈It0 ∧
                ⊢ g →(n,sf↓n,sf(n),c) next(g1) }
           =
           { (t,g1,c) ∈ TInstance |
                ∃g∈TFormula: (t,g,c)∈It0’ ∧
                ⊢ g →(n,sf↑(max(0,n-hf),min(n,hf)),sf(n),c) next(g1) }
```

But this is exactly [25] which we have already proved. Hence, [42] holds.

It means, we proved also [35]. It finished the proof of [5.2] and, hence, of the soundness theorem.

A.2 Proposition 1: The Invariant Statement

```
∀X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, n∈ℕ, s∈Stream, rs∈ℙ(ℕ),
   Y∈Variable Ft∈TFormula, It∈ℙ(TInstance):
  ⊢ (monitor X : F): (h,d) ∧
  ⊢ T(monitor X : F) →*(n,s,rs) TM(Y,Ft,It) ⇒
    invariant(X,Y,F,Ft,It,n,s,d)

PROOF
-----

Parameterization
----------------

P(n):⇔
  ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
     Y∈Variable Ft∈TFormula, It∈ℙ(Instance):
  ⊢ (monitor X : F): (h,d) ∧
  ⊢ T(monitor X : F) →*(n,s,rs) TM(Y,Ft,It) ⇒
    invariant(X,Y,F,Ft,It,n,s,d)

We want to show

∀n∈ℕ: P(n)

For this it suffices to show

1. P(0)
2. ∀n∈ℕ: P(n) ⇒ P(n+1)

Proof of 1
----------
P(0)

  ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
     Y∈Variable Ft∈TFormula, c∈Context, It∈ℙ(Instance):
  ⊢ (monitor X : F): (h,d) ∧
  ⊢ T(monitor X : F) →*(0,s,rs) TM(Y,Ft,It) ⇒
    invariant(X,Y,F,Ft,It,0,s,d)

We take Xf,Ff,df,hf,sf,rsf,Yf,Ftf,Itf arbitrary but fixed.

Assume

(1) ⊢ (monitor Xf : Ff): (hf,df)
// (2) df∈ℕ
(3) T(monitor Xf : Ff) →*(0,sf,rsf) TM(Yf,Ftf,Itf)

and show

[a] invariant(Xf,Yf,Ff,Ftf,Itf,0,sf,df)
```

```
From (3) and def. →*, we know

(4) rsf = ∅
(5) T(monitor Xf : Ff) = TM(Yf,Ftf,Itf)

From (5) and Def. of T(M), we know

(6) Yf = Xf
(7) Ftf = T(Ff)
(8) Itf = ∅

From (6,7,8) and the definitions of alldiff, allnext, and the invariant,
we get [a].
=============================

Proof of 2
----------
∀n∈ℕ: P(n) ⇒ P(n+1)

Take arbitrary n∈ℕ.

Assume P(n), i.e.,

(1) ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
       Y∈Variable, Ft∈TFormula, It∈ℙ(Instance):
    ⊢ (monitor X : F) : (h,d)  ∧
    ⊢ T(monitor X : F) →*(n,s,rs) TM(Y,Ft,It) ⇒
    invariant(X,Y,F,Ft,It,n,s,d)

Show P(n+1), i.e.,

(a) ∀ X∈Variable, F∈Formula, h∈ℕ∞, d∈ℕ∞, s∈Stream, rs∈ℙ(ℕ),
       Y∈Variable Ft∈TFormula, It∈ℙ(Instance) :
    ⊢ (monitor X : F) : (h,d) ∧
    ⊢ T(monitor X : F) →*(n+1,s,rs) TM(Y,Ft,It) ⇒
    invariant(X,Y,F,Ft,It,n+1,s,d)

We take Xf,Ff,df,hf,sf,rsf,Yf,Ftf,Itf arbitrary but fixed.

Assume

(2) ⊢ (monitor Xf : Ff) : (hf,df)
// (3) df∈ℕ
(4) T(monitor Xf : Ff) →*(n+1,sf,rsf) TM(Yf,Ftf,Itf)

and show

[b] invariant(Xf,Yf,Ff,Ftf,Itf,n+1,sf,df)

From (4) and def. →* for TMonitors, we know for some rs1,rs2 and
Mt=TM(X',Ft',It')

(5) ⊢ T(monitor Xf : Ff) →*(n,sf,rs1) TM(X',Ft',It')
(6) ⊢ TM(X',Ft',It') →(n,sf↓n,sf(n),rs2) TM(Yf,Ftf,Itf)
```

```
(7) rsf = rs1∪rs2

From (6) by the definition of → for TMonitors, we know

(8) X'=Yf,
(9) Ft'=Ftf, and
(10) Itf = {(t0,next(Fc1),c0) ∈ TInstance |
            ∃ Ft0∈Tformula such that (t0,Ft0,c0)∈It0 and
               ⊢ Ft0 → (n,sf↓n,sf(n),c0) next(Fc1)}

where

(11) It0 = It' ∪ {(n,Ftf,({(Yf,n)},{(Yf,sf(n))}))}

From (1), for X=Xf, F=Ff,  h=hf, d=df, s=sf, rs=rs1, Y=Yf, Ft=Ftf,
and It= It', we obtain

(12)    ⊢ (monitor Xf : Ff) : (hf,df) ∧
        ⊢ T(monitor Xf : Ff) →*(n,sf,rs1) TM(Yf,Ftf,It') ⇒
        invariant(Xf,Yf,Ff,Ftf,It',n,sf,df)

From (14,2,3,5,8,9) we obtain

(13)   invariant(Xf,Yf,Ff,Ftf,It',n,sf,df)

It means, we know

(14) Xf = Yf
(15) Ftf = T(Ff)
(16) alldiffs(It')
(17) allnext(It')
(18) ∀ t∈ℕ, Ft∈TFormula, c∈Context:
      (t,Ft,c) ∈ It' ∧ d∈ℕ ⇒
         c.1={(Xf,t)} ∧ c.2={(Xf,sf(t))} ∧
         n-df ≤ t ≤ n-1 ∧
         T(Ff) →* (n-t,t,sf,c.1) Ft ∧
         ∃b∈Bool ∃d'∈ℕ :
         d'≤df ∧ ⊢ Ft →*(max(0,t+d'-n),n,sf,c.1) done(b)

Showing [b] means that we want to show

[b1] Xf = Yf
[b2] Ftf = T(Ff)
[b3] alldiff(Itf)
[b4] allsnext(Itf)
[b5] ∀ t∈ℕ, Ft∈TFormula, c∈Context:
     (t,Ft,c) ∈ Itf ∧ d∈ℕ ⇒
       c.1={(Xf,t)} ∧ c.2={(Xf,sf(t))} ∧
        n+1-df ≤ t ≤ n ∧
        T(Ff) →* (n+1-t,t,sf,c.1) Ft ∧
        ∃b∈Bool ∃d'∈ℕ :
        d'≤df ∧ ⊢ Ft →*(max(0,t+d'-n-1),n+1,sf,c.1) done(b)
```

```
Proof of [b1]
-------------
[b1] is proved by (14).


Proof of (b2)
-------------
[b2] is proved by (15).


Proof of [b3]
-------------
From (10) one can see that the elements (t,Ft,c) in Itf inherit their tag t from
It0, which is It' ∪ {(n,Ftf,(cp,cm))}. From (18) we know alldiff(It'). From (18)
we have t ≤ n-1 for all (t,Ft1,c) ∈ It'. Adding {(n,Ftf,cf)} to It', will
guarantee all instances in It0 have different tags. Since these tags are
transfered to Itf, we conclude that [b3] holds.


Proof of [b4]
-------------
(b4) follows directly from (10), since every element in Itf has a form
(t,next(Fc),c).


Proof of [b5]
-------------
Recall that we have to prove

∀ t∈ℕ, Ft∈TFormula, c∈Context:
     (t,Ft,c) ∈ Itf ∧ d∈ℕ ⇒
        c.1={(Xf,t)} ∧ c.2={(Xf,sf(t))} ∧
        n+1-df ≤ t ≤ n ∧
        T(Ff) →* (n+1-t,t,sf,c.1) Ft ∧
        ∃b∈Bool ∃d'∈ℕ :
        d'≤df ∧ ⊢ Ft →*(max(0,t+d'-n-1),n+1,sf,c.1) done(b)

We take tb, Ftb, cb arbitrary but fixed, assume

(19) (tb,Ftb,cb) ∈ Itf ∧ d∈ℕ

and prove

[b5.1] cb.1={(Xf,tb)} ∧ cb.2={(Xf,sf(tb))}
[b5.2] n+1-df ≤ tb ≤ n
[b5.3] T(Ff) →* (n+1-tb,tb,sf,cb.1) Ftb ∧
[b5.4] ∃b∈Bool ∃d'∈ℕ :
          d'≤df ∧ ⊢ Ftb →*(max(0,tb+d'-n-1),n+1,sf,cb.1) done(b)

From (19) and (b4) we know that there exists Fcb∈TFormulaCore such that

(20) Ftb=next(Fcb)

From (19), (20) and (10) of we know there exists Ft0∈TFormula such that
```

(21) (tb,Ft0,cb) $\in$ It0 and
(22) $\vdash$ Ft0 $\rightarrow$ (n,sf$\downarrow$n,sf(n),cb) next(Fcb).

Proof of [b5.1]

We want to prove

[b5.1] cb.1={(Xf,tb)} $\wedge$ cb.2={(Xf,sf(tb))}

From (21) and (11), we have two cases:

(C1) (tb,Ft0,cb) = (n,Ftf,({(X',n)},{(X',sf(n))})) and
(C2) (tb,Ft0,cb) $\in$ It'.

In case (C1) we have tb=n, Ft0 = Ftf, and cb = ({(X',n)},{(X',sf(n))}).
From the latter, by (8) and (14), we have cb = ({(Xf,n)},{(Xf,sf(n))}) and,
hence, since tb=n, we get cb.1={(Xf,tb)} and cb.2={(Xf,sf(tb))}, which proves
(b5.1) for the case (C1).

In case (C2), [b5.1] follows from (18).

Hence, [b5.1] is proved.

Proof of [b5.2]

We want to prove

[b5.2] n+1-df $\leq$ tb $\leq$ n.

Again, from (21) and (11), we have two cases:

(C1) (tb,Ft0,cb) = (n,Ftf,({(X',n)},{(X',sf(n))})) and
(C2) (tb,Ft0,cb) $\in$ It'.

The case (C1)

In case (C1) we have tb=n, Ft0 = Ftf, and cb = ({(X',n)},{(X',sf(n))}).
From the latter, by (8) and (14), we have cb = ({(Xf,n)},{(Xf,sf(n))}).
To show [b5.2], it just remains to prove

[23] df > 0.

Assume by contradiction that df=0. Then from (2) we get that there exists
re0$\in$RangeEnv such that re0(Xf) = (0,0) and

(24) re0 $\vdash$ Ff:(hf,0)

Now we apply Statement 1 of Lemma 1 with F=Ff, re=re0, e={(Xf,n)}, s=sf, d=df=0,
h=hf, s=sf, p=n, and since T(Ff)=Ftf by (17), we obtain

(25) ∃b∈Bool ∃d'∈ℕ: d'≤1 ∧ ⊢ Ftf →*(d',n,sf,{(Xf,n)}) done(b))

From (25), there exist bl∈Bool and dl'∈ℕ such that

(26) dl'≤1 and
(27) Ftf →*(dl',n,sf,{(Xf,n)}) done(bl).

Note that since Ftf = T(Ff), by the definition of the translation T, Ftf is a 'next' formula. Hence, dl'≠0, because otherwise by (27) and the definition of →* we would get Fft=done(bl), which would contradict the fact that Ftf is a 'next' formula. Therefore, from (26) we get

(28) dl'=1.

From (27) and (28) we get

(29) Ftf →*(1,n,sf,{(Xf,n)}) done(bl).

From (29), by the definition of →* for TFormulas, we get that there exists Ft' such that

(30) Ftf →(n,sf↓n,sf(n),({(Xf,n)},{(Xf,sf(n))})) Ft'
(31) Ft' →*(0,n+1,sf,{(Xf,n)}) done(bl).

On the other hand, from (22), by Ft0=Ftf and (b5.1) we get

(32) Ftf →(n,sf↓n,sf(n),({(Xf,n)},{(Xf,sf(n))})) next(Fcb)

From (30) and (32) and by the fact that the reduction → is deterministic (one can not perform two different reductions from Ftf with the same n,sf↓n,sf(n),and ({(Xf,n)},{(Xf,sf(n))}): This can be seen by inspecting the rules for →), we obtain

(33) Ft'=next(Fcb).

Then from (31) and (33) we get

(34) next(Fcb) →*(0,n+1,sf,({(Xf,n)},{(Xf,sf(n))) done(bl).

But this contradicts the definition of →*: A 'next' formula can not be reduced to a 'done' formula in 0 steps. Hence, the obtained contradiction proves [23] and, therefore, [b5.2] for the case (C1).

Now we consider the case (C2).

From (tb,Ft0,cb) ∈ It', by (18), we get

(35) n-df ≤ tb ≤ n-1.

In order to prove [b5.2], we need to show

[36] n+1-df ≤ tb.

Assume by contradiction that n+1-df > tb. By (35) it means n-df = tb.
From (18) with t=tb, Ft=Ft0, c=cb we get

(37) ∃b∈Bool ∃d'∈ℕ :
        d'≤df ∧ ⊢ Ft0 →*(max(0,tb+d'-n),sf,cb.1) done(b)

Since tb+d'-n = n-df+d'-n = d'-df, from (37), we obtain that there exist b and d' such that

(38) d'≤df ∧ ⊢ Ft0 →*(max(0,d'-df),sf,cb.1) done(b)

holds. But then max(0,d'-df)=0 and we get

(39) Ft0 →*(0,sf,cb.1) done(b)

which, by definition of →* for TFormulas, implies

(40) Ft0 = done(b).

However, this contradicts (22) and the definition of → for TFormulas, because no 'done' formula can be reduced. Hence, (36) holds, which implies [b5.2] also in this case.

Proof of [b5.3]

We have to prove T(Ff) →* (n+1-tb,tb,sf,cb.1) Ftb, which, by Lemma 2, is equivalent to proving

(41) T(Ff) →l* (n+1-tb,tb,sf,cb.1) Ftb

Since n+1-tb>0 (by b5.2), by the definition of →l*, proving (41) reduces to proving that there exists such a Ft' that

[42] T(Ff) →l* (n-tb,tb,sf,cb.1) Ft' and
[43] Ft' →(n,sf↓(n),s(n),c') Ftb

where c'=(cb.1,{(X,sf(cb.1(X))) | X ∈ dom(cb.1)}). But since dom(cb.1)={Xf}, we actually get

(44) c'=cb.

Let us take Ft'=Ft0. Then (43) follows from (22). To prove (41), we reason as follows:

From (21), we know that (tb,Ft0,cb) ∈ It0. By (11) and (14), we have

(45) It0 = It' ∪ {(n,Ftf,({(Xf,n)},{(Xf,sf(n))}))}

Let us first consider the case when (tb,Ft0,cb) ∈ It'. From (18) we have

(46) T(Ff) →* (n-tb,tb,sf,cb.1) Ft0

From (46), by Lemma 2, we get (42).

Now assume (tb,Ft0,cb)∈{(n,Ftf,({(Xf,n)},{(Xf,sf(n))}))}. That means, taking tb=n, Ft0=Ftf, and cb=({(Xf,n)},{(Xf,sf(n))}). Then, from (42), we need to prove

[47] T(Ff) →l* (0,n,sf,{(Xf,n)}) Ftf.

This follows from the definition of →l* and [b2].

Hence, [b5.3] is proved.

Proof of [b5.4]

Recall that we took tb, Ftb, cb arbitrary but fixed and assumed

(21) (tb,Ftb,cb) ∈ Itf.

We are looking for b*∈Bool and d'*∈ℕ such that

[48] d'*≤df and
[49] ⊢ Ftb →*(max(0,tb+d'*-n-1),n+1,sf,cb.1) done(b*)

hold.

From (21) and (b4) we know that there exists Fcb∈TFormulaCore such that

(50) Ftb=next(Fcb)

From (21), by (11) there are two cases:

(C1) (tb,Ft0,cb) = (n,Ftf,({(X',n)},{(X',sf(n))}))
(C2) (tb,Ft0,cb) ∈ It'

Case (C1):

From (C1) we know

(51) tb = n
(52) Ft0 = Ftf
(53) cb = ({(Xf,n)},{(Xf,sf(n))})

From (51), to show [b5.3], it suffices to show

[b5.3.a] ∃b∈Bool, d'∈ℕ:
           d'≤df ∧ ⊢ Ftb →*(max(0,d'-1),n+1,sf,cb.1) done(b)

From (53), we know

(54) cb.1 = {(Xf,n)}
(55) cb.2 = {(Xf,sf(n))}

From (2) and the definition of ⊢ we have some re∈RangeEnv such that

(56) re(Xf) = (0,0)
(57) re ⊢ Ff: (hf,df)

From (Statement 1 of Lemma 1), (57), (19), (15), we have some b1∈Bool and d1'∈ℕ such that

(58) d1'≤df+1
(59) ⊢ Ftf →*(d1',n,sf,{(Xf,n)}) done(b1)

From (20,59) and the definition of →*, we know for some Ftb'∈TInstance

(60) d1' > 0
(61) ⊢ Ftf →(n,sf↓n,sf(n),({(Xf,n)},{(Xf,sf(n))})) Ftb'
(62) ⊢ Ftb' →*(d1'-1,n+1,sf,{(Xf,n)}) done(b1)

From (22,52,53), we know

(63) ⊢ Ftf → (n,sf↓n,sf(n),({(Xf,n)},{(Xf,sf(n))})) Ftb

From (61,63) and the fact that the rules for → are deterministic
(i.e., ∀Ftf,Ftb,Ftb': (⊢ Ftf → Ftb) ∧ (⊢ Ftf → Ftb') ⇒ Ftb = Ftb',
a lemma easy to prove), we know

(64) Ftb' = Ftb

From (62,64), we know

(65) ⊢ Ftb →*(d1'-1,n+1,sf,{(Xf,n)}) done(b1)

From (60), we know

(66) d1'-1 = max(0,d1'-1)

From (58,65,66,54), we know [b5.3.a] with b:=b1 and d:=d1'-1.

Case (C2).

Recall that in this case (tb,Ft0,cb) ∈ It'.

By the induction hypothesis (18) there exist bi∈Bool and di'∈ℕ such that

(67) di'≤df and
(68) ⊢ Ft0 →*(max(0,tb+di'-n),n,sf,cb.1) done(bi)

This implies that

(69) tb+di'-n>0,

otherwise we would have Ft0=done(bi), which contradicts the assumption
(tb,Ft0,cb) ∈ It' and (20). Hence, we have

(70) ⊢ Ft0 →*(tb+di'-n,n,sf,cb.1) done(bi)

Therefore, we can apply the definition →* for TFormulas to (70) and (22),
concluding ⊢ next(Fcb) →*(tb+di'-n-1,n+1,sf,cb.1) done(bi) and, hence

(71) ⊢ Ftb →*(tb+di'-n-1,n+1,sf,cb.1) done(bi)

Now we can take d'*=d' and b*=bi. From (59) we get

(72) tb+di*'-n-1 = max(0,tb+di*'-n-1).

From (71) and (72) we get [49]. From (67) and the assumption d'*=d' we get [48]. Hence, [b5.3] is true also in case (b6.2 C2).

This finishes the invariant proof.

A.3 Lemma 1: Soundness Lemma for Formulas

```
∀F,F'∈Formula, re∈RangeEnv, e∈Environment, Ft∈TFormula, n∈ℕ, p∈ℕ,
 s∈Stream, d∈ℕ∞, h∈ℕ:
  ⊢ (re ⊢ F: (h,d)) ∧ dom(e) = dom(re) ∧
    ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p
    ⇒
      ( d∈ℕ ⇒
          ∃b∈Bool, ∃d'∈ℕ:
          d'≤d+1 ∧ ⊢ T(F) →*(d',p,s,e) done(b)) ∧
      ( ∀h'∈ℕ: h'≥h ⇒
          ( T(F) →* (n,p,s,e)   Ft ⇔
            T(F) →* (n,p,s,e,h') Ft   ) )

=====================================================

We split the lemma in two parts:

Statement 1.
 ∀F∈Formula, re∈RangeEnv, e∈Environment, s∈Stream, d∈ℕ∞, h∈ℕ, p∈ℕ:
   (⊢ (re ⊢ F: (h,d)) ∧ dom(e) = dom(re) ∧
   ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧
   d∈ℕ )  ⇒
         ∃b∈Bool ∃d'∈ℕ:
         d'≤d+1 ∧ ⊢ T(F) →*(d',p,s,e) done(b))

Statement 2.
 ∀F∈Formula, re∈RangeEnv, e∈Environment, Ft∈TFormula, n∈ℕ, p∈ℕ,
  s∈Stream, d∈ℕ∞, h∈ℕ, h'∈ℕ:
  (⊢ (re ⊢ F: (h,d)) ∧ dom(e) = dom(re) ∧
  ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧
  h'≥h ) ⇒
      ( T(F) →* (n,p,s,e)   Ft ⇔
        T(F) →* (n,p,s,e,h') Ft   )


===================================================

Statement 1.
 ∀F∈Formula, re∈RangeEnv, e∈Environment, s∈Stream, d∈ℕ∞, h∈ℕ:
   (⊢ (re ⊢ F: (h,d)) ∧ dom(e) = dom(re) ∧
   ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧
   d∈ℕ ) ⇒
         ∀p∈ℕ ∃b∈Bool ∃d'∈ℕ:
         d'≤d+1 ∧ ⊢ T(F) →*(d',p,s,e) done(b))


Parametrization
---------------

R(F) :⇔

 ∀re∈RangeEnv, e∈Environment, s∈Stream, d∈ℕ∞, h∈ℕ:
```

```
   (⊢ (re ⊢ F: (h,d)) ∧ dom(e) = dom(re) ∧
   ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧
   d∈ℕ) ⇒
     (  ∀p∈ℕ ∃b∈Bool ∃d'∈ℕ:
        d'≤d+1 ∧ ⊢ T(F) →*(d',p,s,e) done(b))
```

We want to prove

∀F∈Formula: R(F)

By structural induction over F:

C1: F=@X. Then T(F) = next(TV(X)).

We take ref, ef, sf, df, hf, pf arbitrary but fixed. Assume

(1.1) ⊢ (ref ⊢ @X: (hf,df))
(1.2) df∈ℕ,
(1.3) dom(ef) = dom(ref) ∧ ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

and look for b*∈Bool and d*'∈ℕ such that

[1.4] d*'≤df+1 and
[1.5] ⊢ next(TV(X)) →*(d*',pf,sf,ef) done(b*)

hold.

From (1.1) we get

(1.6) hf=0 and
(1.7) df=0.

We define

(1.8) c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}),

and take

(1.9) d*'=1

and

```
(1.10) b* =
        if X∈dom(c.2)) then
          c.2(X)
        else
          false
```

From (1.7,1.9), we see that d*' satisfies [1.4]. Hence, we only need to prove
the following formula obtained from [1.5]:

[1.11] ⊢ next(TV(X)) →*(1,pf,sf,ef) done(b*).

where b* is defined in (1.10). By the definition of →*, to prove [1.11], we need to find Ft'∈TFormula such that

[1.12] next(TV(X)) →(pf,sf↓pf,sf(pf),c) Ft' and
[1.13] Ft' →*(0,pf+1,sf,ef) done(b*)

hold, where c is defined as in (1.8).

We take Ft'=done(b*). Then [1.12] holds by (1.10) and the definition of → for next(TV(X)), and [1.13] holds by the definition of →*.

C2. F = ~F1. Then T(F) = next(TN(T(F1))).

We take ref, ef, sf, df, hf, pf arbitrary but fixed. Assume

(2.1) ⊢ (ref ⊢ ¬F1: (hf,df))
(2.2) df∈ℕ,
(2.3) dom(ef) = dom(ref) ∧ ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

and look for such b*∈Bool and d*'∈ℕ such that

[2.4] d*'≤df+1 and
[2.5] ⊢ next(TN(T(F1))) →*(d*',pf,sf,ef) done(b*)

hold.

From (2.1) by the definition of the ⊢ relation we get

(2.6) ⊢ (re ⊢ F1): (hf,df).

From (2.6), (2.3) and the induction hypothesis there exist bi∈Bool and di'∈ℕ such that

(2.7) di'≤df+1 and
(2.8) ⊢ T(F1) →*(di',pf,sf,ef) done(bi).

We take

(2.9) d*'=di'

and define

```
(2.10)  b* :=
         if bi = true then
            false
         else
            true
```

By (2.7,2.9), the inequality [2.4] holds. From (2.8), (2.9), (2.10), by the Statement 1 of the Lemma 4 we get [2.5].

C3. F = F1&F2. Then T(F) = next(TCS(T(F1),T(F2))).

We take ref, ef, sf, df, hf, pf arbitrary but fixed. Assume

(3.1) ⊢ (ref ⊢ F1&F2: (hf,df)),
(3.2) df∈ℕ,
(3.3) dom(ef) = dom(ref) ∧ ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

and look for such b*∈Bool and d*'∈ℕ such that

[3.4] d*'≤df+1 and
[3.5] ⊢ next(TCS(T(F1),T(F2))) →*(d*',pf,sf,ef) done(b*)

From (3.1), by the definition of the ⊢ relation we get

(3.6) ⊢ (ref ⊢ F1: (h1,d1)
(3.7) ⊢ (ref ⊢ F2: (h2,d2)

such that h1,d1,h2,d2∈ℕ and

(3.8) df = max∞(d1,d2) = max(d1,d2)

From (3.6), (3.3), and the induction hypothesis there exist b1i∈Bool and d1i'∈ℕ such that

(3.9) d1i'≤d1+1 and
(3.10) ⊢ T(F1) →*(d1i',pf,sf,ef) done(b1i).

From (3.7) and the induction hypothesis there exist b2i∈Bool and d2i'∈ℕ such

(3.11) d2i'≤d2+1 and
(3.12) ⊢ T(F2) →*(d2i',pf,sf,ef) done(b2i).

From (3.10) and (3.12) we have

(3.13) d1i'>0 and
(3.14) d2i'>0

(Otherwise we would have a 'next' formula reducing to a 'done' formula in 0 steps, which is impossible.)

We proceed by case distinction over b1i.

b1i = false

We take

(3.15) b*=b1i=false and
(3.16) d*'=d1i'.

From (3.8,3.9,3.16) we get [3.4]. From (3.10, 3.13, 3.15, 3.16) and Statement 2 of Lemma 4 we get [3.5].

b1i = true.

We take

(3.17) b*=b2i' and
(3.18) d*'=max(d1i',d2i').

From (3.18, 3.9, 3.11) we get

(3.19) d*'=max(d1i',d2i') $\leq$ max(d1+1, d2+1)=max(d1,d2)+1 = df+1

Hence, (3.19) gives [3.4].

From (3.10, 3.12, 3.13, 3.14, 3.18) and Statement 2 of Lemma 4 we get [3.5].

C4. F = F1/\F2. Then T(F) = next(TCP(T(F1),T(F2))).

We take ref, ef, sf, df, hf, pf arbitrary but fixed. Assume

(4.1) $\vdash$ (re $\vdash$ F1$\wedge$F2: (hf,df)),
(4.2) df$\in\mathbb{N}$,
(4.3) dom(ef) = dom(ref) $\wedge$ $\forall$Y$\in$dom(ef): ref(Y).1 +i pf $\leq$i ef(Y) $\leq$i ref(Y).2 +i pf

and look for such b*$\in$Bool and d*'$\in\mathbb{N}$ such that

[4.4] d*'$\leq$df+1 and
[4.5] $\vdash$ next(TCP(T(F1),T(F2))) $\rightarrow$*(d*',pf,sf,ef) done(b*)

From (4.1), by the definition of the $\vdash$ relation we get

(4.6) $\vdash$ (re $\vdash$ F1: (h1,d1)
(4.7) $\vdash$ (re $\vdash$ F2: (h2,d2)

such that h1,d1,h2,d2$\in\mathbb{N}$ and

(4.8) df = max∞(d1,d2) = max(d1,d2)

From (4.6), (4.3), and the induction hypothesis there exist b1i$\in$Bool and d1i'$\in\mathbb{N}$
such that

(4.9) d1i'$\leq$d1+1 and
(4.10) $\vdash$ T(F1) $\rightarrow$*(d1i',pf,sf,ef) done(b1i).

From (4.7), (4.3) and the induction hypothesis there exist b2i$\in$Bool and d2i'$\in\mathbb{N}$
such that

(4.11) d2i'$\leq$d2+1 and
(4.12) $\vdash$ T(F2) $\rightarrow$*(d2i',pf,sf,ef) done(b2i).

From (4.10) and (4.12) we have

(4.13) d1i'>0 and
(4.14) d2i'>0

(Otherwise we would have a 'next' formula reducing to a 'done' formula in 0 steps, which is impossible.)

We proceed by case distinction over b1i and b2i.

b1i = false, b2i = true

We take

(4.15) b* = false,
(4.16) d*'= d1i'.

From (4.8, 4.9, 4.16) we get d*'=d1i'$\leq$ d1+1 $\leq$ max(d1,d2)+1 =df+1 and, hence [4.4].
From (4.10, 4.12, 4.13, 4.14, 4.15, 4.16) and the case [TCP1] of the Statement 3 of Lemma 4 we get [4.5].

b1i = false, b2i = false

We take

(4.17) b* = false,
(4.18) d*'= min(d1i',d2i').

From (4.9,4.11,4.18) we get

(4.19) d*'=min(d1i',d2i') $\leq$ min(d1+1,d2+1) = min(d1,d2)+1 $\leq$ max(d1,d2)+1 = df+1.

Hence, (4.19) proves [4.4].
From (4.10, 4.12, 4.13, 4.14, 4.17, 4.18) and the case [TCP2] of the Statement 3 of Lemma 4 we get [4.5].

b1i = true, b2i = true

We take

(4.20) b*=b2i' and
(4.21) d*'=max(d1i',d2i').

From (4.20, 4.9, 4.11) we get

(4.22) d*'=max(d1i',d2i') $\leq$ max(d1+1, d2+1)=max(d1,d2)+1=df+1

Hence, (4.22) gives [4.4].

From (4.10, 4.12, 4.13, 4.14, 4.20, 4.22) and the case [TCP3] of the Statement 3 of Lemma 4 we get [4.5].

b1i = true, b2i = false

We take

(4.23) b*=b2i' and
(4.24) d*'=d2i'.

From (4.18, 4.9, 4.11) we get

(4.25) d*'=d2i' $\leq$ d2+1 $\leq$ max(d1+1, d2+1)=max(d1,d2)+1=df+1

Hence, (4.25) gives [4.4].

From (4.10, 4.12, 4.13, 4.14, 4.23, 4.24) and the case [TCP4] of the Statement 3 of Lemma 4 we get [4.5].

C5. F = forall X in B1..B2:F1. Then T(F) = next(TA(X,T(B1),T(B2),T(F1)))

This case follows from the induction hypothesis for F1 and Lemma 5.

It finishes the proof of Statement 1 of Lemma 1.

==

Statement 2.
```
 ∀F∈Formula, re∈RangeEnv, e∈Environment, Ft∈TFormula, n∈ℕ, p∈ℕ,
  s∈Stream, d∈ℕ∞, h∈ℕ, h'∈ℕ:
  ⊢ (re ⊢ F: (h,d)) ∧ ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧ h'≥h ⇒
      ( T(F) →* (n,p,s,e)   Ft ⇔
        T(F) →* (n,p,s,e,h') Ft   )
```

Proof

Parametrization:

```
S(n) :⇔
 ∀F∈Formula, re∈RangeEnv, e∈Environment, Ft∈TFormula, p∈ℕ,
  s∈Stream, d∈ℕ∞, h∈ℕ, h'∈ℕ:
  ⊢ (re ⊢ F: (h,d)) ∧ ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧ h'≥h ⇒
      ( T(F) →* (n,p,s,e)   Ft ⇔
        T(F) →* (n,p,s,e,h') Ft   )
```

We need to prove

(a) S(0)
(b) $\forall$n$\in\mathbb{N}$: S(n) $\Rightarrow$ S(n+1)

Proof of (a)

We take Ff$\in$Formula, ref$\in$RangeEnv, ef$\in$Environment, Ftf$\in$TFormulas, pf$\in\mathbb{N}$, sf∈Stream, df∈ℕ∞, hf∈ℕ, hf'∈ℕ arbitrary but fixed, assume

```
(a.1) ⊢ (ref ⊢ Ff: (hf,df))
(a.2) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf
(a.3) hf'≥hf
```

and prove

```
(a.4)   T(Ff) →* (0,pf,sf,ef)     Ftf  ⇔
        T(Ff) →* (0,pf,sf,ef,hf') Ftf
```

(⟹)
Assume

```
(a.5) T(Ff) →* (0,pf,sf,ef) Ftf
```

and prove

```
(a.6) T(Ff) →* (0,pf,sf,ef,hf') Ftf.
```

From (a.5), by the definition of →* without history, we have Ftf=T(Ff).
Then (a.6) follows from the definition of →* with history.

(⟸). Analogous.

Proof of (b)

We assume

```
(b.1)
∀F∈Formula, re∈RangeEnv, e∈Environment, Ft∈TFormula, p∈ℕ, s∈Stream,
 d∈ℕ, h∈ℕ, h'∈ℕ:
   ⊢ (re ⊢ F: (h,d)) ∧ ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧ h'≥h ⇒
      ( T(F) →* (n,p,s,e)    Ft ⇔
        T(F) →* (n,p,s,e,h') Ft   )
```

and prove

```
[b.2]
∀F∈Formula, re∈RangeEnv, e∈Environment, Ft∈TFormula, p∈ℕ, s∈Stream,
 d∈ℕ, h∈ℕ, h'∈ℕ:
   ⊢ (re ⊢ F: (h,d)) ∧ ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ∧ h'≥h ⇒
      ( T(F) →* (n+1,p,s,e)    Ft ⇔
        T(F) →* (n+1,p,s,e,h') Ft   )
```

We take Ff, ref, ef, Ftf, pf, sf, df, hf, hf' arbitrary but fixed. Assume

```
(b.3) ⊢ (ref ⊢ Ff: (hf,df))
(b.4) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf
(b.5) hf'≥hf
```

and prove

(b.6)   T(Ff) →* (n+1,pf,sf,ef)   Ftf ⇔
        T(Ff) →* (n+1,pf,sf,ef,hf') Ftf

(⟹) Assume

(b.7) T(Ff) →* (n+1,pf,sf,ef) Ftf

and prove

[b.8] T(Ff) →* (n+1,pf,sf,ef,hf') Ftf

From (b.7), by the definition of →* without history, we know for some Ft'∈TFormula

(b.9) T(Ff) → (pf,sf↓pf, sf(pf),c) Ft'
(b.10) Ft' →* (n, pf+1, sf, ef) Ftf,

(b.11) c:= (ef, {(X, sf(ef(X))) | X in dom(ef)}).

Then from (b.3), (b.4), (b.11), (b.5), (b.9) and Lemma 3 we get

(b.12) T(Ff) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) Ft'.

Assume Ft' is a 'next' formula, i.e., there exists F'∈Formula such that

(b.13) Ft'=T(F').

From (b.3), (b.4), (b.5), (b.10), by the induction hypothesis (b.1) we get

(b.14) Ft' →* (n, pf+1, sf, ef, hf') Ftf.

If Ft' is a 'done' formula, then from (b.10) by the definition of →* without history we get n=0. Then, (b.14) again holds by the definition of →* with history.

From (b.11), (b.12) and (b.14), by the definition of →* with history we get [b.8].

(⟸) Assume

(b.15) T(Ff) →* (n+1,pf,sf,ef,hf') Ftf

and prove

[b.16] T(Ff) →* (n+1,pf,sf,ef) Ftf

From (b.15), by the definition of →* without history, we know for some Ft'∈TFormula

(b.17) T(Ff) → (pf,sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf),c) Ft'
(b.18) Ft' →* (n, pf+1, sf, ef, hf') Ftf,

where

(b.19) c:= (ef, {(X, sf(ef(X))) | X in dom(ef)}).

Then from (b.3), (b.19), (b.4), (b.5), (b.17) and Lemma 3 we get

(b.20) T(Ff) $\to$ (pf, sf$\downarrow$pf, sf(pf), c) Ft'.

Assume Ft' is a 'next' formula, i.e., there exists F'$\in$Formula such that

(b.21) Ft'=T(F').

From (b.3), (b.4), (b.5), (b.18) by the induction hypothesis (b.1) we get

(b.22) Ft' $\to$* (n, pf+1, sf, ef) Ftf.

If Ft' is a 'done' formula, then from (b.18) by the definition of $\to$* without history we get n=0. Then, (b.22) again holds by the definition of $\to$* with history.

From (b.19), (b.20) and (b.22), by the definition of $\to$* with history we get [b.16].

It finishes the proof of Statement 2 of Lemma 1.

A.4 Lemma 2: Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions

```
Lemma 2 (Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions):

(a) ∀n,p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula
          Ft1 →*  (n,p,s,e) Ft2  ⇔
          Ft1 →l* (n,p,s,e) Ft2

(b) ∀n,p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, h∈ℕ
          Ft1 →*  (n,p,s,e,h) Ft2  ⇔
          Ft1 →l* (n,p,s,e,h) Ft2

Proof of (a)
------------

Parametrization:
---------------

S(n,Ft1,Ft2,p,s,e) :⇔
  Ft1 →* (n,p,s,e) Ft2 ⇔ Ft1 →l* (n,p,s,e) Ft2

We want to prove

[G] ∀Ft1,Ft2∈TFormula,p∈ℕ, s∈Stream, e∈Environment,∀n∈ℕ:
    S(n,Ft1,Ft2,p,s,e).

We take Ftf1,Ftf2,pf,sf, and ef arbitrary but fixed.

We have to prove

[G1] ∀k,n∈ℕ: S(k,Ftf1,Ftf2,pf,sf,ef)∧n>k ⇒ S(n,Ftf1,Ftf2,pf,sf,ef).

Proof of [G1]
-------------

We take n arbitrary but fixed, assume

(1) ∀k<n: Ftf1 →* (k,pf,sf,ef) Ftf2 ⇔ Ftf1 →l* (k,pf,sf,ef) Ftf2

and prove

[2] Ftf1 →* (n,pf,sf,ef) Ftf2 ⇔ Ftf1 →l* (n,pf,sf,ef) Ftf2.

(⟹):
-----
We assume

(3) Ftf1 →*(n,pf,sf,ef) Ftf2

and prove

[4] Ftf1 →l*(n,pf,sf,ef) Ftf2.
```

From (3) we know that there exists Ft'∈TFormula such that

(5) Ftf1 →(pf,sf↓pf,sf(pf),c) Ft' and
(6) Ft' →*(n-1,pf+1,sf,ef) Ftf2

hold, where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (6), by the induction hypothesis we get

(7) Ft' →l*(n-1,pf+1,sf,ef) Ftf2.

From (7), by the definition of →l*, there are two alternatives:

(i) n-1 = 0
(ii) n-1 > 0.

In case (i), we get
----
(8) Ft'= Ftf2.

From (8) and (5) we get

(9) Ftf1 →(pf,sf↓pf,sf(pf),c) Ftf2.

On the other hand, by the definition of →l* we have

(10) Ftf1 →l*(0,pf,sf,ef) Ftf1.

From (10) and (9), by the definition of →l*, we get

(11) Ftf1 →l*(1,pf,sf,ef) Ftf2.

Since n-1=0, we get that [4] holds:

[4] Ftf1 →l* (n,pf,sf,ef) Ftf2.

Case (ii)

From (7), by the definition of →l*, there exists Ft''∈TFormula such that

(12) Ft' →l*(n-2,pf+1,sf,ef) Ft''
(13) Ft'' →(pf+n-1,sf↓(pf+n-1),sf(pf+n-1),c) Ftf2,

where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (12), by the induction hypothesis, we get

(14) Ft' →*(n-2,pf+1,sf,ef) Ft''.

From (5) and (14), by the definition of →* we get

(15) Ftf1 →*(n-1,pf,sf,ef) Ft''.

From (15), by the induction hypothesis, we get

(16) Ftf1 →l*(n-1,pf,sf,ef) Ft''.

From (16) and (13), by the definition of →l*, we get

[4] Ftf1 →l*(n,pf,sf,ef) Ftf2.

(⟸)

We assume

(17) Ftf1 →l* (n,pf,sf,ef) Ftf2

and prove

[18] Ftf1 →* (n,pf,sf,ef) Ftf2.

From (17), by the definition of →l*, we know that there exists
Ft'∈TFormula such that

(19) Ftf1 →l*(n-1,pf,sf,ef) Ft' and
(20) Ft' →(pf+n-1,sf↓(pf+n-1),sf(pf+n-1),c) Ftf2,

hold, where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (19), by the induction hypothesis we get

(21) Ftf1 →*(n-1,pf,sf,ef) Ft'

from (20), by the definition of →l*, there are two alternatives:

(i) n-1 = 0
(ii) n-1 > 0.

Case (i)

In this case, from (21) we get Ft'=Ftf1, which together with (20) and the fact
n-1=0 implies

(22) Ftf1 →(pf,sf↓pf,sf(pf),c) Ftf2.

On the other hand, by the definition of →* we have

(23) Ftf2 →*(0,pf+1,sf,ef) Ftf2.

From (22) and (23), by the definition of →*, w get

(24) Ftf2 →*(1,pf,sf,ef) Ftf2.

Since n-1=0, from (24) we get [18].

Case (ii)

```
From (21), by the definition of →*, there exists Ft''∈TFormula such that

(25) Ftf1 →(pf,sf↓pf,sf(pf),c) Ft''
(26) Ft'' →*(n-2,pf+1,sf,ef) Ft',

where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (26), by the induction hypothesis, we get

(27) Ft'' →l*(n-2,pf+1,sf,ef) Ft'.

From (27) and (20), by the definition of →l* we get

(28) Ft'' →l*(n-1,pf+1,sf,ef) Ftf2.

From (28), by the induction hypothesis we get

(29) Ft'' →*(n-1,pf+1,sf,ef) Ftf2.

From (25) and (29), by the definition of →*, we get

[18] Ftf1 →*(n,pf,sf,ef) Ftf2.
================================================================================

Proof of (b)
------------

Parametrization:
---------------

Q(n,Ft1,Ft2,p,s,e,h) :⇔
  Ft1 →* (n,p,s,e,h) Ft2 ⇔ Ft1 →l* (n,p,s,e,h) Ft2

We want to prove

(G) ∀Ft1,Ft2∈TFormula,p∈ℕ, s∈Stream, e∈Environment,h∈ℕ, ∀n∈ℕ :
    S(n,Ft1,Ft2,p,s,e,h).

We take Ftf1,Ftf2,pf,sf,ef, and hf arbitrary but fixed.

We have to prove

(G1) ∀k,n∈ℕ: S(k,Ftf1,Ftf2,pf,sf,ef,hf)∧n>k ⇒ S(n,Ftf1,Ftf2,pf,sf,ef,hf).

Proof of (G1)
-------------

We take n arbitrary but fixed, assume n>k and

(1) ∀k<n: Ftf1 →* (k,pf,sf,ef,hf) Ftf2 ⇔ Ftf1 →l* (k,pf,sf,ef,hf) Ftf2

and prove

(2) Ftf1 →* (n,pf,sf,ef,hf) Ftf2 ⇔ Ftf1 →l* (n,pf,sf,ef,hf) Ftf2.
```

(⟹):

We assume

(3) Ftf1 →*(n,pf,sf,ef,hf) Ftf2

and prove

(4) Ftf1 →l*(n,pf,sf,ef,hf) Ftf2.

From (3) we know that there exists Ft'∈TFormula such that

(5) Ftf1 →(pf,s↑(max(0,pf-hf),min(pf,hf)),sf(pf),c) Ft' and
(6) Ft' →*(n-1,pf+1,sf,ef,hf) Ftf2

hold, where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (6), by the induction hypothesis we get

(7) Ft' →l*(n-1,pf+1,sf,ef,hf) Ftf2.

From (7), by the definition of →l*, there are two alternatives:

(i) n-1 = 0
(ii) n-1 > 0.

In case (i), we get
----
(8) Ft'= Ftf2.

From (8) and (5) we get

(9) Ftf1 →(pf,s↑(max(0,pf-hf),min(pf,hf)),sf(pf),c) Ftf2.

On the other hand, by the definition of →l* we have

(10) Ftf1 →l*(0,pf,sf,ef,hf) Ftf1.

From (10) and (9), by the definition of →l*, we get

(11) Ftf1 →l*(1,pf,sf,ef,hf) Ftf2.

Since n-1=0, we get that [4] holds:

[4] Ftf1 →l* (n,pf,sf,ef,hf) Ftf2.

Case (ii)

From (7), by the definition of →l* with history, there exists Ft''∈TFormula such that

(12) Ft' →l*(n-2,pf+1,sf,ef,hf) Ft''

(13) Ft'' →(pf+n-2,sf↑(max(0,pf+n-2-hf),min(pf+n-2,hf)),sf(pf+n-2),c) Ftf2,

where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (12), by the induction hypothesis, we get

(14) Ft' →*(n-2,pf+1,sf,ef,hf) Ft''.

From (5) and (14), by the definition of →* with history we get

(15) Ftf1 →*(n-1,pf,sf,ef,hf) Ft''.

From (15), by the induction hypothesis, we get

(16) Ftf1 →l*(n-1,pf,sf,ef,hf) Ft''.

From (16) and (13), by the definition of →* with history, we get

[4] Ftf1 →l*(n,pf,sf,ef,hfx) Ftf2.

(⟸)

We assume

(17) Ftf1 →l* (n,pf,sf,ef,hf) Ftf2

and prove

[18] Ftf1 →* (n,pf,sf,ef,hf) Ftf2.

From (17), by the definition of →l* with history, we know that there exists
Ft'∈TFormula such that

(19) Ftf1 →l*(n-1,pf,sf,ef) Ft' and
(20) Ft' →(pf+n-1,s↑(max(0,pf+n-1-hf),min(pf+n-1,hf)),sf(pf+n-1),c) Ftf2,

hold, where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (19), by the induction hypothesis we get

(21) Ftf1 →*(n-1,pf,sf,ef,hf) Ft'

from (20), by the definition of →l* with history, there are two alternatives:

(i) n-1 = 0
(ii) n-1 > 0.

Case (i)

In this case, from (21) we get Ft'=Ftf1, which together with (20) and the fact
n-1=0 implies

(22) Ftf1 →(pf,s↑(max(0,pf-hf),min(pf,hf)),sf(pf),c) Ftf2.

On the other hand, by the definition of →* with history we have

(23) Ftf2 →*(0,pf+1,sf,ef,hf) Ftf2.

From (22) and (23), by the definition of →* with history, w get

(24) Ftf2 →*(1,pf,sf,ef,hf) Ftf2.

Since n-1=0, from (24) we get [18].

Case (ii)

From (21), by the definition of →* with history, there exists Ft''∈TFormula such that

(25) Ftf1 →(pf,s↑(max(0,pf-hf),min(pf,hf)),sf(pf),c) Ft''
(26) Ft'' →*(n-2,pf+1,sf,ef,hf) Ft',

where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

From (26), by the induction hypothesis, we get

(27) Ft'' →l*(n-2,pf+1,sf,ef,hf) Ft'.

From (27) and (20), by the definition of →l* with history we get

(28) Ft'' →l*(n-1,pf+1,sf,ef,hf) Ftf2.

From (28), by the induction hypothesis we get

(29) Ft'' →*(n-1,pf+1,sf,ef,hf) Ftf2.

From (25) and (29), by the definition of →*, we get

[18] Ftf1 →*(n,pf,sf,ef,hf) Ftf2.

A.5 Lemma 3: History Cut-Off Lemma

```
∀F∈Formula, Ft∈TFormula, p∈ℕ, s∈Stream, h∈ℕ, d∈ℕ∞, e∈Environment, re∈RangeEnv:
  ⊢ (re ⊢ F : (h,d)) ∧ dom(e) = dom(re) ∧
    ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ⇒
      let c:=(e, {(X, s(e(X))) | X ∈ dom(e)})
      ∀h'∈ℕ : h'≥h ⇒
        T(F) → (p, s↓p, s(p), c) Ft
        ⇔
        T(F) → (p, s↑(max(0,p-h'),min(p,h')), s(p), c) Ft
```

Proof

Parametrization:

```
S(F) :⇔
  ∀Ft∈Tformula, p∈ℕ, s∈Stream, h∈ℕ, d∈ℕ∞, e∈Environment, re∈RangeEnv:
  ⊢ (re ⊢ F : (h,d)) ∧ dom(e) = dom(re) ∧
    ∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p ⇒
      let c:=(e, {(X, s(e(X))) | X ∈ dom(e)})
      ∀h'∈ℕ : h'≥h ⇒
        T(F) → (p, s↓p, s(p), c) Ft
        ⇔
        T(F) → (p, s↑(max(0,p-h'),min(p,h')), s(p), c) Ft
```

We prove ∀F∈Formula S(F) by structural induction over F.

CASE 1. F = @X. T(F) = next(TV(X)).

We take Ftf,pf,sf,hf,df,ef,ref arbitrary but fixed and assume
Ftf∈Tformula, pf∈ℕ, sf∈Stream, hf∈ℕ, df∈ℕ∞, ef∈Environment,
ref∈RangeEnv.

Assume

```
(1.1)  ⊢ (ref ⊢ F : (hf,df))
(1.2') dom(ef) = dom(ref)
(1.2)  ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf
```

Define

```
(1.3) c:=(ef, {(X, sf(ef(X))) | X ∈ dom(ef)})
```

Take hf' arbitrary but fixed. Assume

```
(1.4) hf'≥hf
```

And prove

```
[1.5]  T(F) → (pf, sf↓pf, sf(pf), c) Ftf
       ⇔
```

T(F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) Ftf.

T(F)=next(TV(X)). By the definition of → for next(TV(X)), Ftf in [1.5] depends only whether X∈dom(c.1), which is the same in both sides if the equivalence. Hence, [1.5] holds.

CASE 2. F = ~F1. T(F) = next(TN(T(F1))).

We take Ftf,pf,sf,hf,df,ef,ref arbitrary but fixed and assume Ftf∈Tformula, pf∈ℕ, sf∈Stream, hf∈ℕ, df∈ℕ∞, ef∈Environment, ref∈RangeEnv.

Assume

(2.1) ⊢ (ref ⊢ F : (hf,df))
(2.2') dom(ef) = dom(ref)
(2.2) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

Define

(2.3) c:=(ef, {(X, sf(ef(X))) | X ∈ dom(ef)})

Take hf' arbitrary but fixed. Assume

(2.4) hf'≥hf

And prove

[2.5] T(F) → (pf, sf↓pf, sf(pf), c) Ftf
⇔
T(F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) Ftf.

From (2.1),by the definition of → for next(TN(T(F1))), we get

(2.6) ⊢ (ref ⊢ ~F1 : (hf,df)).

We prove [2.5] in both directions.

(⟹) We assume

(2.7) T(~F1) → (pf, sf↓pf, sf(pf), c) Ftf

and prove

[2.8] T(F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) Ftf.

From (2.7), we prove [2.8] by case distinction over Ftf:

C1. Ftf=next(TN(next(f'))) for some f'∈TFormulaCore, such that

(2.8) T(F1) → (pf, sf↓pf, sf(pf), c) next(f').

We instantiate the induction hypothesis with

From (2.8), by (2.6), (2.2), (2.3), (2.4), and the
induction hypothesis, we get

(2.9) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) next(f').

From (2.9), by the definition of → for T(¬F), we get [2.8].

C2. Ftf=done(false). This happens when

(2.10) T(F1) → (pf, sf↓pf, sf(pf), c) done(true).

From (2.10), by (2.6), (2.2), (2.3), (2.4), and the
induction hypothesis, we get

(2.11) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) done(true).

From (2.11), by the definition of → for T(~F), we get [2.8].

C3. Ftf=done(false). Similar to the cacse C2.

(⟸) We assume

(2.12) T(~F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) Ftf

and prove

[2.13] T(~F1) → (pf, sf↓pf, sf(pf), c) Ftf.

[2.13] can be proved by the same reasoning as the case (⟹) above. It finishes
the proof of CASE2.

CASE 3. F = F1&&F2. T(F) = next(TCS(T(F1),T(F2))).

We take Ftf,pf,sf,hf,df,ef,ref arbitrary but fixed and assume
Ftf∈Tformula, pf∈ℕ, sf∈Stream, hf∈ℕ, df∈ℕ∞, ef∈Environment,
ref∈RangeEnv.

Assume

(3.1) ⊢ (ref ⊢ F : (hf,df))
(3.2') dom(ef) = dom(ref)
(3.2) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

Define

(3.3) cf:=(ef, {(X, sf(ef(X))) | X ∈ dom(ef)})

Take hf'∈ℕ arbitrary but fixed. Assume

(3.4) hf'≥hf

And prove

[3.5] T(F) → (pf, sf↓pf, sf(pf), cf) Ftf
⇔
T(F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf.

From (3.1) and the assumption that hf∈ℕ, df∈ℕ∞, by the definition of ⊢ for F1&&F2, there exist h1,d1,h2∈ℕ,d2∈ℕ∞ such that

(3.6) ⊢ (ref ⊢ F1 : (h1,d1))
(3.7) ⊢ (ref ⊢ F2 : (h2,d2))
(3.8) hf=max(h1,h2+d1).

We prove [3.5] in both directions.

(⟹) We assume

(3.9) T(F1&&F2) → (pf, sf↓pf, sf(pf), cf) Ftf

and prove

[3.10] T(F1&&F2) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf.

From (3.9), we prove [3.10] by case distinction over Ftf:

C1. Ftf=next(TCS(next(f1),T(F2))) for some f1∈TFormulaCore such that

(3.11) T(F1) → (pf, sf↓pf, sf(pf), cf) next(f1).

We instantiate the induction hypothesis as F:=F1, Ft:=next(f1), p:=pf, s:=sf, h:=h1, d:=d1 (since d1∈ℕ, we have d1∈ℕ∞), e:=ef, re:=ref, c:=cf, h':=hf. Then from the IH by (3.2'),(3.2),(3.3),(3.4),(3.6), (3.8),(3.11) we get

(3.12) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) next(f').

From (3.12), by the definition of → for T(F1&&F2), we get [3.10].

C2. Ftf=done(false). This happens when

(3.13) T(F1) → (pf, sf↓pf, sf(pf), cf) done(false).

We instantiate the induction hypothesis as F:=F1, Ft:=done(false), p:=pf, s:=sf, h:=h1, d:=d1 (since d1∈ℕ, we have d1∈ℕ∞), e:=ef, re:=ref, c:=cf, h':=hf. Then from the IH by (3.2'),(3.2),(3.3),(3.4),(3.6), (3.8),(3.13), we get

(3.14) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) done(false).

From (3.14), by the definition of → for T(F1&&F2), we get [3.10].

C3. Ftf=Ft2 for some Ft2∈TFormula. This happens when we have

(3.15) T(F1) → (pf, sf↓pf, sf(pf), cf) done(true) and
(3.16) T(F2) → (pf, sf↓pf, sf(pf), cf) Ft2.

From (3.4,3.8), we have

(3.17) hf'≥hf≥h1
(3.18) hf'≥hf≥h2

We instantiate the induction hypothesis as F:=F1, Ft:=done(true), p:=pf, s:=sf, h:=h1, d:=d1 (since d1∈ℕ, we have d1∈ℕ∞), e:=ef, re:=ref, c:=cf, h':=hf'. Then from the IH by (3.2'),(3.2),(3.3),(3.6),(3.17), (3.15) we get

(3.19) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) done(true).

Next, we instantiate the induction hypothesis as F:=F2, Ft:=Ft2, p:=pf, s:=sf, h:=h1, d:=d2, e:=ef, re:=ref, c:=cf, h':=hf. Then from the IH by (3.2'),(3.2),(3.3),(3.7),(3.16),(3.18) we get

(3.20) T(F2) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ft2.

From (3.19) and (3.20), by the definition of → for T(F1&&F2), we get [3.10].

(⟸) We assume

(3.21) T(F1&&F2) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf.

and prove

[3.22] T(F1&&F2) → (pf, sf↓pf, sf(pf), cf) Ftf

[3.22] can be proved by the same reasoning as the case (⟹) above. It finishes the proof of CASE3.

CASE 4. F = F1/\F2. T(F) = next(TCP(T(F1),T(F2))).

\---

We take Ftf,pf,sf,hf,df,ef,ref arbitrary but fixed and assume Ftf∈Tformula, pf∈ℕ, sf∈Stream, hf∈ℕ, df∈ℕ∞, ef∈Environment, ref∈RangeEnv.

Assume

(4.1) ⊢ (ref ⊢ F : (hf,df))
(4.2') dom(ef) = dom(ref)
(4.2) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

Define

(4.3) cf:=(ef, {(X, sf(ef(X))) | X ∈ dom(ef)})

Take hf' arbitrary but fixed. Assume

(4.4) hf'≥hf

And prove

[4.5] T(F) → (pf, sf↓pf, sf(pf), cf) Ftf
⇔
T(F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf

From (4.1) and the assumption that hf∈ℕ, df∈ℕ∞, by the definition of ⊢ for F1∧F2, there exist h1,h2∈ℕ,d1,d2∈ℕ∞ such that

(4.6) ⊢ (ref ⊢ F1 : (h1,d1))
(4.7) ⊢ (ref ⊢ F2 : (h2,d2))
(4.8) hf=max(h1,h2).

From (4.4,4.8), we have

(4.9) hf'≥hf≥h1
(4.10) hf'≥hf≥h2

We prove [4.5] in both directions.

(⟹) We assume

(4.11) T(F1/\F2) → (pf, sf↓pf, sf(pf), cf) Ftf

and prove

[4.12] T(F1/\F2) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf.

From (4.11), we prove [4.10] by case distinction over Ftf:

C1. Ftf=next(TCS(next(f1),next(f2))) for some f1,f2∈TFormulaCore such that

(4.13) T(F1) → (pf, sf↓pf, sf(pf), cf) next(f1).
(4.14) T(F2) → (pf, sf↓pf, sf(pf), cf) next(f2).

We instantiate the induction hypothesis as F:=F1, Ft:=next(f1), p:=pf, s:=sf, h:=h1, d:=d1, e:=ef, re:=ref, c:=cf, h':=hf'. Then from the IH, by (4.6),(4.2'),(4.2),(4.3),(4.9),(4.13) we get

(4.15) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) next(f1).

Next, we instantiate the induction hypothesis as F:=F1, Ft:=next(f2), p:=pf, s:=sf, h:=h2, d:=d2, e:=ef, re:=ref, c:=cf, h':=hf'. Then from the IH, by (4.7),(4.2'),(4.2),(4.3),(4.10),(4.14) we get

(4.16) T(F2) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) next(f2).

From (4.15,4.16), by the definition of → for T(F1∧F2), we get [4.12].

C2. Ftf=next(f1) for some f1∈TFormulaCore such that

```
    (4.17) T(F1) → (pf, sf↓pf, sf(pf), c) next(f1).
    (4.18) T(F2) → (pf, sf↓pf, sf(pf), c) done(true).

    By the same reasoning as in C1 above we get that [4.12] holds.

C3. Ftf=done(false). This happens in one of the following possible cases:

    C3.1
       (4.19) T(F1) → (pf, sf↓pf, sf(pf), c) next(f1).
       (4.20) T(F2) → (pf, sf↓pf, sf(pf), c) done(false).

       By the same reasoning as in C1 above we get that [4.12] holds.

    C3.2
       (4.21) T(F1) → (pf, sf↓pf, sf(pf), c) done(false).

       We instantiate the induction hypothesis as F:=F1,Ft:=done(false),
       p:=pf, s:=sf, h:=h1, d:=d1, e:=ef, re:=ref, c:=cf, h':=hf'.
       Then from the IH, by (4.6),(4.2'),(4.2),(4.3),(4.9),(4.21)
       we get

       (4.22) T(F1) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) done(false).

       From (4.22), by the definition of → for T(F1∧F2), we get [4.12].

C4. Ftf=Ft2 for some Ft2∈TFormula. This happens when

    (4.23) T(F1) → (pf, sf↓pf, sf(pf), c) done(true).
    (4.24) T(F2) → (pf, sf↓pf, sf(pf), c) Ft2.

    By the same reasoning as in C1 above we get that [4.12] holds.

(⟸) We assume

(4.25) T(F1/\F2) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c) Ftf.

and prove

[4.26] T(F1/\F2) → (pf, sf↓pf, sf(pf), c) Ftf

[4.26] can be proved by the same reasoning as the case (⟹) above.

It finishes the proof of CASE 4.


CASE 5. F = forall X in B1..B2:F1  T(F) = next(TA(X, T(B1), T(B2), T(F1))).
-------------------------------------------------------------------------
We take Ftf,pf,sf,hf,df,ef,ref arbitrary but fixed and assume
Ftf∈Tformula, pf∈ℕ, sf∈Stream, hf∈ℕ, df∈ℕ∞, ef∈Environment,
ref∈RangeEnv.

Assume
```

(5.1) ⊢ (ref ⊢ F : (hf,df))
(5.2') dom(ef) = dom(ref)
(5.2) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

Define

(5.3) cf:=(ef, {(X, sf(ef(X))) | X ∈ dom(ef)})

Take hf' arbitrary but fixed. Assume

(5.4) hf'≥hf

And prove

[5.5] T(F) → (pf, sf↓pf, sf(pf), cf) Ftf
       ⇔
       T(F) → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf

Let b1,b2∈BoundValue and Ft1∈TFormula be such that

(5.6) b1=T(B1)
(5.7) b2=T(B2)
(5.7') Ft1=T(F1)

From (5.1), taking into account the assumptions hf∈ℕ and df∈ℕ∞,
we know by the definition of ⊢ for "forall" for some l1∈ℤ,
u1,l2,u2∈ℤ∞, h1∈ℕ, d1∈ℕ∞:

(5.I.1) ⊢ (ref ⊢ B1 : (l1, u1))
(5.I.2) ⊢ (ref ⊢ B2 : (l2, u2))
(5.I.3) ⊢ (ref[X↦(l1, u2)] ⊢ F1 : (h1, d1))
(5.I.4) hf = max∞(h1, ℕ∞(-i(l1))) = (by h1∈ℕ, l1∈ℤ) max(h1,|l1|).
(5.I.5) df = max∞(d1, ℕ∞(u2))

We define

(5.I.6) p1 = b1(cf)
(5.I.7) p2 = b2(cf)

From (5.I.1) (5.I.2), (5.2'), (5.2), (5.3), (5.6), (5.7), (5.I.6),
(5.I.7), we know by Lemma 9 (soundness of bound analysis)

(5.I.B.1) l1 +i pf ≤i p1 ≤i u1 +i pf
(5.I.B.2) l2 +i pf ≤i p2 ≤i u2 +i pf

(In fact, instead of l1 +i pf we can write l1 + pf in (5.I.B.1),
because neither l1 nor pf can be ∞.)

Instantiating the induction hypothesis S(F1) with s:=sf, h:=h1, d:=d1,
re:=ref[X↦(l1, u2)], we know with (5.I.3), (5.2'), (5.3), (5.7')

(5.I.F)

  ∀Ft∈TFormula, p∈ℕ, e∈Environment:

```
    dom(e) = dom(ef)∪{X} ∧
    (∀Y∈dom(ef)\{X}): ref(Y).1 +i p ≤i e(Y) ≤i ref(Y).2 +i p) ∧
    (l1 +i p ≤i e(X) ≤i u2 +i p) ⇒
      let c:=(e, {(Y, sf(e(Y))) | Y ∈ dom(ef)∪{X}})
        ∀h'∈ℕ : h'≥h1 ⇒
          Ft1 → (p, sf↓p, sf(p), c) Ft
            ⇔
          Ft1 → (p, sf↑(max(0,p-h'),min(p,h')), sf(p), c) Ft
```

We prove [5.5] in both directions.

(⟹) We assume

```
(5.8) next(TA(X,b1,b2,Ft1)) → (pf, sf↓pf, sf(pf), cf) Ftf
```

and prove

```
[5.9] next(TA(X,b1,b2,Ft1))
        →(pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf.
```

From (5.8), we have two cases:

CASE 1 (Rule 1 for TA)

We know from the rule, (5.I.6) and (5.I.7) that

```
(5.10.1) p1 = ∞ ∨ p1 >∞ b2(cf)
(5.10.2) Ftf = done(true)
```

From (5.I.6), (5.I.7), (5.10.1), (5.10.2), we can derive
with "Rule 1 for TA" [5.9].

CASE 2: (Rule 2 for TA)

We know from the rule, (5.I.6) and (5.I.7) that

```
(5.16)  p1 ≠ ∞
(5.16') p1 ≤∞ p2
(5.17)  next(TA0(X,p1,p2,Ft1)) → (pf, sf↓pf, sf(pf), cf) Ftf
```

To prove [5.9], it suffices, by "Rule 2 for TA", together with
(5.I.6), (5.I.7), (5.16), (5.16') to prove

```
[5.21] next(TA0(X,p1,p2,Ft1))
         →(pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), cf) Ftf
```

Subcase 1.
(5.23) pf < p1.

In this case from (5.17) and "Rule 1 for TA0" we have
Ftf=next(TA0(X,p1,p2,Ft1)). Then [5.21] follows from (5.17), (5.23) and
"Rule 1 for TA0".

Subcase 2.
(5.24) $pf \geq p1$.

We define

(5.25) ms := $sf\uparrow(\max(0,pf-hf'),\min(pf,hf'))$

Before proving [5.21], we establish the following auxiliary fact:

[aux] $\forall p0$: $p1 \leq p0 <_\infty \min_\infty(pf,p2+_\infty 1) \Rightarrow 0 \leq p0-pf+|ms| < |ms|$
============================
Proof of [aux]: Take arbitrary p0 and assume

(aux1) $p1 \leq p0$
(aux2) $p0 <_\infty \min_\infty(pf,p2+_\infty 1)$

We have to show

[aux3] $0 \leq p0-pf+|ms|$
[aux4] $p0-pf+|ms| < |ms|$

From (aux2) we have p0<pf and thus [aux4] holds.

To show [aux3], we show

[aux3.1] $pf \leq p0+|ms|$

From (5.25), we know

(aux3) $|ms| = \min(pf,hf')$

From (aux3), to show [aux3.1], it suffices to show

[aux3.2] $pf \leq p0+\min(pf,hf')$

We proceed by case distinction:

(aux4) Case pf <= hf'
- ------------------

From (aux4), to show [aux3.2], it suffices to show

[aux3.2.1] $pf \leq p0+pf$

From p0 in Nat, we have

(aux5) p0 >= 0

and thus [aux3.2.1]

(aux6) Case pf > hf'

From (aux6), to show [aux3.2], it suffices to show

```
[aux3.2.2] pf ≤ p0+hf'
```

From (5.4) we know hf'≥hf. It thus suffices to show

```
[aux3.2.3] pf ≤ p0+hf
```

From (aux5.1.4), it suffices to show

```
[aux3.2.4] pf ≤ p0+max(h1,|l1|).
```

We know

```
(aux7) p0+max(h1,|l1|) ≥ (by (aux1))
       p1+max(h1,|l1|) ≥ (by l1∈ℤ)
       p1-l1 ≥ (by 5.I.B.1) pf
```

and thus have [aux3.2.4].

It proves [aux].
===============

From [aux] we can conclude

```
(5.25') ∀p0: p1 ≤ p0 <∞ min∞(pf,p2+∞1) ⇒ (sf↓pf)(p0)=ms(p0-pf+|ms|).
```

Now, to prove [5.21], it suffices by "Rule 2 for TA0" to prove

```
[5.26] next(TA1(X,p2,Ft1,fs)) → (pf, ms, sf(pf), cf) Ftf
```

where

```
(5.27) fs = {(p0,Ft1,(cf.1[X↦p0],cf.2[X↦ms(p0-pf+|ms|)])) |
                 p1 ≤ p0 <∞ min∞(pf,p2+∞1)}.
```

We prove [5.26] by case distinction over Ftf.

(c1) Ftf=done(false)

We prove

```
[c1.1] next(TA1(X,p2,Ft1,fs)) → (pf, ms, sf(pf), cf) done(false).
```

To prove [c1.1], by Def.→ we need to prove

```
[c1.2] ∃t∈ℕ,g∈TFormula,c∈Context:
           (t,g,c)∈fs0 ∧ ⊢ g →(pf,ms,sf(pf),c) done(false),
```

where

```
(c1.3) fs0 =
           if pf >∞ p2 then fs else fs ∪ {(pf,Ft1,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}
```

From (5.17), by (c1) we know

```
(c1.4) next(TA1(X,p2,Ft1,fs')) →(pf,sf↓pf,sf(pf),cf) done(false)
```

where (since p0-pf+|sf↓pf|=p0)

```
(c1.5) fs' = {(p0,Ft1,(cf.1[X↦p0],cf.2[X↦(sf↓pf)(p0)])) |
                  p1 ≤ p0 <∞ min∞(pf,p2+∞1)}.
```

From (c1.4) we know by the definition of →

```
(c1.6) ∃t∈ℕ,g∈TFormula,c∈Context:
          (t,g,c)∈fs1 ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) done(false),
```

where

```
(c1.7) fs1 =
         if pf >∞ p2 then fs' else fs' ∪ {(pf,Ft1,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}
```

From (c1.6), we have (t1,g1,c1) such that

```
(c1.8) (t1,g1,c1)∈fs1 and
(c1.9) ⊢ g1 →(pf,sf↓pf,sf(pf),c1) done(false).
```

From (c1.8), (c1.7), (c1.5) we see that

```
(c1.10) g1=Ft1
```

and, hence, T(F1)=g1.

From (c1.8), (c1.7), (c1.5), we have

```
Case 1: c1 = (cf.1[X↦t1],cf.2[X↦(sf↓pf)(t1)] ∧ p1 ≤ t1 <∞ min∞(pf,p2+∞1)
Case 2: c1 = (cf.1[X↦t1],cf.2[X↦sf(t1)]) ∧ pf ≤∞ p2 ∧ t1 = pf
```

and with (5.24) consequently (in both cases)

```
(c1.12.1) p1 ≤ t1 ≤∞ min∞(pf, p2)
(c1.12.2) c1 = (cf.1[X↦t1],cf.2[X↦(sf↓(pf+1))(t1)])
```

We have from (c1.12.2)

```
(c1.13.1) c1.1(X) = t1
```

We have from (5.2), (5.3) and (c1.12.2),

```
(c1.13.2) ∀Y∈dom(cf.1)\{X}: ref(Y).1 +i pf ≤i c1.1(Y) ≤i ref(Y).2 +i pf
```

From (c1.12.1), (5.I.B.1) and (5.I.B.2), we know

```
(c1.13.3) l1 +i pf ≤i t1 ≤i u2 +i pf
```

We instantiate (5.I.F) with Ft:=done(false), p:=pf, e:=c1.1.
With (5.2'), (5.3), (c1.12.2), (c1.13.2), (c1.13.3), we then have

```
(c1.14) ∀h1'∈ℕ : h1'≥h1 ⇒
          Ft1 → (pf, sf↓(pf), sf(pf), c1) done(false)
          ⇔
          Ft1 → (pf, sf↑(max(0,pf-h1'),min(pf,h1')), sf(pf), c1) done(false)
```

Since (c1.14) is true for all h1'≥h1, it is true, in particular, for hf', because by (5.4) we have hf'≥hf, and in itself, hf≥h1 by (5.I.4). Hence, from (c1.14) we get

```
(c1.15)
          Ft1 → (pf, sf↓pf, sf(pf), c1) done(false)
          ⇔
          Ft1 → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c1) done(false)
```

From (c1.15) and (c1.9) we get

```
(c1.16) Ft1 → (pf, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c1) done(false)
```

(c1.16), by (5.25), proves the second conjunct of [c1.2].

Hence, it remains to prove the first conjunct of [c1.2]:

[c1.3] (t1,g1,c1)∈fs0.

By (c1.8), (t1,g1,c1)∈fs1. By (c1.7) it means either

(c1.17) (t1,g1,c1)=(pf,Ft1,(cf.1[X↦pf],cf.2[X↦sf(pf)]))

or

(c1.18) (t1,g1,c1)∈fs'.

From (c1.17) we get [c1.3] due to the definition of fs0 in (c1.3).

From (c1.18) we have

(c1.19) (t1,g1,c1)=(p0,Ft1,(cf.1[X↦p0],c.2[X↦(sf↓pf)(p0)]))

for some p1 ≤ p0 <∞ min∞(pf,p2+∞1).

From (5.25'), (c1.19) and the definition of fs in(5.27) we get

(c1.21) (t1,g1,c1)∈fs.

From (c1.2) we have fs ⊆ fs0 and, hence, [c1.3] holds also in this case.

It proves (c1).

(c2) Ftf=done(true)

We prove

```
[c2.1] next(TA1(X,p2,Ft1,fs)) → (pf, ms, sf(pf), cf) done(true).
```

To prove [c2.1], by Def. of → ("Rule 2 for TA1") we need to prove

```
[c2.2] ¬∃t∈ℕ,g∈TFormula,c∈Context:
            (t,g,c)∈fs0 ∧ ⊢ g →(pf,ms,sf(pf),c) done(false) and
[c2.3] fs1 = ∅ ∧ pf ≥∞ p2
```

where

```
(c2.4) fs0 =
          if pf >∞ p2 then fs else fs ∪ {(pf,Ft1,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}
(c2.5) fs1 = { (t,next(fc),c) ∈ TInstance |
                 ∃g∈TFormula: (t,g,c)∈fs0 ∧ ⊢ g →(pf,ms,sf(pf),c) next(fc) }
```

From (5.17), by (c2) we know

```
(c2.5') next(TA0(X,p1,p2,Ft1)) →(pf,sf↓pf,sf(pf),cf) done(true).
```

From (c2.5') and (5.24), by the definiton of → ("Rule 2 for TA0") we know

```
(c2.6) next(TA1(X,p2,Ft1,fs')) →(pf,sf↓pf,sf(pf),cf) done(true),
```

where

```
(c2.6') fs' = {(p0,Ft1,(cf.1[X↦p0],cf.2[X↦(sf↓pf)(p0-pf+|sf↓pf|)])) |
                  p1 ≤ p0 <∞ min∞(pf,p2+∞1)}
```

Since p0-pf+|sf↓pf|=p0, from (c2.6') we get

```
(c2.7) fs' = {(p0,Ft1,(cf.1[X↦p0],cf.2[X↦(sf↓pf)(p0)])) |
                 p1 ≤ p0 <∞ min∞(pf,p2+∞1)}
```

From (c2.6), by Def. of → ("Rule 2 for TA1") we know

```
(c2.8) ¬∃t∈ℕ,g∈TFormula,c∈Context:
            (t,g,c)∈fs0' ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) done(false) and
(c2.9) fs1' = ∅ ∧ pf ≥∞ p2
```

where

```
(c2.10) fs0' =
         if pf >∞ p2 then fs' else fs' ∪ {(pf,Ft1,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}
(c2.11) fs1' = { (t,next(fc),c) ∈ TInstance |
                 ∃g∈TFormula: (t,g,c)∈fs0' ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) next(fc) }.
```

From (5.25'), (5.27) and (c2.7) we get

```
(c2.13) fs = fs',
```

which, by (c2.4) and (c2.10), implies

```
(c2.14) fs0=fs0'.
```

To prove [c2.2], we take

(c2.15) (t0,g0,c0)∈fs0

and prove that

[c2.16] g0 →(pf,ms,sf(pf),c0) done(false) does not hold.

From (c2.15) and (c2.14) we have

(c2.17) (t0,g0,c0)∈fs0'.

From (c2.17) and (c2.8) we know

(c2.18) g0 →(pf,sf↓pf,sf(pf),c0) done(false) does not hold.

From (c2.4), (5.27) and (c2.15) we get

(c2.19) g0=Ft1 and two cases:

```
Case 1: t0=p0 ∧ c0=cf.1[X↦p0],cf.2[X↦ms(p0-pf+|ms|)] ∧
        p1 ≤ p0 <∞ min∞(pf,p2+∞1) ∧ pf >∞ p2
Case 2: t0=pf ∧ c0=cf.1[X↦pf],cf.2[X↦sf(pf)] ∧ pf ≤∞ p2
```

These cases can be rewritten and simplified (taking into account (5.25')
and (5.24)) into

```
Case 1: c0=cf.1[X↦t0],cf.2[X↦(sf↓pf)(t0)] ∧ p1 ≤ t0 <∞ min∞(pf,p2)
Case 2: c0=cf.1[X↦t0],cf.2[X↦sf(t0)] ∧ p1 ≤ pf ≤∞ p2 ∧ pf = t0.
```

Consequently, in both cases we get

(c2.20) p1 ≤ t0 <∞ min∞(pf,p2) and
(c2.21) c0=cf.1[X↦t0],cf.2[X↦(sf↓(pf+1))(t0)]

From (c2.21) we have

(c2.22) c0.1(X)=t0.

From (5.2), (5.3), and (c2.21) we get

(c2.23) ∀Y∈dom(cf.1)\{X}: ref(Y).1 +i pf ≤i c0.1(Y) ≤i ref(Y).2 +i pf.

From (c2.20), (5.I.B.1) and (5.I.B.2), we know

(c2.24) l1 +i pf ≤i t0 ≤i u2 +i pf.

We instantiate (5.I.F) with Ft:=done(false), p:=pf, e:=c0.1.
With (5.2'), (5.3), (c2.21), (c2.22), (c2.23), (c2.24), we then have

```
(c2.25)  ∀h1'∈ℕ : h1'≥h1 ⇒
          Ft1 → (pf, sf↓pf, sf(pf), c0) done(false)
            ⇔
          Ft1 → (p, sf↑(max(0,pf-h1'),min(pf,h1')), sf(pf), c0) done(false)
```

Since (c2.25) is true for all h1'≥h1, it is true, in particular, for hf', because by (5.4) we have hf'≥hf, and in itself, hf≥h1 (5.I.4). Hence, from (c2.25) we get

```
(c2.26)   Ft1 → (pf, sf↓pf, sf(pf), c0) done(false)
            ⇔
          Ft1 → (p, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c0) done(false)
```

From (c2.26), (c2.18), and (c2.19) we get

```
(c2.27) Ft1 → (p, sf↑(max(0,pf-hf'),min(pf,hf')), sf(pf), c0) done(false)
        does not hold.
```

From (c2.27), by (5.25), we get [c2.16].

To prove [c2.3], note that from (c2.14), (c2.5) and (c2.11) we get

(c2.28) fs1 = fs1'.

Now [c2.3] follows from (c2.28) and (c2.9). It proves (c2).

(c3) Ftf=next(TA1(X,p2,Ft1,fs'))

We prove

```
[c3.1] next(TA1(X,p2,Ft1,fs)) → (pf, ms, sf(pf), cf) next(TA1(X,p2,Ft1,fs')).
```

To prove [c3.1], by Def. of → (("Rule 3 for TA1") we need to prove

```
[c3.2] ¬∃t∈ℕ,g∈TFormula,c∈Context:
           (t,g,c)∈fs0 ∧ ⊢ g →(pf,ms,sf(pf),c) done(false) and
[c3.3] ¬(fs1 = ∅ ∧ pf ≥∞ p2)
```

where

```
(c3.4) fs0 =
          if pf >∞ p2 then fs else fs ∪ {(pf,f,(cf.1[X↦p],cf.2[X↦sf(pf)]))}
(c3.5) fs1 = { (t,next(fc),c) ∈ TInstance |
                ∃g∈TFormula: (t,g,c)∈fs0 ∧ ⊢ g →(pf,ms,sf(pf),c) next(fc) }
```

From (5.17) by (c3) we know

```
(c3.5') next(TA0(X,p1,p2,Ft1)) → (pf, sf↓pf, sf(pf), cf) next(TA1(X,p2,Ft1,fs')).
```

From (c3.5') and by the definiton of → ("Rule 2 for TA0") we know

```
(c3.6) next(TA1(X,p2,Ft1,fs')) →(pf,sf↓pf,sf(pf),cf) next(TA1(X,p2,Ft1,fs'))
```

where

```
(c3.6') fs' = {(p0,Ft1,(cf.1[X↦p0],cf.2[X↦(sf↓pf)(p0-pf+|sf↓pf|)])) |
```

```
                 p1 ≤ p0 <∞ min∞(pf,p2+∞1)}
```

Since p0-pf+|sf↓pf|=p0, from (c3.6’) we get

```
(c3.7) fs’ = {(p0,Ft1,(cf.1[X↦p0],cf.2[X↦(sf↓pf)(p0)])) |
                 p1 ≤ p0 <∞ min∞(pf,p2+∞1)}
```

From (c3.6), by Def. of → ("Rule 3 for TA1") we know

```
(c3.8) ¬∃t∈ℕ,g∈TFormula,c∈Context:
           (t,g,c)∈fs0’ ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) done(false) and
(c3.9) ¬ (fs1’ = ∅ ∧ pf ≥∞ p2)
```

where

```
(c3.10) fs0’ =
         if pf >∞ p2 then fs’ else fs’ ∪ {(pf,Ft1,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}
(c3.11) fs1’ = { (t,next(fc),c) ∈ TInstance |
                 ∃g∈TFormula: (t,g,c)∈fs0’ ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) next(fc) }.
```

From (5.25’), (5.27) and (c3.7) we get

(c3.13) fs = fs’,

which, by (c3.4) and (c3.10), implies

(c3.14) fs0=fs0’.

Now [c3.2] can be proved in the same as [c2.2] was proved above.

To prove [c3.3], note that from (c3.14), (c3.5) and (c3.11) we get

(c3.28) fs1 = fs1’.

Now [c3.3] follows from (c3.28) and (c3.9). It proves (c3).

Hence, the direction (⟹) is proved.

(⟸) This direction can be proved with the same reasoning as (⟹).

It finishes the proof of CASE 5.

It finishes the proof of Lemma 3.

A.6 Lemma 4: n-Step Reductions to **done** Formulas for TN, TCS, TCP

Statement 1. TN Formulas.

```
 ∀F∈Formula, n∈ℕ, p∈ℕ, s∈Stream, e∈Environment, Ft∈TFormula :
   T(F) →*(n,p,s,e) done(false) ⇒ next(TN(T(F))) →*(n,p,s,e) done(true) ∧
   T(F) →*(n,p,s,e) done(true) ⇒ next(TN(T(F))) →*(n,p,s,e) done(false)
```

Proof

We take Ff, sf, ef arbitrary but fixed and prove the formula

```
  ∀n∈ℕ, p∈ℕ :
     T(Ff) →*(n,pf,sf,ef) done(false) ⇒
              next(TN(T(Ff))) →*(n,pf,sf,ef) done(true)
     ∧
     T(Ff) →*(n,pf,sf,ef) done(true) ⇒
              next(TN(T(Ff))) →*(n,p,s,e) done(false)
```

by induction over n. Since T(Ff) is a next formula, for n=0 the antecedents of both conjuncts are false and the statement is trivially true.

Assume

```
(TN.1)   ∀p∈ℕ:
            T(Ff) →*(n,p,sf,ef) done(false) ⇒
            next(TN(T(Ff))) →*(n,p,sf,ef) done(true)
(TN.2)   ∀p∈ℕ:
            T(Ff) →*(n,pf,sf,ef) done(true) ⇒
            next(TN(T(Ff))) →*(n,p,s,e) done(false)
```

Prove

```
[TN.3]   ∀p∈ℕ:
            T(Ff) →*(n+1,p,sf,ef) done(false) ⇒
            next(TN(T(Ff))) →*(n+1,p,sf,ef) done(true)
and
[TN.4]   ∀p∈ℕ:
            T(Ff) →*(n+1,p,sf,ef) done(true) ⇒
            xsnext(TN(T(Ff))) →*(n+1,p,s,e) done(false)
```

To prove [TN.3], we take pf arbitrary but fixed, assume

```
(TN.5)  T(Ff) →*(n+1,pf,sf,ef) done(false)
```

and prove

```
[TN.6]  next(TN(T(Ff))) →*(n+1,pf,sf,ef) done(true)
```

From (TN.5) by definition →* without history we know that there exists Ft∈TFormula such that

```
(TN.7)  T(Ff) →(pf,sf↓pf,sf(pf),c) Ft
(TN.8)  Ft →*(n,pf+1,sf,ef) done(false)
```

where c = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)}).

We proceed by case distinction over Ft.

Case 'next': If Ft is a next formula, then there exists F1∈Formula such that
----
(TN.9) Ft=T(F1)

From (TN.9) and (TN.8) by (TN.1) we get

(TN.10) next(TN(T(F1))) →*(n,pf+1,sf,ef) done(true)

From (TN.7) by the definition of → we get

(TN.11) next(TN(T(Ff))) →(pf,sf↓pf,sf(pf),c) next(TN(T(F1)))

From (TN.11) and (TN.10) by the definition of →* without history we get [TN.6].

Case 'done': If Ft is a 'done' formula, then by (TN.8), we have
----
(TN.12) n=0 and
(TN.13) Ft=done(false).

From (TN.7) and (TN.13), by the definition of →, we get

(TN.14) next(TN(T(Ff))) →(pf,sf↓pf,sf(pf),c) done(true).

On the other hand, from the definition of →* we know

(TN.15) done(true) →*(0,pf+1,sf,ef) done(true).

From (TN.14), (TN.15), (TN.12), by the definition of →* we get [TN.6].

Hence, we proved [TN.6] for both cases of Ft. This proves [TN.3].
[TN.4] can be proved analogously.

Statement 2. TCS Formulas.

```
 ∀ p∈ℕ, s∈Stream, e∈Environment :
   ∀Ft1,Ft2∈TFormula, n∈ℕ,
      n>0 ∧ Ft1 →*(n,p,s,e) done(false) ⇒
      next(TCS(Ft1,Ft2)) →*(n,p,s,e) done(false) ∧
   ∀Ft1,Ft2∈TFormula, n1,n2∈ℕ, b∈Bool:
      n1>0 ∧ n2>0 ∧Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(b) ⇒
      next(TCS(Ft1,Ft2)) →*(max(n1,n2),p,s,e) done(b)
```

Proof

We split the statement in two:

```
[TCS1] ∀p∈ℕ, s∈Stream, e∈Environment,
         Ft1,Ft2∈TFormula, n∈ℕ :
            n>0 ∧ Ft1 →*(n,p,s,e) done(false) ⇒
            next(TCS(Ft1,Ft2)) →*(n,p,s,e) done(false)

[TCS2] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ, b∈Bool :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(b) ⇒
       next(TCS(Ft1,Ft2)) →*(max(n1,n2),p,s,e) done(b).


Proof of [TCS1]
--------------

We take sf,ef arbitrary but fixed and define

Φ(n) :⇔
    ∀p∈ℕ, Ft1,Ft2∈TFormula:
       n>0 ∧ Ft1 →*(n,p,sf,ef) done(false) ⇒
       next(TCS(Ft1,Ft2)) →*(n,p,sf,ef) done(false))

We prove ∀n∈ℕ: Φ(n) by induction over n. For n=0 the formula is trivially true.
We start the induction from 1. Prove:

[TCS1.a] Φ(1) and
[TCS1.b] ∀n∈ℕ: Φ(n) ⇒ Φ(n+1)

Proof of [TCS1.a]
-----------------
We take pf,Ft1f,Ft2f arbitrary but fixed and assume

(TCS1.1) 1>0
(TCS1.2) Ft1f →*(1,pf,sf,ef) done(false).

We want to prove

[TCS1.3] next(TCS(Ft1f,Ft2f)) →*(1,pf,sf,ef) done(false).

From (TCS1.2), by the definition of →* without history, there exists
Ft∈TFormula such that

(TCS1.4) Ft1f →(p,sf↓pf,sf(pf),c) Ft and
(TCS1.5) Ft →*(0,pf+1,sf,ef) done(false)

where

(TCS1.6) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCS1.5), by the definition of →* without history, we get

(TCS1.7) Ft=done(false).

From (TCS1.7) and (TCS1.4), by the definition of → for TCS, we get

(TCS1.8) next(TCS(Ft1f,Ft2f)) →(p,sf↓pf,sf(pf),c) done(false).
```

From (TCS1.8, TCS1.5, TCS1.7, TCS1.6), by the definition of →* without history, we get [TCS1.2].

This finishes the proof of [TCS1.a]

Proof of [TCS1.b]

We take n arbitrary but fixed, assume

```
(TCS1.8)  ∀p∈ℕ, Ft1,Ft2∈TFormula:
            n>0 ∧ Ft1 →*(n,p,sf,ef) done(false) ⇒
            next(TCS(Ft1,Ft2)) →*(n,p,sf,ef) done(false))
```

and prove

```
[TCS1.9]  ∀p∈ℕ, Ft1,Ft2∈TFormula:
           n+1>0 ∧ Ft1 →*(n+1,p,sf,ef) done(false) ⇒
           next(TCS(Ft1,Ft2)) →*(n+1,p,sf,ef) done(false)).
```

To prove [TCS1.9], we take pf,Ft1f,Ft2f arbitrary but fixed, assume

```
(TCS1.10) n+1>0
(TCS1.11) Ft1f →*(n+1,pf,sf,ef) done(false)
```

and prove

```
[TCS1.12] next(TCS(Ft1f,Ft2f)) →*(n+1,p,sf,ef) done(false)).
```

From (TCS1.11), by the definition of →* without history, there exists Ft∈TFormula such that

```
(TCS1.13) Ft1f →(pf,sf↓pf, sf(pf),c) Ft
(TCS1.14) Ft →*(n,pf+1,sf,ef) done(false)
```

where

```
(TCS1.15) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).
```

We proceed by case distinction over Ft.

Case 1. Ft=next(f) for some f∈TFormulaCore

From (TCS1.13), by the definition of → for TCS, we get

```
(TCS1.16) next(TCS(Ft1f,Ft2f)) →(pf,sf↓pf, sf(pf),c) next(TCS(Ft,Ft2f))
```

Since Ft is a 'next' formula, we have

```
(TCS1.17) n>0.
```

From (TCS1.17) and (TCS1.14), by the induction hypothesis (TCS1.8) we get

(TCS1.18) next(TCS(Ft,Ft2f)) →*(n,pf+1,sf,ef) done(false)

From (TCS1.10), (TCS1.15), (TCS1.16), and (TCS1.18), by the definition of →*
without history, we get [TCS1.12]

Case 2. Ft=done(b) for some b∈Bool

In this case we have

(TCS1.19) n=0 (a 'done' formula can be reduced only in 0 steps)
(TCS1.20) b=false.

Then from (TCS1.13) and (TCS1.20), by the definition of → for TCS we get

(TCS1.21) next(TCS(Ft1f,Ft2f)) →(pf,sf↓pf, sf(pf),c) done(false).

From (TCS1.14), (TCS1.19), and (TCS1.20), we have

(TCS1.22) done(false) →*(0,pf+1,sf,ef) done(false).

From (TCS1.19), (TSC1.15), (TSC1.21), (TCS1.22), by the definition of →*
without history, we get [TCS1.12].

This finishes the proof of [TCS1].

==

Proof of [TCS2]

Recall

[TCS2] ∀s∈Stream, e∈Environment,p∈ℕ, Ft1,Ft2∈TFormula, n1,n2∈ℕ, b∈Bool:
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(b) ⇒
       next(TCS(Ft1,Ft2)) →*(max(n1,n2),p,s,e) done(b).

We take sf,ef,bf arbitrary but fixed and define

Φ(n1) :⇔
    ∀p∈dsN, Ft1,Ft2∈TFormula,n2∈ℕ :
    n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,sf,ef) done(true) ∧ Ft2 →*(n2,p,sf,ef) done(bf) ⇒
    next(TCS(Ft1,Ft2)) →*(max(n1,n2),p,sf,ef) done(bf).

We need to prove ∀n1∈ℕ: Φ(n1). We use induction. Prove:

[TCS2.a] : Φ(1)
[TCS2.b] ∀n1∈ℕ: Φ(n1) ⇒ Φ(n1+1).

Proof of [TCS2.a]

We need to prove

 ∀n2,p∈dsN, Ft1,Ft2∈TFormula :

1>0 ∧ n2>0 ∧ Ft1 →*(1,p,sf,ef) done(true) ∧ Ft2 →*(n2,p,sf,ef) done(bf) ⇒
next(TCS(Ft1,Ft2)) →*(max(1,n2),p,sf,ef) done(bf).

We take n2,pf,Ft1f,Ft2f arbitrary but fixed. Assume

(TCS1.a.1) n2>0
(TCS1.a.2) Ft1f →*(1,pf,sf,ef) done(true)
(TCS1.a.3) Ft2f →*(n2,pf,sf,ef) done(bf)

and prove

[TCS1.a.4] next(TCS(Ft1f,Ft2f)) →*(max(1,n2),pf,sf,ef) done(bf).

From (TCS1.a.2), by the definition of →*, we have for some Ft'

(TCS1.a.5) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'
(TCS1.a.6) Ft' →*(0,pf+1,sf,ef) done(true)

where

(TCS1.a.7) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCS1.a.6), by the definition pf →*, we know

(TCS1.a.8) Ft'=done(true).

From (TCS1.a.5) and (TCS1.a.8) we have

(TCS1.a.9) Ft1f →(pf,sf↓pf,sf(pf),c) done(true).

From (TCS1.a.3), by the definition of →*, we have for some Ft''

(TCS1.a.10) Ft2f →(pf,sf↓pf,sf(pf),c) Ft''
(TCS1.a.11) Ft''→* (n2-1,pf+1,sf,ef) done(bf),

where c is defined as in (TCS1.a.7).

From (TCS1.a.9) and (TCS1.a.10), by the definition of → for TCS, we have

(TCS1.a.13) next(TCS(Ft1f,Ft2f)) → (pf,sf↓pf,sf(pf),c) Ft''.

From (TCS1.a.13), (TCS1.a.7), and (TCS1.a.11), by the definition of →*, we have

(TCS1.a.14) next(TCS(Ft1f,Ft2f)) → (n2,pf,sf,ef) done(bf).

From (TCS1.a.1), we have n2=max(1,n2). Therefore, (TCS1.a.14) proves [TCS1.a.4]

This finishes the proof of [TCS2.a].

Proof of [TCS2.b]

We take n1 arbitrary but fixed. Assume Φ(n1), i.e.,

```
(TCS2.b.1) ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
      n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,sf,ef) done(true) ∧
         Ft2 →*(n2,p,sf,ef) done(bf)
      ⇒
         next(TCS(Ft1,Ft2)) →*(max(n1,n2),p,sf,ef) done(bf).
```

and prove

```
[TCS2.b.2] ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
      n1+1>0 ∧ n2>0 ∧ Ft1 →*(n1+1,p,sf,ef) done(true) ∧
         Ft2 →*(n2,p,sf,ef) done(bf)
      ⇒
         next(TCS(Ft1,Ft2)) →*(max(n1+1,n2),p,sf,ef) done(bf).
```

To prove [TCS2.b.2], we take n2, pf, Ft1f, Ft2f arbitrary but fixed. Assume

```
(TCS2.b.3) n1+1>0
(TCS2.b.4) n2>0
(TCS2.b.5) Ft1f →*(n1+1,pf,sf,ef) done(true)
(TCS2.b.6) Ft2f →*(n2,pf,sf,ef) done(bf)
```

and prove

```
[TCS2.b.7] next(TCS(Ft1f,Ft2f)) →*(max(n1+1,n2),pf,sf,ef) done(bf).
```

From (TCS2.b.5), by the definition of →*, we have for some Ft'

```
(TCS2.b.8) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'
(TCS2.b.9) Ft' →*(n1,pf+1,sf,ef) done(true)
```

where

```
(TCS2.b.10) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).
```

From (TCS2.b.6), by the definition of →*, we have for some Ft''

```
(TCS2.b.11) Ft2f →(pf,sf↓pf,sf(pf),c) Ft''
(TCS2.b.12) Ft'' →*(n2-1,pf+1,sf,ef) done(bf),
```

where c is defined as in (TCS2.b.10).

Case n1=0

In this case we have Ft'=done(true) and from (TCS2.b.8) we get

```
(TCS2.b.13) Ft1f →(pf, sf↓pf,sf(pf),c) done(true).
```

From (TCS2.b.13) and (TCS2.b.11), by the definition of → for TCS, we have

```
(TCS2.b.14) next(TCS(Ft1f,Ft2f)) →(pf,sf↓pf,sf(pf),c) Ft''.
```

From (TCS2.b.4), (TCS2.b.10), (TCS2.b.14), (TCS2.b.12) by the definition of →*, we get

(TCS2.b.15) next(TCS(Ft1f,Ft2f)) →*(n2,pf,sf,ef) done(bf).

By (TCS2.b.4) and n1=0, we have n2=max(1,n2)=max(n1+1,n2).
Hence, (TCS2.b.16) proves [TCS2.b.7].

Case n1>0, n2-1>0

In this case Ft'=next(f') for some f'∈TFormulaCore.
Therefore, from (TCS3.b.8), by the definition of → for TCS we have

(TCS2.b.16) next(TCS(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) next(TCS(Ft',Ft2f)).

Since n2-1>0 and, hence, n2>0, from (TCS2.b.6) by the Shifting Lemma 7 we get

(TCS2.b.17) Ft2f →*(n2-1,pf+1,sf,ef) done(bf)

From n1>0, n2-1>0, (TCS2.b.9), (TCS2.b.17), by the induction hypothesis
(TCS2.b.1) we get

(TCS2.b.18) next(TCS(Ft',Ft2f)) →*(max(n1,n2-1),pf+1,sf,ef) done(bf)

From max(n1,n2-1)+1>0, (TCS2.b.10), (TCS2.b.16), (TCS2.b.18) we get

(TCS2.b.18) next(TCS(Ft1f,Ft2f)) →*(max(n1,n2-1)+1,pf,sf,ef) done(bf)

Since max(n1,n2-1)+1=max(n1+1,n2), (TCS2.b.18) proves [TCS2.b.7]

Case 2. n1>0, n2-1=0

In this case from (TCS2.b.12) we have Ft''=done(bf), which from (TCS2.b.12) gives

(TCS2.b.19) Ft2f →(pf,sf↓pf,sf(pf),c) done(bf).

From (TCS2.b.5), by Lemma 2, we have

(TCS2.b.23) Ft1f →l*(n1+1,pf,sf,ef) done(true).

From (TCS2.b.23), by the definition of →l*, we obtain for some Ft0

(TCS2.b.24) Ft1f →l*(n1,pf,sf,ef) Ft0
(TCS2.b.25) Ft0 →(pf+n1,s↓(pf+n1),s(pf+n1),c) done(true),

where c is defined as in (TCS2.b.10).

From (TCS2.b.19), by the Lemma 6, we have

(TCS2.b.26) Ft2f →(pf+n1,sf↓(pf+n1),sf(pf+n1),c) done(bf).

From (TCS2.b.25) and (TCS2.b.26), by the definition of → for TCS, we get

```
(TCS2.b.27) next(TCS(Ft0,Ft2f)) → (pf+n1,sf↓(pf+n1),sf(pf+n1),c) done(bf).
```

From (TCS2.b.24), by Lemma 2 we have

```
(TCS2.b.28) Ft1f →*(n1,pf,sf,ef) Ft0.
```

Moreover, (TCS2.b.23) implies that Ft1f is not a 'done' formula. Also, from (TCS2.b.25) since pf+n1>0 due to n1>0, we have that Ft0 is a 'next' formula.
Hence, there exists f0∈TFormulaCore such that

```
(TCS2.b.29) Ft0=next(f0)
```

and from (TCS2.b.28) we have

```
(TCS2.b.30) Ft1f →*(n1,pf,sf,ef) next(f0).
```

Now we would like to use the following proposition, which will be proved below:

```
(Prop) ∀Ft1,Ft2∈TFormula, n∈ℕ, f∈TFormulaCore, p∈ℕ, s∈Stream, e∈Environment:
       n>0 ⇒
         Ft1→*(n,p,s,e) next(f) ⇒
         next(TCS(Ft1,Ft2)) →*(n,p,s,e) next(TCS(next(f),Ft2))
```

Using (Prop) under the assumptions n1>0 and (TCS2.b.30), we obtain

```
(TCS2.b.31) next(TCS(Ft1f,Ft2f)) →*(n1,pf,sf,ef) next(TCS(next(f0),Ft2f))
```

which, by (TCS2.b.29) and Lemma 2 is

```
(TCS2.b.32) next(TCS(Ft1f,Ft2f)) →l*(n1,pf,sf,ef) next(TCS(Ft0,Ft2f))
```

From n1+1>0, (TCS2.b.10), (TCS2.b.32), (TCS2.b.27), by the definition of →l* we get

```
(TCS2.b.33) next(TCS(Ft1f,Ft2f)) →l*(n1+1,pf,sf,ef) done(bf)
```

Since n2=1, we have n1+1=max(n1+1,1)=max(n1+1,n2). Therefore, from (TCS2.b.33) by Lemma 2 we obtain [TCS2.b.7]

This finishes the proof of [TCS2.b].

This finishes the proof of [TCS2].

This finishes the proof of the Statement 2 of Lemma 4.

```
--------------
Proof of (Prop)
--------------
Parametrization:

Θ(n) :⇔
  ∀Ft1,Ft2∈TFormula, f∈TFormulaCore, p∈ℕ, s∈Stream, e∈Environment:
       n>0 ⇒
         Ft1→*(n,p,s,e) next(f) ⇒
         next(TCS(Ft1,Ft2)) →*(n,p,s,e) next(TCS(next(f),Ft2))
```

We need to prove ∀n∈ℕ: Θ(n). Induction:

[Prop.a] Θ(1)
[Prop.b] ∀n∈ℕ: Θ(n)⇒Θ(n+1)

Proof of [Prop.a]

We take Ft1f, Ft2f, f0, pf, sf, ef arbitrary but fixed. Assume

(p1) Ft1f→*(1,pf,sf,ef) next(f0)

and prove

[p2] next(TCS(Ft1f,Ft2f)) →*(1,pf,sf,ef) next(TCS(next(f0),Ft2f)).

From (p1), by the definition of →* there exists Ft'∈TFormula such that

(p3) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'
(p4) Ft' →*(0,pf+1,sf,ef) next(f0)

where

(p5) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (p4), we have Ft'=next(f0) and, hence, from (p3) we get

(p6) Ft1f →(pf,sf↓pf,sf(pf),c) next(f0).

From (p6), by the definition of → for TCS, we have

(p7) next(TCS(Ft1f,Ft2f)) →(pf,sf↓pf,sf(pf),c) next(TCS(next(f0),Ft2f)).

On the other hand, we have by the dfinition of →*:

(p8) next(TCS(next(f0),Ft2f)) →*(0,pf+1,sf,ef) next(TCS(next(f0),Ft2f)).

From (p7), (p5), (p8), by the definition of →* we get [p2].

Proof of [Prop.b]

We take n arbitraty but fixed, assume

```
(p9)   ∀Ft1,Ft2∈TFormula, f∈TFormulaCore, p∈ℕ, s∈Stream, e∈Environment:
       n>0 ⇒
         Ft1→*(n,p,s,e) next(f) ⇒
         next(TCS(Ft1,Ft2)) →*(n,p,s,e) next(TCS(next(f),Ft2))
```

and prove

```
[p10]  ∀Ft1,Ft2∈TFormula, f∈TFormulaCore, p∈ℕ, s∈Stream, e∈Environment:
       n+1>0 ⇒
         Ft1→*(n+1,p,s,e) next(f) ⇒
         next(TCS(Ft1,Ft2)) →*(n+1,p,s,e) next(TCS(next(f),Ft2)).
```

To prove (p10), we take Ft1f,Ft2f,f0,pf,sf,ef arbitrary but fixed, assume

```
(p11) Ft1f→*(n+1,pf,sf,ef) next(f0)
```

and prove

```
[p12] next(TCS(Ft1f,Ft2f)) →*(n+1,pf,sf,ef) next(TCS(next(f0),Ft2f)).
```

Case n>0

From (p11), by the definition of →*, we obtain for some Ft'∈TFormula

```
(p13) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'
(p14) Ft' →*(n,pf+1,sf,ef) next(f0)
```

where

```
(p15) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).
```

Since n>0, from (p14) and the induction hypothesis (p9) we obtain

```
(p16) next(TCS(Ft',Ft2f)) →*(n,pf+1,sf,ef) next(TCS(next(f0),Ft2f)).
```

Morover, Ft' is a 'next' formula. Therefore, from (p13), by the definition of → for TCS we have

```
(p17) next(TCS(Ftf1,Ft2f)) →(pf,sf↓pf,sf(pf),c) next(TCS(Ft',Ft2f)).
```

From (p16), (p15), (p17), since n+1>9, by the definition of →* we get [p12].

Case n=0

In this [p12] can be proved as it has been done in the base case [Prop.a]

This finishes the proof of [Prop.b] and, hence of (Prop).

Statement 3. TCP Formulas.

```
 ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ:
  n1>0 ∧ Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
  next(TCP(Ft1,Ft2)) →*(n1,p,s,e) done(false)
  ∧
  n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(false)⇒
  next(TCP(Ft1,Ft2)) →*(min(n1,n2),p,s,e) done(false)
  ∧
  n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
  next(TCP(Ft1,Ft2)) →*(max(n1,n2),p,s,e) done(true)
  ∧
  n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(false) ⇒
  next(TCP(Ft1,Ft2)) →*(n2,p,s,e) done(false)
```

```
Proof
--------------------------------------

We split the statement in four:

[TCP1] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
       next(TCP(Ft1,Ft2)) →*(n1,p,s,e) done(false)

[TCP2] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(false) ∧
       Ft2 →*(n2,p,s,e) done(false)
       ⇒
       next(TCP(Ft1,Ft2)) →*(min(n1,n2),p,s,e) done(false)

[TCP3] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
       next(TCP(Ft1,Ft2)) →*(max(n1,n2),p,s,e) done(true).

[TCP4] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(false) ⇒
       next(TCP(Ft1,Ft2)) →*(n2,p,s,e) done(false).

==========================================


Proof of [TCP1]
---------------

We take sf,ef arbitrary but fixed and define

Φ(n) :⇔
   ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
       next(TCP(Ft1,Ft2)) →*(n1,p,s,e) done(false)

We prove ∀n1∈ℕ: Φ(n1) by induction over n1. For n1=0 the formula is trivially true.

We start the induction from 1. Prove:

[TCP1.a] Φ(1) and
[TCP1.b] ∀n1∈ℕ: Φ(n1) ⇒ Φ(n1+1)


Proof of [TCP1.a]
------------
We take pf,Ft1f,Ft2f,n2 arbitrary but fixed. 1>0 is satisfied. Assume

(TCP1.1) n2>0
(TCP1.2) Ft1f →*(1,pf,sf,ef) done(false).
(TCP1.3) Ft2f →*(n2,p,s,e) done(true).

We want to prove
```

```
[TCP1.4] next(TCP(Ft1f,Ft2f)) →*(1,pf,sf,ef) done(false).
```

From (TCP1.2), by the definition of →* without history, there exists Ft∈TFormula such that

```
(TCP1.5) Ft1f →(p,sf↓pf,sf(pf),c) Ft and
(TCP1.6) Ft →*(0,pf+1,sf,ef) done(false)
```

where

```
(TCP1.7) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).
```

From (TCP1.6), by the definition of →* without history, we get

```
(TCP1.8’) Ft=done(false).
```

which from (TCP1.5) gives

```
(TCP1.9’) Ft1f →(p,sf↓pf,sf(pf),c) done(false) and
```

From (TCP1.9’) and (TCP1.3), by the definition of → for TCP, we get

```
(TCP1.10’) next(TCP(Ft1f,Ft2f)) →(p,sf↓pf,sf(pf),c) done(false).
```

From (TCP1.10’, TCP1.6, TCP1.8’, TCP1.7), by the definition of →* without history, we get [TCP1.4].

Proof of [TCP1.b]

We take n1 arbitrary but fixed, assume

```
(TCP1.8) ∀p∈ℕ, Ft1,Ft2∈TFormula, n2∈ℕ :
           n1>0 ∧ n2>0 ∧
           Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
             next(TCP(Ft1,Ft2)) →*(n1,p,s,e) done(false)
```

and prove

```
[TCP1.9] ∀p∈ℕ, Ft1,Ft2∈TFormula, n2∈ℕ :
           n1+1>0 ∧ n2>0 ∧
           Ft1 →*(n1+1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
             next(TCP(Ft1,Ft2)) →*(n1+1,p,s,e) done(false)
```

To prove [TCP1.9], we take pf,Ft1f,Ft2f,n2 arbitrary but fixed, assume

```
(TCP1.10) n+1>0
(TCP1.11) n2>0
(TCP1.12) Ft1f →*(n1+1,pf,sf,ef) done(false)
(TCP1.13) Ft2f →*(n2,pf,sf,ef) done(true)
```

and prove

```
[TCP1.14] next(TCP(Ft1f,Ft2f)) →*(n1+1,pf,sf,ef) done(false).
```

```
From (TCP1.12), by (TCP1.10) and the definition of →* without history,
there exists Ft'∈TFormula such that

(TCP1.15) Ft1f →(pf,sf↓pf, sf(pf),c) Ft'
(TCP1.16) Ft' →*(n1,pf+1,sf,ef) done(false)

where

(TCP1.17) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP1.13), by (TCP1.11) and the definition of →* without history,
there exists Ft''∈TFormula such that

(TCP1.18) Ft2f →(pf,sf↓pf, sf(pf),c) Ft''
(TCP1.19) Ft'' →*(n2-1,pf+1,sf,ef) done(true)

where c is defined as in (TCP1.17).


Case n1>0, n2-1>0
-----------------
In this case Ft'=next(f'), Ft''=next(f'') for some f',f''∈TFormulaCore.
Therefore, from (TCP1.15,TCP1.18), by the definition of → for TCP we have

(TCP1.20) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) next(TCP(Ft',Ft'')).

From n1>0, n2-1>0, (TCP1.16,TCP1.19), by the induction hypothesis (TCP1.8)
we have

(TCP1.21) next(TCP(Ft',Ft'')) →*(n1,pf+1,sf,ef) done(false).

From n1+1>0, (TCP1.17), (TCP1.20), (TCP1.21), by the definition of →* we have

(TCP1.22) next(TCP(Ftf1,Ftf2))  →*(n1+1,pf,sf,ef) done(false)

which is [TCP1.14]


Case n1>0, n2-1=0
-----------------
In this case Ft'=next(f') for some f'∈TFormulaCore and, from (TCP1.18)

(TCP1.23) Ft2f →(pf,sf↓pf, sf(pf),c) done(true).

Therefore, from (TCP1.15,TCP1.23), by the definition of → for TCP we have

(TCP1.24) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) Ft'

From n1+1>0, (TCP1.17), (TCP1.24), (TCP1.16), by the definition of →*
we get [TCP1.14].


Case n1=0
----------------
```

In this case Ft''=next(f'') for some f''∈TFormulaCore and, from (TCP1.15)

(TCP1.25) Ft1f →(pf,sf↓pf, sf(pf),c) done(false).

From (TCP1.25) by the definition of → for TCP we have

(TCP1.26) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) done(false).

From n1+1>0, (TCP1.17), (TCP1.26), (TCP1.16), by the definition of →*
we get [TCP1.14].

This finishes the proof of (b) and, therefore, the proof of [TCP1].

==

Proof of [TCP2]

Recall

```
[TCP2] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,s,e) done(false) ∧
       Ft2 →*(n2,p,s,e) done(false)
       ⇒
       next(TCP(Ft1,Ft2)) →*(min(n1,n2),p,s,e) done(false)
```

Proof

We take sf,ef arbitrary but fixed and define

```
Φ(n) :⇔
   ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
       n1>0 ∧ n2>0 ∧
       Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(false) ⇒
          next(TCP(Ft1,Ft2)) →*(min(n1,n2),p,s,e) done(false)
```

We prove ∀n1∈ℕ: Φ(n1) by induction over n1. For n1=0 the formula is
trivially true.

We start the induction from 1. Prove:

[TCP2.a] Φ(1) and
[TCP2.b] ∀n1∈ℕ: Φ(n1) ⇒ Φ(n1+1)

Proof of [TCP2.a]

We take pf,Ft1f,Ft2f,n2 arbitrary but fixed. 1>0 is satisfied. Assume

(TCP2.1) n2>0
(TCP2.2) Ft1f →*(1,pf,sf,ef) done(false).
(TCP2.3) Ft2f →*(n2,p,s,e) done(false).

We want to prove

```
[TCP2.4] next(TCP(Ft1f,Ft2f)) →*(min(1,n2),pf,sf,ef) done(false).

From (TCP2.2), by the definition of →* without history, there exists
Ft∈TFormula such that

(TCP2.5) Ft1f →(p,sf↓pf,sf(pf),c) Ft and
(TCP2.6) Ft →*(0,pf+1,sf,ef) done(false)

where

(TCP2.7) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP2.6), by the definition of →* without history, we get

(TCP2.8) Ft=done(false).

which from (TCP2.5) gives

(TCP2.9) Ft1f →(p,sf↓pf,sf(pf),c) done(false).

From (TCP2.9) and (TCP2.3), by the definition of → for TCP, we get

(TCP2.10) next(TCP(Ft1f,Ft2f)) →(p,sf↓pf,sf(pf),c) done(false).

From (TCP2.10, TCP2.6, TCP2.8, TCP2.7), by the definition of →*
without history, we get next(TCP(Ft1f,Ft2f)) →*(1,pf,sf,ef) done(false),
but since by (TCP2.1) we have 1=min(1,n2), we actually proved [TCP2.4].

Proof of [TCP2.b]
-----------------
We take n1 arbitrary but fixed, assume

(TCP2.8) ∀p∈ℕ, Ft1,Ft2∈TFormula, n2∈ℕ :
           n1>0 ∧ n2>0 ∧
           Ft1 →*(n1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(false) ⇒
              next(TCP(Ft1,Ft2)) →*(min(n1,n2),p,s,e) done(false)
and prove

[TCP2.9] ∀p∈ℕ, Ft1,Ft2∈TFormula, n2∈ℕ :
           n1+1>0 ∧ n2>0 ∧
           Ft1 →*(n1+1,p,s,e) done(false) ∧ Ft2 →*(n2,p,s,e) done(false) ⇒
               next(TCP(Ft1,Ft2)) →*(min(n1+1,n2),p,s,e) done(false).

To prove [TCP2.9], we take pf,Ft1f,Ft2f,n2 arbitrary but fixed, assume

(TCP2.10) n+1>0
(TCP2.11) n2>0
(TCP2.12) Ft1f →*(n1+1,pf,sf,ef) done(false)
(TCP2.13) Ft2f →*(n2,pf,sf,ef) done(false)

and prove

[TCP2.14] next(TCP(Ft1f,Ft2f)) →*(min(n1+1,n2),pf,sf,ef) done(false).
```

From (TCP2.12), by (TCP2.10) and the definition of →* without history,
there exists Ft'∈TFormula such that

(TCP2.15) Ft1f →(pf,sf↓pf, sf(pf),c) Ft'
(TCP2.16) Ft' →*(n1,pf+1,sf,ef) done(false)

where

(TCP2.17) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP2.13), by (TCP2.11) and the definition of →* without history,
there exists Ft''∈TFormula such that

(TCP2.18) Ft2f →(pf,sf↓pf, sf(pf),c) Ft''
(TCP2.19) Ft'' →*(n2-1,pf+1,sf,ef) done(false)

where c is defined as in (TCP2.17).

Case n1>0, n2-1>0

In this case Ft'=next(f'), Ft''=next(f'') for some f',f''∈TFormulaCore.
Therefore, from (TCP2.15,TCP2.18), by the definition of → for TCP we have

(TCP2.20) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) next(TCP(Ft',Ft'')).

From n1>0, n2-1>0, (TCP2.16,TCP2.19), by the induction hypothesis (TCP2.8)
we have

(TCP2.21) next(TCP(Ft',Ft'')) →*(min(n1,n2-1),pf+1,sf,ef) done(false).

From n1+1>0, (TCP2.17), (TCP2.20), (TCP2.21), by the definition of →* we have

(TCP2.22) next(TCP(Ftf1,Ftf2)) →*(min(n1,n2-1)+1,pf,sf,ef) done(false)

which is [TCP2.14]

Case n1>0, n2-1=0

In this case Ft'=next(f') for some f'∈TFormulaCore and, from (TCP2.18) we have

(TCP2.23) Ft2f →(pf,sf↓pf, sf(pf),c) done(false).

Therefore, from (TCP2.15,TCP2.23), by the definition of → for TCP we have

(TCP2.24) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) done(false)

From 1>0, (TCP2.17), (TCP2.24), (TCP2.19), by the definition of →* we get

(TCP2.25) next(TCP(Ftf1,Ftf2)) →*(1,pf,sf,ef) done(false)

But by n1>0 and n2=1 we have 1=min(n1+1,n2). Hence, (TCP2.25) proves [TCP2.14].

Case n1=0

In this case Ft''=next(f'') for some f''∈TFormulaCore and, from (TCP2.15) we have

```
(TCP2.26) Ft1f →(pf,sf↓pf, sf(pf),c) done(false).
```

From (TCP2.26) by the definition of → for TCP we have

```
(TCP2.27) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) done(false).
```

From 1>0, (TCP2.17), (TCP2.27), (TCP2.16), by the definition of →* we get

```
(TCP2.28) next(TCP(Ftf1,Ftf2))  →*(1,pf,sf,ef) done(false).
```

But by n1=0 and n2>0 we have 1=min(n1+1,n2). Hence, (TCP2.28) proves [TCP2.14].

This finishes the proof of (b) and, therefore, the proof of [TCP2].

===

Proof of [TCP3]

```
[TCP3] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ, b∈Bool :
         n1>0 ∧ n2>0 ∧
         Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(true) ⇒
           next(TCP(Ft1,Ft2)) →*(max(n1,n2),p,s,e) done(true).
```

Proof

We take sf,ef arbitrary but fixed and define

```
Φ(n1) :⇔
    ∀p∈dsN, Ft1,Ft2∈TFormula, n2∈ℕ :
      n1>0 ∧ n2>0 ∧
      Ft1 →*(n1,p,sf,ef) done(true) ∧ Ft2 →*(n2,p,sf,ef) done(true) ⇒
        next(TCP(Ft1,Ft2)) →*(max(n1,n2),p,sf,ef) done(true).
```

We need to prove ∀n1∈ℕ: Φ(n1). We use induction. Prove:

```
[TCP3.a] ∀n2∈ℕ: Φ(1)
[TCP3.b] ∀n1∈ℕ: Φ(n1) ⇒ Φ(n1+1).
```

Proof of [TCP3.a]

We need to prove

```
 ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
    1>0 ∧ n2>0 ∧
    Ft1 →*(1,p,sf,ef) done(true) ∧ Ft2 →*(n2,p,sf,ef) done(true) ⇒
      next(TCP(Ft1,Ft2)) →*(max(1,n2),p,sf,ef) done(true).
```

We take n2,pf,Ft1f,Ft2f arbitrary but fixed. Assume

(TCP3.a.1) n2>0

(TCP3.a.2) Ft1f →*(1,pf,sf,ef) done(true)

(TCP3.a.3) Ft2f →*(n2,pf,sf,ef) done(true)

and prove

[TCP3.a.4] next(TCP(Ft1f,Ft2f)) →*(max(1,n2),pf,sf,ef) done(true).

From (TCP3.a.2), by the definition of →*, we have for some Ft'

(TCP3.a.5) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'

(TCP3.a.6) Ft' →*(0,pf+1,sf,ef) done(true)

where

(TCP3.a.7) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP3.a.6), by the definition pf →*, we know

(TCP3.a.8) Ft'=done(true).

From (TCP3.a.5) and (TCP3.a.8) we have

(TCP3.a.9) Ft1f →(pf,sf↓pf,sf(pf),c) done(true).

From (TCP3.a.3), by the definition of →*, we have for some Ft''

(TCP3.a.10) Ft2f →(pf,sf↓pf,sf(pf),c) Ft''

(TCP3.a.11) Ft''→* (n2-1,pf+1,sf,ef) done(true),

where c is defined as in (TCP3.a.7).

From (TCP3.a.9) and (TCP3.a.10), by the definition of → for TCP, we have

(TCP3.a.13) next(TCP(Ft1f,Ft2f)) → (pf,sf↓pf,sf(pf),c) Ft''.

From (TCP3.a.13), (TCP3.a.7), and (TCP3.a.11), by the definition of →*, we have

(TCP3.a.14) next(TCP(Ft1f,Ft2f)) →* (n2,pf,sf,ef) done(true).

From (TCP3.a.1), we have n2=max(1,n2). Therefore, (TCP3.a.14) proves [TCP3.a.4]

This finishes the proof of [TCP3.a].

Proof of [TCP3.b]

We take n1 arbitrary but fixed. Assume Φ(n1), i.e.,

```
(TCP3.b.1) ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
      n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,sf,ef) done(true) ∧
         Ft2 →*(n2,p,sf,ef) done(true)
```

⇒
next(TCP(Ft1,Ft2)) →*(max(n1,n2),p,sf,ef) done(true).

and prove

[TCP3.b.2] ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
n1+1>0 ∧ n2>0 ∧ Ft1 →*(n1+1,p,sf,ef) done(true) ∧
Ft2 →*(n2,p,sf,ef) done(true)
⇒
next(TCP(Ft1,Ft2)) →*(max(n1+1,n2),p,sf,ef) done(true).

To prove [TCP3.b.2], we take n2, pf, Ft1f, Ft2f arbitrary but fixed. Assume

(TCP3.b.3) n1+1>0
(TCP3.b.4) n2>0
(TCP3.b.5) Ft1f →*(n1+1,pf,sf,ef) done(true)
(TCP3.b.6) Ft2f →*(n2,pf,sf,ef) done(true)

and prove

[TCP3.b.7] next(TCP(Ft1f,Ft2f)) →*(max(n1+1,n2),pf,sf,ef) done(true).

From (TCP3.b.5), by the definition of →*, we have for some Ft'

(TCP3.b.8) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'
(TCP3.b.9) Ft' →*(n1,pf+1,sf,ef) done(true)

where

(TCP3.b.10) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP3.b.6), by the definition of →*, we have for some Ft''

(TCP3.b.11) Ft2f →(pf,sf↓pf,sf(pf),c) Ft''
(TCP3.b.12) Ft'' →*(n2-1,pf+1,sf,ef) done(true)

where c is defined as in (TCP3.b.10).

Case 1. n1=0

In this case we have Ft'=done(true) and from (TCP3.b.8) we get

(TCP3.b.13) Ft1f →(pf, sf↓pf,sf(pf),c) done(true).

From (TCP3.b.13) and (TCP3.b.11), by the definition of → for TCP, we have

(TCP3.b.14) next(TCP(Ft1f,Ft2f)) →(pf,sf↓pf,sf(pf),c) Ft''.

From (TCP3.b.4), (TCP3.b.10), (TCP3.b.14), (TCP3.b.12) by the definition of →*
we get

(TCP3.b.15) next(TCP(Ft1f,Ft2f)) →*(n2,pf,sf,ef) done(true).

By (TCP3.b.4) and n1=0, we have n2=max(1,n2)=max(n1+1,n2). Hence, (TCP3.b.15) proves [TCP3.b.7].

Case n1>0, n2-1>0

In this case Ft'=next(f'), Ft''=next(f'') for some f',f''∈TFormulaCore.
Therefore, from (TCP3.b.8,TCP3.b.11), by the definition of → for TCP we have

(TCP3.b.16) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) next(TCP(Ft',Ft'')).

From n1>0, n2-1>0, (b9,b12), by the induction hypothesis (TCP3.b.1) we have

(TCP3.b.17) next(TCP(Ft',Ft'')) →*(max(n1,n2-1),pf+1,sf,ef) done(true).

From n1+1>0, (TCP3.b.10), (TCP3.b.16), (TCP3.b.17), by the definition of →* we have

(TCP3.b.18) next(TCP(Ftf1,Ftf2)) →*(max(n1,n2-1)+1,pf,sf,ef) done(true)

which is [TCP3.b.7]

Case n1>0, n2-1=0

In this case Ft'=next(f') for some f'∈TFormulaCore. From (TCP3.b.11) we have

(TCP3.b.19) Ft2f →(pf,sf↓pf, sf(pf),c) done(true).

From (TCP3.b.8,TCP3.b.19), by the definition of → for TCP we have

(TCP3.b.20) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) Ft'

From n1+1>0, (TCP3.b.10), (TCP3.b.20), (TCP3.b.9), by the definition of →* we get

(TCP3.b.21) next(TCP(Ftf1,Ftf2)) →*(n1+1,pf,sf,ef) done(true)

But by n1>0 and n2=1 we have n1+1=max(n1+1,n2). Hence, from (TCP3.b.21) we get [TCP3.b.7].

This finishes the proof of [TCP3.b].

This finishes the proof of [TCP3].

==

Proof of [TCP4]

```
[TCP4] ∀p∈ℕ, s∈Stream, e∈Environment, Ft1,Ft2∈TFormula, n1,n2∈ℕ :
         n1>0 ∧ n2>0 ∧
         Ft1 →*(n1,p,s,e) done(true) ∧ Ft2 →*(n2,p,s,e) done(false) ⇒
```

```
          next(TCP(Ft1,Ft2)) →*(n2,p,s,e) done(false).

Proof
-----

We take sf,ef,bf arbitrary but fixed and define

Φ(n1) :⇔
    ∀p∈dsN, Ft1,Ft2∈TFormula, n2∈ℕ :
      n1>0 ∧ n2>0 ∧
      Ft1 →*(n1,p,sf,ef) done(true) ∧ Ft2 →*(n2,p,sf,ef) done(false) ⇒
        next(TCP(Ft1,Ft2)) →*(n2,p,sf,ef) done(false).

We need to prove ∀n1∈ℕ: Φ(n1). We use induction. Prove:

[TCP4.a] ∀n2∈ℕ: Φ(1)
[TCP4.b] ∀n1∈ℕ: Φ(n1) ⇒ Φ(n1+1).


Proof of [TCP4.a]
-----------------
We need to prove

 ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
    1>0 ∧ n2>0 ∧
    Ft1 →*(1,p,sf,ef) done(true) ∧ Ft2 →*(n2,p,sf,ef) done(false) ⇒
      next(TCP(Ft1,Ft2)) →*(n2,p,sf,ef) done(false).

We take n2,pf,Ft1f,Ft2f arbitrary but fixed. Assume

(TCP4.a.1) n2>0
(TCP4.a.2) Ft1f →*(1,pf,sf,ef) done(true)
(TCP4.a.3) Ft2f →*(n2,pf,sf,ef) done(false)

and prove

[TCP4.a.4] next(TCP(Ft1f,Ft2f)) →*(n2,pf,sf,ef) done(false).

From (TCP4.a.2), by the definition of →*, we have for some Ft'

(TCP4.a.5) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'
(TCP4.a.6) Ft' →*(0,pf+1,sf,ef) done(true)

where

(TCP4.a.7) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP4.a.6), by the definition pf →*, we know

(TCP4.a.8) Ft'=done(true).

From (TCP4.a.5) and (TCP4.a.8) we have

(TCP4.a.9) Ft1f →(pf,sf↓pf,sf(pf),c) done(true).
```

From (TCP4.a.3), by the definition of →*, we have for some Ft''

(TCP4.a.10) Ft2f →(pf,sf↓pf,sf(pf),c) Ft''
(TCP4.a.11) Ft''→* (n2-1,pf+1,sf,ef) done(false),

where c is defined as in (TCP4.a.7).

From (TCP4.a.9) and (TCP4.a.10), by the definition of → for TCP, we have

(TCP4.a.13) next(TCP(Ft1f,Ft2f)) → (pf,sf↓pf,sf(pf),c) Ft''.

From (TCP4.a.13), (TCP4.a.7), and (TCP4.a.11), by the definition of →*, we have

(TCP4.a.14) next(TCP(Ft1f,Ft2f)) →* (n2,pf,sf,ef) done(false).

(TCP4.a.14) is [TCP4.a.4].

This finishes the proof of [TCP4.a].

Proof of [TCP4.b]

We take n1 arbitrary but fixed. Assume Φ(n1), i.e.,

```
(TCP4.b.1) ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
      n1>0 ∧ n2>0 ∧ Ft1 →*(n1,p,sf,ef) done(true) ∧
         Ft2 →*(n2,p,sf,ef) done(false)
      ⇒
         next(TCP(Ft1,Ft2)) →*(n2,p,sf,ef) done(false).
```

and prove

```
[TCP4.b.2] ∀n2,p∈dsN, Ft1,Ft2∈TFormula :
      n1+1>0 ∧ n2>0 ∧ Ft1 →*(n1+1,p,sf,ef) done(true) ∧
         Ft2 →*(n2,p,sf,ef) done(bf)
      ⇒
         next(TCP(Ft1,Ft2)) →*(false,p,sf,ef) done(false).
```

To prove [TCP4.b.2], we take n2, pf, Ft1f, Ft2f arbitrary but fixed. Assume

(TCP4.b.3) n1+1>0
(TCP4.b.4) n2>0
(TCP4.b.5) Ft1f →*(n1+1,pf,sf,ef) done(true)
(TCP4.b.6) Ft2f →*(n2,pf,sf,ef) done(false)

and prove

[TCP4.b.7] next(TCP(Ft1f,Ft2f)) →*(n2,pf,sf,ef) done(false).

From (TCP4.b.5), by the definition of →*, we have for some Ft'

(TCP4.b.8) Ft1f →(pf,sf↓pf,sf(pf),c) Ft'

(TCP4.b.9) Ft' →*(n1,pf+1,sf,ef) done(true)

where

(TCP4.b.10) c=(ef, {(X,sf(ef(X)))| X∈dom(ef)}).

From (TCP4.b.6), by the definition of →*, we have for some Ft''

(TCP4.b.11) Ft2f →(pf,sf↓pf,sf(pf),c) Ft''
(TCP4.b.12) Ft'' →*(n2-1,pf+1,sf,ef) done(false)

where c is defined as in (TCP4.b.10).

Case 1. n1=0

In this case we have Ft'=done(true) and from (TCP4.b.8) we get

(TCP4.b.13) Ft1f →(pf, sf↓pf,sf(pf),c) done(true).

From (TCP4.b.13) and (TCP4.b.11), by the definition of → for TCP, we have

(TCP4.b.14) next(TCP(Ft1f,Ft2f)) →(pf,sf↓pf,sf(pf),c) Ft''.

From (TCP4.b.4), (TCP4.b.10), (TCP4.b.14), (TCP4.b.12) by the definition of →*, we get

(TCP4.b.15) next(TCP(Ft1f,Ft2f)) →*(n2,pf,sf,ef) done(false).

Hence, (TCP4.b.15) proves [TCP4.b.7].

Case n1>0, n2-1>0

In this case Ft'=next(f'), Ft''=next(f'') for some f',f''∈TFormulaCore.
Therefore, from (TCP4.b.8,TCP4.b.11), by the definition of → for TCP we have

(TCP4.b.16) next(TCP(Ftf1,Ftf2)) →(pf,sf↓pf, sf(pf),c) next(TCP(Ft',Ft'')).

From n1>0, n2-1>0, (b9,b12), by the induction hypothesis (TCP4.b.1) we have

(TCP4.b.17) next(TCP(Ft',Ft'')) →*(n2-1,pf+1,sf,ef) done(false).

From (TCP4.b.4), (TCP4.b.10), (TCP4.b.16), (TCP4.b.17), by the definition of →* we have

(TCP4.b.18) next(TCP(Ftf1,Ftf2)) →*(n2,pf,sf,ef) done(false)

which is [TCP4.b.7]

Case n1>0, n2-1=0

In this case Ft'=next(f') for some f'∈TFormulaCore. From (TCP4.b.11) we have

(TCP4.b.19) Ft2f →(pf,sf↓pf, sf(pf),c) done(false).

From (TCP4.b.8,TCP4.b.19), by the definition of $\rightarrow$ for TCP we have

(TCP4.b.23) next(TCP(Ftf1,Ftf2)) $\rightarrow$(pf,sf$\downarrow$pf, sf(pf),c) done(false).

From (TCP4.b.12), by n2-1=0 and bf=false we have

(TCP4.b.24) done(false) $\rightarrow$*(n2-1,pf+1,sf,ef) done(false)

From (TCP4.b.4), (TCP4.b.10), (TCP4.b.23), (TCP4.b.24) by the definition of $\rightarrow$*
we get

(TCP4.b.20) next(TCP(Ftf1,Ftf2)) $\rightarrow$*(n2,pf,sf,ef) done(false)

which is [TCP4.b.7]

This finishes the proof of [TCP4.b].

This finishes the proof of [TCP4].

This finishes the proof of the Statement 3 of Lemma 4.

A.7 Lemma 5: Soundness Lemma for Universal Formulas

```
  ∀F∈Formula, X∈Variable, B1,B2∈Bound:
  R(F) ⇒ R(forall X in B1..B2: F)

  where

  R(F) :⇔
   ∀re∈RangeEnv, e∈Environment, s∈Stream, d∈ℕ∞, h∈ℕ, p∈ℕ:
     ⊢ (re ⊢ F: (h,d)) ∧ d∈ℕ ∧ dom(e) = dom(re) ∧
       (∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p)  ⇒
       (∃b∈Bool ∃d'∈ℕ:
         d'≤d+1 ∧ ⊢ T(F) →*(d',p,s,e) done(b) )

PROOF:
-----

We take F,X,B1,B2 arbitrary but fixed, assume

(1) R(F)

and prove

[2] R(forall X in B1..B2: F).

We denote b1=T(B1), b2=T(B2), f=T(F).

From the definition of T and f, we know

(2) ∃fc∈TFormulaCore: f=next(fc)

We take ref∈RangeEnv, ef∈Environment, sf∈Stream, df∈ℕ∞, hf∈ℕ, pf∈ℕ
arbitrary but fixed. Assume

(3) ⊢ (ref ⊢ (forall X in B1..B2: F): (hf,df))
(4) df∈ℕ
(4') dom(ef) = dom(ref)
(5) ∀Y∈dom(ef): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf

and prove

[6] ∃b∈Bool ∃d'∈ℕ: d'≤df+1 ∧ ⊢ next(TA(X,b1,b2,f)) →*(d',pf,sf,ef) done(b).

We prove [6] by contradiction. Assume

(7) ∀b∈Bool ∀d'∈ℕ: d'≤df+1 ⇒ ¬ (⊢ next(TA(X,b1,b2,f)) →*(d',pf,sf,ef) done(b)).

Note that by the operational semantics,
   ¬(⊢ next(TA(X,b1,b2,f)) →*(d',pf,sf,ef) done(b))
is equivalent to
   ∃fc∈TFormulaCore: ⊢ next(TA(X,b1,b2,f)) →*(d',pf,sf,ef) next(fc).
Hence, (7) can be rewritten to
```

```
(8) ∀d'∈ℕ: (d'≤df+1 ⇒
       ∃fc∈TFormulaCore: ⊢ next(TA(X,b1,b2,f)) →*(d',pf,sf,ef) next(fc)).
```

We thus know for some fc∈TFormulaCore

```
(9) ⊢ next(TA(X,b1,b2,f)) →*(df+1,pf,sf,ef) next(fc)
```

From the invariant, (2) and (9), there exist c∈Context, p0,p1,p2∈ℕ such that

```
(10) c  = (ef,{(X,sf(ef(X))) | X ∈ dom(ef)})
(11) p0 = pf+df+1
(12) p1 = b1(c)
(13) p2 = b2(c)
```

and we have 2 cases:

```
CASE 1:
------
(20) df+1 ≥ 1
(21) p1 ≠ ∞
(22) p0 ≤ p1
(23) p1 ≤∞ p2
(23) fc = TA0(X,p1,p2,f)
```

From (3), by the analysis, we know for some l1,u1,l2,u2∈ℤ∞ and h',d'∈ℕ∞:

```
(24) ref ⊢ B1 : (l1, u1)
(25) ref ⊢ B2 : (l2, u2)
(26) ref[X↦(l1, u2)] ⊢ F : (h1, d1)
(27) hf = max∞(h1, ℕ∞(-i(l1)))
(28) df = max∞(d1, ℕ∞(u2))
```

From (4), (28), and the definition of max∞, we know

```
(29) d1∈ℕ
(30) (u2∈ℤ ∧ u2<0 ∧ df=d1) ∨
      (u2∈ℕ ∧ df=max(d1,u2))
```

From (29) and (30), we can conclude

```
(31) u2∈ℤ
(32) df=max(d1,u2)
```

Hence, from (32) we have

```
(33) df ≥ u2.
```

From (33) we have

```
(34) pf+df+1 ≥ pf+u2+1 > pf+u2.
```

On the other hand, from (25), (4'), (5), and (10), by Lemma 9 we get

```
(35) l2 +i pf ≤i b2(c) ≤i u2 +i pf.
```

From (11), (12), (22), and (35) we have

```
(36) pf+df+1 = p0 ≤ p1 = b1(c) ≤ b2(c) ≤i u2 +i pf.
```

From (31), (34) and (36) we get a contradiction:
```
pf+df+1 > pf+u2 and pf+df+1 ≤ pf+u2.
```

This proves CASE 1.

CASE 2:

There exist some fs,gs∈ℙ(TInstance) such that
```
(100) df+1 ≥ 1
(101) p1 ≠ ∞
(102) p1 ≤∞ p2
(103) p0 > p1
(104) gs ≠ ∅ ∨ pf+df+1 ≤∞ p2
(105) forallInstances(X,p,p0,p1,p2,f,sf,ef,gs)
(106) fc = TA1(X,p2,f,gs)
```

From (3) and the definition of the analysis, we know for some
l1,u1,l2,u2∈ℤ∞ and h',d'∈ℕ∞:

```
(111) ref ⊢ B1 : (l1, u1)
(112) ref ⊢ B2 : (l2, u2)
(113) ref[X↦(l1, u2)] ⊢ F : (h', d')
(114) hf = max∞(h', ℕ∞(-i(l1)))
(115) df = max∞(d', ℕ∞(u2))
```

From (4), (115), and the definition of max∞, we know

```
(116) d'∈ℕ
(117) (u2∈ℤ ∧ u2<0 ∧ df=d') ∨
      (u2∈ℕ ∧ df=max(d',u2))
```

From (116) and (117), we can conclude

```
(118) u2∈ℤ
(119) df=max(d',u2)
```

From (104), we have two subcases:

Subcase 2.1

```
(200) pf+df+1 ≤∞ p2
```

From (119), we know

```
(201) df ≥ u2
```

From Lemma 9 with (4'), (5), (10), (13), (112), (118) and the definition of b2,
we know

```
(202) p2 ≤ pf+u2
```

From (200) and (202), we have

```
(203) pf+df+1 ≤ pf+u2
```

and thus

```
(204) df+1 ≤ u2
```

which contradicts (201).

Subcase 2.2

```
(300) pf+df+1 >∞ p2
(301) gs ≠ ∅
```

From (301), (105) and the definition of "forallInstances", we know
for some t∈ℕ, g∈TFormula, c0∈Context, gc∈TFormulaCore:

```
(302) (t,g,c0)∈gs
(303) (∀t1∈ℕ,g1∈TFormula,c1∈Context:
        (t1,g1,c1)∈gs ∧ t=t1 ⇒ (t,g,c0)=(t1,g1,c1)
(304) g=next(gc)
(305) c0.1=ef[X ↦ t]
(306) c0.2={(Y,s(ef(Y))) | Y ∈ dom(ef) ∨ Y = X}
(307) p1 ≤ t
(308) t ≤ min∞(p0-1,p2)
(309) ⊢ f →*(p0-max(pf,t),max(pf,t),sf,c0.1) g
```

We define

```
(310) ref':=ref[X ↦ (l1, u2)]
(311) ef':=ef[X ↦ t]
```

From (311), we know

```
(312) dom(ef') = dom(ef)∪{X}
```

and claim

```
(313) ∀Y∈dom(ef'): ref'(Y).1+pf ≤i ef'(Y) ≤i ref'(Y).2+pf
```

Proof: take arbitrary Y∈dom(ef'). From (312), we have
two cases:

* case Y≠X: we have Y ∈ dom(ef) and by (4') ref'(Y)=ref(Y) and ef'(Y)=ef(Y);
  it thus suffices to show ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf
  which follows from (5).

* case Y=X: we have ref'(Y)=(l1,u2) and ef'(Y)=t; it thus suffices to
  show l1 +i pf ≤i t ≤i u2+pf. From (307) and (308) it suffices to show

  [1] l1 +i pf ≤i p1
  [2] min∞(p0-1,p2) ≤i u2 +i pf.

  From Lemma 9, (4’), (5), (10), (12), (111), and the definition of b1,
  we have l1 +i pf ≤i p1 and thus [1].
  From Lemma 9, (4’), (5), (10), (13), (112), and the definition of b2,
  we have p2 ≤i u2 +i pf and thus [2].

From (1), (113), (116), (305), (310), (311), (313) and the definitions of
R and f, we know that there exists some b∈Bool and d0∈ℕ such that

(314) d0 ≤ d’+1
(315) ⊢ f →*(d0,pf,sf,c0.1) done(b)

We proceed by case distinction.

Subcase 2.2.1
----
(400) t < pf

From (304), (309) and (400), we know

(401) ⊢ f →*(p0-pf,pf,sf,c0.1) next(gc)

Because the rule system is deterministic and there is no transition
starting with done(b), to derive a contradiction, it suffices
with (315) and (401) to show

[402] d0 ≤ p0-pf

which holds because

  d0 ≤(314) d’+1 ≤(119) df+1 = (pf+df+1)-pf =(11) p0-pf

Subcase 2.2.2
----
(500) t ≥ pf

From (304), (309) and (500), we know

(501) ⊢ f →*(p0-t,t,sf,c0.1) next(gc)

By a generalization of Lemma 7, we know from (2), (315) and (500)

(502) ⊢ f →*(max(1,d0-(t-pf)),t,sf,c0.1) done(b)

Because the rule system is deterministic and there is no transition
starting with done(b), to derive a contradiction, it suffices
with (501) and (502) to show

[503] max(1,d0-(t-pf)) ≤ p0-t

From (308), we know

(504) t $\leq$ p0-1

and thus

(505) 1 $\leq$ p0-t

From (505), to show [503] it suffices to show

[506] d0-(t-pf) $\leq$ p0-t

for which it suffices to show

[507] d0+pf $\leq$ p0

which holds because

  d0+pf $\leq$(314) d'+1+pf $\leq$(119) df+1+pf =(11) p0

QED.

A.8 Lemma 6: Monotonicity of Reduction to **done**

```
∀ Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context, b∈Bool :
   ∀ k ≥ p:
      Ft →(p,s↓p,s(p),c) done(b) ⇒ Ft →(k,s↓k,s(k),c) done(b)

PROOF
-----

We take pf,sf,bf,kf arbitrary but fixed, assume

(1) kf ≥ pf

and prove

(2) ∀ Ft∈TFormula ∀c∈Context:
    Ft →(pf,sf↓pf,s(pf),c) done(bf) ⇒
      Ft →(kf,sf↓kf,sf(kf),c) done(bf)

We prove (2) by structural induction over Ft:

C1. Ft=next(TV(X))
------------------
We take cf arbitrary but fixed, assume

(1.1) next(TV(X)) →(pf,sf↓pf,s(pf),cf) done(bf)

and prove

(1.2) next(TV(X)) →(kf,sf↓kf,sf(kf),cf) done(bf)

By definition of →, the value of bf depends only on cf, which is the same in
(1.1) and (1.2). Hence, (1.1) implies (1.2)

It proves C1.

C2. Ft=next(TN(f)) for some f∈TFormula
-----------------
We take cf arbitrary but fixed, assume

(2.1) next(TN(f)) →(pf,sf↓pf,s(pf),cf) done(bf)

and prove

(2.2) next(TN(f)) →(kf,sf↓kf,sf(kf),cf) done(bf)

From (2.1), by the definition of →, we have

(2.3) f →(pf,sf↓pf,s(pf),cf) done(b1)

where

(2.4) b1 = if bf = false true else false.
```

By the induction hypothesis, from (2.3) we get

(2.5) f →(kf,sf↓kf,s(kf),cf) done(b1).

From (2.5), by the definition of → and (2.4) we get (2.2).

It proves C2.

C3. Ft=next(TCS(f1,f2)) for some f1,f2∈TFormula

We take cf arbitrary but fixed, assume

(3.1) next(TCS(f1,f2)) →(pf,sf↓pf,s(pf),cf) done(bf)

and prove

(3.2) next(TCS(f1,f2)) →(kf,sf↓kf,sf(kf),cf) done(bf)

From (3.1) we have two alternatives:

(a) We have

(3.3) bf=false and

(3.4) f1 →(pf,sf↓pf,s(pf),cf) done(false).

By the induction hypothesis, from (3.4) we get

(3.5) f1 →(kf,sf↓kf,s(kf),cf) done(false).

From (3.5), by the definition of → we get (3.2).

(b) We have

(3.6) f1 →(pf,sf↓pf,s(pf),cf) done(true)
(3.7) f2 →(pf,sf↓pf,s(pf),cf) done(bf).

By the induction hypothesis, we get from (3.6) and (3.7) respectively

(3.8) f1 →(kf,sf↓kf,s(kf),cf) done(true)
(3.9) f2 →(kf,sf↓pf,s(kf),cf) done(bf).

From (3.8) and (3.9), by the definition of → we get (3.2).

It proves C3.

C4. Ft=next(TCP(f1,f2)) for some f1,f2∈TFormula

We take cf arbitrary but fixed, assume

(4.1) next(TCP(f1,f2)) $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) done(bf)

and prove

(4.2) next(TCP(f1,f2)) $\rightarrow$(kf,sf$\downarrow$kf,sf(kf),cf) done(bf)

From (4.1) we have three alternatives:

(a) We have

(4.3) bf=false
(4.4) f1 $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) next(f1') for some f1'$\in$TFormulaCore
(4.5) f2 $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) done(false).

From (4.4) and (4.5) we obtain by the induction hypothesis, respectively,

(4.6) f1 $\rightarrow$(kf,sf$\downarrow$kf,s(kf),cf) next(f1')
(4.7) f2 $\rightarrow$(kf,sf$\downarrow$kf,s(kf),cf) done(false).

From (4.6) and (4.7), by the definition of $\rightarrow$ and (4.3) we get (4.2).

(b) We have

(4.8) bf=false and

(4.9) f1 $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) done(false).

By the induction hypothesis, from (4.4) we get

(4.5) f1 $\rightarrow$(kf,sf$\downarrow$kf,s(kf),cf) done(false).

From (3.5), by the definition of $\rightarrow$ we get (4.2).

(c) We have

(4.6) f1 $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) done(true)
(4.8) f2 $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) done(bf).

By the induction hypothesis, we get from (3.6) and (3.7) respectively

(4.9) f1 $\rightarrow$(kf,sf$\downarrow$kf,s(kf),cf) done(true)
(4.10) f2 $\rightarrow$(kf,sf$\downarrow$pf,s(kf),cf) done(bf).

From (4.9) and (4.10), by the definition of $\rightarrow$ we get (4.2).

It proves C4.

C5. Ft=next(TA(X,b1,b2,f))

We take cf arbitrary but fixed, assume

(5.1) next(TA(X,b1,b2,f)) $\rightarrow$(pf,sf$\downarrow$pf,s(pf),cf) done(bf)

and prove

[5.2] next(TA(X,b1,b2,f)) $\rightarrow$(kf,sf$\downarrow$kf,sf(kf),cf) done(bf)

(a) bf=true.

From (5.1) we have

p1 = b1(cf)
p1 = ∞

which immediately imply [5.2].

(b) bf=false

To prove [5.2], we need to find p1*,p2* such that

[5.3] p1* = b1(cf)
[5.4] p2* = b2(cf)
[5.5] p1* $\neq$ ∞
[5.6] next(TA0(X,p1*,p2*,f)) $\rightarrow$(kf,sf$\downarrow$kf,sf(kf),cf) done(false)

From (5.1) we know

(5.7) p1 = b1(cf)
(5.8) p2 = b2(cf)
(5.9) p1 $\neq$ ∞
(5.10) next(TA0(X,p1,p2,f)) $\rightarrow$(pf,sf$\downarrow$pf,sf(pf),cf) done(false)

We take p1*=p1,p2*=p2. Then [5.3-5.5] follow from (5.7-5.9) and we need to prove

[5.11] next(TA0(X,p1,p2,f)) $\rightarrow$(kf,sf$\downarrow$kf,sf(kf),cf) done(false).

By Def.$\rightarrow$, to prove [5.11], we need to prove

[5.12] kf $\geq$ p1
[5.13] next(TA1(X,p2,f,fsk)) $\rightarrow$(kf,sf$\downarrow$kf,sf(kf),cf) done(false)

where

(5.14) fsk = {(p0,f,(cf.1[X$\mapsto$p0],cf.2[X$\mapsto$(sf$\downarrow$kf)(p0)])) | p1 $\leq$ p0 <∞ min∞(kf,p2+∞1)}

From (5.10), by the definition of $\rightarrow$, we know

(5.15) pf$\geq$p1
(5.16) next(TA1(X,p2,f,fsp)) $\rightarrow$(pf,sf$\downarrow$pf,sf(pf),cf) done(false)

where

(5.17) fsp = {(p0,f,(cf.1[X↦p0],cf.2[X↦(sf↓pf)(p0)])) | p1 ≤ p0 <∞ min∞(pf,p2+∞1)}

Then [5.12] follows from (1) and (5.15).

To prove [5.13], by Def.→ we need to prove

[5.18] ∃t∈ℕ,g∈TFormula,c∈Context: (t,g,c)∈fs0k ∧ ⊢ g →(kf,sf↓kf,sf(kf),c) done(false)

where

(5.19) fs0k = if kf >∞ p2 then fsk else fsk ∪ {(kf,f,(cf.1[X↦kf],cf.2[X↦sf(kf)]))}

From (5.16) we know that there exist tp∈ℕ,gp∈TFormula,cp∈Context such that

(5.20) (tp,gp,cp)∈fs0p
(5.21) gp →(pf,sf↓pf,sf(pf),cp) done(false)

where

(5.22) fs0p = if pf >∞ p2 then fsp else fsp ∪ {(pf,f,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}

Since by (1) kf≥pf, from (5.14) and (5.17) we have

(5.23) fsp ⊆ fsk.

Also, we have either

(5.25) (pf,f,(cf.1[X↦pf],cf.2[X↦sf(pf)])∈fsk (when kf>pf, since (sf↓pf)(kf)=sf(pf))

or

(5.26) (pf,f,(cf.1[X↦pf],cf.2[X↦sf(pf)])∈fs0k, (kf=pf).

From (5.25) and (5.26) we get

(5.27) (pf,f,(cf.1[X↦pf],cf.2[X↦sf(pf)])∈fs0k, when kf≥pf.

From (1), (5.23), (5.27), (5.19), (5.22) we get

(5.28) fs0p ⊆ fs0k.

Then from (5.20) we get

(5.29) (tp,gp,cp)∈fs0k.

From (5.21) and (2) we get

(5.30) gp →(kf,sf↓kf,sf(kf),cp) done(false)

From (5.29) and (5.30) we obtain [5.18].

It proves C5.

It finishes the proof of Lemma 6.

A.9 Lemma 7: Shifting Lemma

Lemma 7 (Shifting Lemma).

∀ f∈TFormulaCore, n,p∈ℕ: s∈Stream, e∈Environment, b∈Bool:
    n>0 ⇒ next(f) →*(n+1,p,s,e) done(b) ⇒ next(f) →*(n,p+1,s,e) done(b)

Proof

We take f,n,p,s,e,b arbitrary but fixed, assume

(1) n>0
(2) next(f) →*(n+1,p,s,e) done(b)

and show

[3] next(f) →*(n,p+1,s,e) done(b).

From (2), by the definition of →*, there exists Ft'∈TFormula such that

(4) next(f) →(p,s↓p,s(p),c) Ft'
(5) Ft' →*(n,p+1,s,e) done(b)

where

(6) c = (e,{(X,s(e(X))) | X ∈ dom(e)}).

Since n>0 by (1), we have that Ft' is a 'next' formula, say next(f').
Then from (5), by the definition of →*, we know that there exists
Ft''∈TFormula such that

(7) next(f') →(p+1,s↓(p+1),s(p+1),c) Ft''
(8) Ft'' →*(n-1,p+2,s,e) done(b).

In order to prove [3], by the definition of →*, we need to find such a
Ft0∈TFormula that

[9] next(f) →(p+1,s↓(p+1),s(p+1),c) Ft0
[10] Ft0 →*(n-1,p+2,s,e) done(b).

We take Ft0=Ft''. Then [10] follows from (8). We only need to prove [9]:

Given

(4) next(f) → (p,s↓p,s(p),c) next(f')
(7) next(f') →(p+1,s↓(p+1),s(p+1),c) Ft''

Prove:

[9] next(f) →(p+1,s↓(p+1),s(p+1),c) Ft''.

It follows from Lemma 8.

A.10 Lemma 8: Triangular Reduction Lemma

```
Lemma 8 (Triangular Reduction G).

  ∀G1,G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(G1) →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(G1) →(p+1,s↓(p+1),s(p+1),c) Ft.

Proof
-----

Φ ⊆ TFormulaCore
Φ(G1) :⇔
  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(G1) →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(G1) →(p+1,s↓(p+1),s(p+1),c) Ft.

  We prove

(G) ∀G'∈TFormulaCore : Φ(G').

Case (C1) G' = TN(Ft) for some Ft ∈ TFormula
----------------------------------------

We show

   Φ(G')

Take F2f,Ftf,pf,sf,cf arbitrary but fixed.

Assume

(C1.1) next(TN(Ft)) →(pf,sf↓pf,sf(pf),cf) next(G2f)
(C1.2) next(G2f) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf

Show

[C1.a] next(TN(Ft)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C1.1) and Def.→, we know for some G2' ∈ TFormula

(C1.3) G2f = TN(next(G2'))
(C1.4) next(TN(Ft)) →(pf,sf↓pf,sf(pf),cf) next(TN(next(G2')))
(C1.5) Ft →(pf,sf↓pf,sf(pf),cf) next(G2')

From (C1.2,C1.3), we thus have

(C1.6) next(TN(next(G2'))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf

From (C1.5) and Def. →, we know for some G ∈ TFormulaCore
```

(C1.7) Ft = next(G)
(C1.8) next(G) →(pf,sf↓pf,sf(pf),cf) next(G2')

From (C1.7) and [C1.a], it suffices to show

[C1.b] next(TN(next(G))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C1,C1.8) and the induction assumption, we know Φ(G) and thus

(C1.9)

```
  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(G) →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(G) →(p+1,s↓(p+1),s(p+1),c) Ft.
```

From (C1.6) and Def.→, we have 3 cases.

Case C1.c1. there exists some Fc'∈TFormulaCore such that

(C1.c1.1) next(G2') →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc')
(C1.c1.2) Ftf=next(TN(next(Fc')))

From (C1.c1.2) and [C1.b], ut suffices thus to show

[C1.c1.b] next(TN(next(G))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(TN(next(Fc')))

From (C1.9), (C1.8), (C1.c1.1), we have

(C1.c1.3) next(G) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc')

From (C1.c1.3) and Def. →, we know [C1.c1.b].

This proves the case C1.c1.

Case C1.c2. we have

(C1.c2.1) next(G2') →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true)
(C1.c2.2) Ftf=done(false)

From (C1.c2.2) and [C1.b], it suffices thus to show

[C1.c2.b] next(TN(next(G))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false)

From (C1.9), (C1.8), (C1.c2.1), we have

(C1.c22.3) next(G) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).

From (C1.c2.3) and Def. →, we know [C1.c2.b].

This proves the case C1.c2.

```
Case C1.c3. we have
-------------

(C1.c3.1) next(G2’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false)
(C1.c3.2) Ftf=done(true)

It suffices thus to show

[C1.c3.b] next(TN(next(G))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true)

From (C1.9), (C1.8) (C1.c3.1), we have

(C1.c3.3) next(G) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).

From (C1.c3.3) and Def. →, we know [C1.c3.b].

This proves the case C1.c3.

This finishes the proof of case C1.
-----------------------------------

Case (C2) G’ = TCS(Ft1,Ft2) for some Ft1,Ft2 ∈ TFormula.

We show

   Φ(G’)

Take F2f,Ftf,pf,sf,cf arbitrary but fixed.

Assume

(C2.1) next(TCS(Ft1,Ft2)) →(pf,sf↓pf,sf(pf),cf) next(G2f)
(C2.2) next(G2f) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf

Show

[C2.a] next(TCS(Ft1,Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C2.1), by Def.→, we have two cases:

Case C2.c1. There exists Fc1∈TFormulaCore such that
----------
(C2.c1.1)  G2f = TCS(next(Fc1),Ft2)
(C2.c1.2)  next(TCS(Ft1,Ft2)) →(pf,sf↓pf,sf(pf),cf) next(TCS(next(Fc1),Ft2) )
(C2.c1.3)  Ft1 →(pf,sf↓pf,sf(pf),cf) next(Fc1)

From (C2.2) and (C2.c1.1) we have

(C2.c1.4) next(TCS(next(Fc1),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C2.c1.3) and Def.→, we know for some Fc0 ∈ TFormulaCore

(C2.c1.5) Ft1 = next(Fc0)
(C2.c1.6) next(Fc0) →(pf,sf↓pf,sf(pf),cf) next(Fc1)
```

From (C2.c1.5) and [C2.a], we need to show

[C2.c1.b] next(TCS(next(Fc0),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C2),(C2.c1.5) and the induction hypothesis, we know Φ(Fc0) and thus

(C2.c1.7)

```
  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(Fc0) →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(Fc0) →(p+1,s↓(p+1),s(p+1),c) Ft.
```

From (C2.c1.4), we have the following cases.

Case C2.c1.c1. There exists Fc'∈TFormulaCore such that

```
(C2.c1.c1.1) Ftf = next(TCS(next(Fc'),Ft2))
(C2.c1.c1.2) next(TCS(next(Fc1),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
               next(TCS(next(Fc'),Ft2)).
(C2.c1.c1.3) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc').
```

From (C2.c1.c1.1) and [C2.c1.b], we need to show

```
[C2.c1.c1.b] next(TCS(next(Fc0),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
               next(TCS(next(Fc'),Ft2)).
```

In this case from (C2.c1.6), (C2.c1.c1.3), and (C2.c1.7) we have

(C2.c1.c1.4) next(Fc0) →(p+1,s↓(p+1),s(p+1),c) next(Fc').

From (C2.c1.c1.4), by the definition of →, we get [C2.c1.c1.b].

This proves the case C2.c1.c1.

Case C2.c1.c2.

```
(C2.c1.c2.1) Ftf = done(false)
(C2.c1.c2.2) next(TCS(next(Fc1),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).
(C2.c1.c2.3) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).
```

From (C2.c1.c2.1) and [C2.c1.b], we need to show

[C2.c1.c2.b] next(TCS(next(Fc0),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).

From (C2.c1.6), (C2.c1.c2.3) and (C2.c1.7) we have

(C2.c1.c2.4) next(Fc0) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).

From (C2.c1.c2.4), by the definition of →, we get [C2.c1.c2.b].

This proves the case C2.c1.c2.

Case C2.c1.c3. There exists Ft2'∈TFormula such that

(C2.c1.c3.1) Ftf = Ft2'
(C2.c1.c3.2) next(TCS(next(Fc1),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ft2'.
(C2.c1.c3.3) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).
(C2.c1.c3.4) Ft2 →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ft2'.

From (C2.c1.c3.1) and [C2.c1.b], we need to show

[C2.c1.c3.b] next(TCS(next(Fc0),Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ft2'.

From (C2.c1.6), (C2.c1.c3.3), and (C2.c1.7) we have

(C2.c1.c3.5) next(Fc0) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).

From (C2.c1.c3.5) and (C2.c1.c3.4), by Def.→, we get [C2.c1.c3.b].

This proves the case C2.c1.c2.

This proves the case C2.c1.

Case C2.c2.

Recall that we consider alternatives of G2f in

(C2.1) next(TCS(Ft1,Ft2)) →(pf,sf↓pf,sf(pf),cf) next(G2f)

Case C2.c1 considered the case when G2f = TCS(next(Fc1),Ft2).

According to Def.→, the other alternative for G2f is the following:
There exists G2'∈TFormulaCore such that

(C2.c2.1) G2f = G2'
(C2.c2.2) next(TCS(Ft1,Ft2)) →(pf,sf↓pf,sf(pf),cf) next(G2')
(C2.c2.3) Ft1 →(pf,sf↓pf,sf(pf),cf) done(true)
(C2.c2.4) Ft2 →(pf,sf↓pf,sf(pf),cf) next(G2')

From (C2.2) and (C2.c2.1) we have

(C2.c2.5) next(G2') →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C2.c2.3) and Def.→, we know for some Fc1 ∈ TFormulaCore

(C2.c2.6) Ft1 = next(Fc1)
(C2.c2.7) next(Fc1) →(pf,sf↓pf,sf(pf),cf) done(true)

From (C2.c2.4) and Def.→, we know for some Fc2 ∈ TFormulaCore

(C2.c2.8) Ft2 = next(Fc2)
(C2.c2.9) next(Fc2) →(pf,sf↓pf,sf(pf),cf) next(G2')

From (C2.c2.6), (C2.c2.8) and [C2.a], we need to show

[C2.c2.b] next(TCS(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C2.c2.7), by Lemma 6, we know

(C2.c2.10) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).

From (C2), (C2.c2.8) and the induction hypothesis, we know Φ(Fc2) and thus

(C2.c2.11)

```
  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(Fc2) →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(Fc2) →(p+1,s↓(p+1),s(p+1),c) Ft.
```

From (C2.c2.9), (C2.c2.5), and (C2.c2.11), we get

(C2.c2.11) next(Fc2) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C2.c2.10) and (C2.c2.11), by Def.→, we get [C2.c2.b].

This proves the case C2.c2.

This finsihes the proof of case C2.

Case (C3) G' = TCP(Ft1,Ft2) for some Ft1,Ft2 ∈ TFormula.

We show

   Φ(G')

Take F2f,Ftf,pf,sf,cf arbitrary but fixed.

Assume

(C3.1) next(TCP(Ft1,Ft2)) →(pf,sf↓pf,sf(pf),cf) next(G2f)
(C3.2) next(G2f) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf

Show

[C3.a] next(TCP(Ft1,Ft2)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C3.1), by Def.→, we have three cases.

Case C3.c1

There exists Fc1,Fc2∈TFormulaCore such that

(C3.c1.1) G2f = TCP(next(Fc1),next(Fc2))
(C3.c1.2) Ft1 →(pf,sf↓pf,sf(pf),cf) next(Fc1)
(C3.c1.3) Ft2 →(pf,sf↓pf,sf(pf),cf) next(Fc2)
(C3.c1.4) next(TCP(Ft1,Ft2)) →(pf,sf↓pf,sf(pf),cf) next(TCP(next(Fc1),next(Fc2)))

From (C3.2) and (C3.c1.1) we have

```
(C3.c1.5) next(TCP(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf

From (C3.c1.2) and Def.→, we know for some Fc1'∈TFormulCore

(C3.c1.6) Ft1 = next(Fc1')
(C3.c1.7) next(Fc1') →(pf,sf↓pf,sf(pf),cf) next(Fc1)

From (C3.c1.3) and Def.→, we know for some Fc2'∈TFormulCore

(C3.c1.8) Ft2 = next(Fc2')
(C3.c1.9) next(Fc2') →(pf,sf↓pf,sf(pf),cf) next(Fc2)

From (C3.c1.6), (C3.c1.8) and [C3.a], we need to show

[C3.c1.b] next(TCP(next(Fc1'),next(Fc2'))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C3), (C3.c1.6) and the induction hypothesis, we know Φ(Fc1') and thus

(C3.c1.10)

  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(Fc1') →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(Fc1') →(p+1,s↓(p+1),s(p+1),c) Ft.

From (C3), (C3.c1.8) and the induction hypothesis, we know Φ(Fc2') and thus

(C3.c1.11)

  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(Fc2') →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(Fc2') →(p+1,s↓(p+1),s(p+1),c) Ft.

From (C3.c1.5), by Def.→, we have the following five cases.

Case C3.c1.c1
-------------
There exist Fc1'', Fc2''∈TFormulaCore such that

(C3.c1.c1.1) Ftf = next(TCP(next(Fc1''),next(Fc2'')))
(C3.c1.c1.2) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc1'')
(C3.c1.c1.3) next(Fc2) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc2'')
(C3.c1.c1.4) next(TCP(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
             next(TCP(next(Fc1''),next(Fc2'')))

From (C3.c1.c1.1) and [C3.c1.b] we need to prove

[C3.c1.c1.b] next(TCP(next(Fc1'),next(Fc2'))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
             next(TCP(next(Fc1''),next(Fc2''))).

From (C3.c1.7), (C3.c1.c1.2), and (C3.c1.c1.10) we have
```

```
(C3.c1.c1.5) next(Fc1') →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc1'').

From (C3.c1.9), (C3.c1.c1.3), and (C3.c1.11) we have

(C3.c1.c1.6) next(Fc2') →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc2'')

From (C3.c1.c1.5) and (C3.c1.c1.6), by Def.→ we get [C3.c1.c1.b].

This proves case the C3.c1.c1.

Case C3.c1.c2
-------------
There exist Fc1''∈TFormulaCore such that

(C3.c1.c2.1) Ftf = next(Fc1'')
(C3.c1.c2.2) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc1'')
(C3.c1.c2.3) next(Fc2) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true)
(C3.c1.c2.4) next(TCP(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
               next(Fc1'')

From (C3.c1.c2.1) and [C3.c1.b] we need to prove

[C3.c1.c2.b] next(TCP(next(Fc1'),next(Fc2'))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
               next(Fc1'').

From (C3.c1.7), (C3.c1.c2.2), and (C3.c1.10) we have

(C3.c1.c2.5) next(Fc1') →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc1'').

From (C3.c1.9), (C3.c1.c2.3), and (C3.c1.11) we have

(C3.c1.c2.6) next(Fc2') →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).

From (C3.c1.c2.5) and (C3.c1.c2.6), by Def.→ we get [C3.c1.c2.b].

This proves the case C3.c1.c2.


Case C3.c1.c3
-------------
There exist Fc1''∈TFormulaCore such that

(C3.c1.c3.1) Ftf = done(false)
(C3.c1.c3.2) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc1'')
(C3.c1.c3.3) next(Fc2) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false)
(C3.c1.c3.4) next(TCP(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
               done(false)

From (C3.c1.c3.1) and [C3.c1.b] we need to prove

[C3.c1.c2.b] next(TCP(next(Fc1'),next(Fc2'))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
               done(false).

From (C3.c1.7), (C3.c1.c3.2), and (C3.c1.10) we have
```

(C3.c1.c3.5) next(Fc1’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc1’’).

From (C3.c1.9), (C3.c1.c3.3), and (C3.c1.11) we have

(C3.c1.c3.6) next(Fc2’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).

From (C3.c1.c3.5) and (C3.c1.c3.6), by Def.→ we get [C3.c1.c3.b].

This proves the case C3.c1.c3.

Case C3.c1.c4

(C3.c1.c4.1) Ftf = done(false)
(C3.c1.c4.2) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false)
(C3.c1.c4.3) next(TCP(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
done(false)

From (C3.c1.c4.1) and [C3.c1.b] we need to prove

[C3.c1.c4.b] next(TCP(next(Fc1’),next(Fc2’))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
done(false).

From (C3.c1.7), (C3.c1.c4.2), and (C3.c1.10) we have

(C3.c1.c4.5) next(Fc1’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false).

From (C3.c1.c4.5) by Def.→ we get [C3.c1.c4.b].

This proves the case C3.c1.c4.

Case C3.c1.c5

There exist Fc2’’∈TFormulaCore such that

(C3.c1.c5.1) Ftf = next(Fc2’’)
(C3.c1.c5.2) next(Fc1) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true)
(C3.c1.c5.3) next(Fc2) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc2’’)
(C3.c1.c5.4) next(TCP(next(Fc1),next(Fc2))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
next(Fc2’’)

From (C3.c1.c5.1) and [C3.c1.b] we need to prove

[C3.c1.c5.b] next(TCP(next(Fc1’),next(Fc2’))) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
next(Fc2’’).

From (C3.c1.7), (C3.c1.c5.2), and (C3.c1.10) we have

(C3.c1.c5.5) next(Fc1’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).

From (C3.c1.9), (C3.c1.c5.3), and (C3.c1.11) we have

(C3.c1.c5.6) next(Fc2’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) next(Fc2’’).

From (C3.c1.c5.5) and (C3.c1.c5.6), by Def.→ we get [C3.c1.c5.b].

This proves the case C3.c1.c3.

This proves the case C3.c1.

Case C3.c2

(C3.c2.1) Ft1 →(pf,sf↓pf,sf(pf),cf) next(G2f)
(C3.c2.2) Ft2 →(pf,sf↓pf,sf(pf),cf) done(true)

From (C3.c2.1) and Def.→, we know for some Fc1’∈TFormulCore

(C3.c2.3) Ft1 = next(Fc1’)
(C3.c2.4) next(Fc1’) →(pf,sf↓pf,sf(pf),cf) next(G2f)

From (C3.c2.2) and Def.→, we know for some Fc2’∈TFormulCore

(C3.c2.5) Ft2 = next(Fc2’)
(C3.c2.6) next(Fc2’) →(pf,sf↓pf,sf(pf),cf) done(true)

From (C3.c2.3), (C3.c2.5) and [C3.a], we need to show

[C3.c2.b] next(TCP(next(Fc1’),next(Fc2’))) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C3), (C3.c2.2) and the induction hypothesis, we know Φ(Fc1’) and thus

(C3.c2.7)

  ∀G2∈TFormulaCore, Ft∈TFormula, p∈ℕ, s∈Stream, c∈Context :
  next(Fc1’) →(p,s↓p,s(p),c) next(G2) ∧ next(G2) →(p+1,s↓(p+1),s(p+1),c) Ft
  ⇒
  next(Fc1’) →(p+1,s↓(p+1),s(p+1),c) Ft.

From (C3.c2.4), (C3.2), and (C3.c2.7) we get

(C3.c2.8) next(Fc1’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

From (C3.c2.6), by Lemma 6, we get

(C3.c3.9) next(Fc2’) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).

From (C3.c2.8) and (C3.c2.9), by Def.→, we get [C3.c2.b].

This proves the vase C3.c2

Case C3.c3

(C3.c3.1) Ft1 $\rightarrow$(pf,sf$\downarrow$pf,sf(pf),cf) done(true)
(C3.c3.2) Ft2 $\rightarrow$(pf,sf$\downarrow$pf,sf(pf),cf) next(G2f)

This case can be proved similarly to case C3.c2.

This finishes the proof of C3.

Case (C4) G' = TA(X,b1,b2,Ft) for some X$\in$Variable, b1,b2$\in$BoundValue,
Ft $\in$ TFormula.

We show

Φ(G')

Take F2f,Ftf,pf,sf,cf arbitrary but fixed.

Assume

(C4.1) next(TA(X,b1,b2,Ft)) $\rightarrow$(pf,sf$\downarrow$pf,sf(pf),cf) next(G2f)
(C4.2) next(G2f) $\rightarrow$(pf+1,sf$\downarrow$(pf+1),sf(pf+1),cf) Ftf

Show

[C4.a] next(TA(X,b1,b2,Ft)) $\rightarrow$(pf+1,sf$\downarrow$(pf+1),sf(pf+1),cf) Ftf.

From (C4.1), by Def.$\rightarrow$, we have that there exist p1,p2$\in\mathbb{N}$ such that

(C4.3) p1=b1(cf)
(C4.4) p2=b2(cf)
(C4.5) p1$\neq\infty$
(C4.6) next(TA0(X,p1,p2,Ft)) $\rightarrow$(pf,sf$\downarrow$pf,sf(pf),cf) next(G2f)

To prove [C4.a], by the definition of $\rightarrow$, we would have two alternatives:
Ftf=done(true) or Ftf$\neq$done(true). But the case Ftf=done(true) is impossible
because of (C4.5). Hence, we assume Ftf$\neq$done(true) and prove

[C4.a.1] p1=b1(cf)
[C4.a.2] p2=b2(cf)
[C4.a.3] p1$\neq\infty$
[C4.a.4] next(TA0(X,p1,p2,Ft)) $\rightarrow$(pf+1,sf$\downarrow$(pf+1),sf(pf+1),cf) Ftf.

[C4.a.1-3] are immediately proved due to (C4.3-5).

To prove [C4.a.4], from (C4.6), by Def.$\rightarrow$, we consider two cases.

Case C4.c1.

In this case from (C4.6) we have

(C4.c1.1) pf<p1

```
(C4.c1.2) next(TA0(X,p1,p2,Ft)) →(pf,sf↓pf,sf(pf),cf) next(TA0(X,p1,p2,Ft))
(C4.c1.3) next(G2f) = next(TA0(X,p1,p2,Ft))

From (C4.2) and (C4.c1.3) we get [C4.a.4]

This finishes the proof of C4.c1.

Case C4.c2.
-----------
In this case  from (C4.6) we have

(C4.c2.1) pf≥p1
(C4.c2.2) fs = {(p0,Ft,(cf.1[X↦p0],cf.2[X↦sf(p0)])) | p1 ≤ p0 <∞ min∞(pf,p2+∞1)}
(C4.c2.3) next(TA1(X,p2,Ft,fs)) →(pf,sf↓pf,sf(pf),cf) next(G2f)

From (C4.c2.3), by the definition of →, we know

(C4.c2.4) G2f = TA1(X,p2,Ft,fs1), where

(C4.c2.5) fs0 =
            if pf >∞ p2 then
               fs
            else
               fs ∪ {(pf,Ft,(cf.1[X↦pf],cf.2[X↦sf(pf)]))}
(C4.c2.6) ¬∃t∈ℕ,g∈TFormula,c∈Context:
            (t,g,c)∈fs0 ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) done(false)
(C4.c2.7) fs1 = { (t,next(fc),c) ∈ TInstance |
                    ∃g∈TFormula: (t,g,c)∈fs0 ∧
                    ⊢ g →(pf,sf↓pf,sf(pf),c) next(fc) }
(C4.c2.8) ¬(fs1 = ∅ ∧ pf ≥∞ p2)

From (C4.2) and (C4.c2.4) we have

(C4.c2.9) next(TA1(X,p2,Ft,fs1)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

Recall that we need to prove

[C4.a.4] next(TA0(X,p1,p2,Ft)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf.

By definition of → and (C4.c2.1), in order to prove [C4.a.4], we need to prove

[C4.a.5] next(TA1(X,p2,Ft,fs')) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf,

where

(C4.c2.10) fs' = {(p0,Ft,(cf.1[X↦p0],cf.2[X↦sf(p0)])) |
                      p1 ≤ p0 <∞ min∞(pf+1,p2+∞1)}.

Note that if pf >∞ p2 then min∞(pf+1,p2+∞1)=min∞(pf,p2+∞1)
else min∞(pf+1,p2+∞1)=pf+1. therefore, from (C4.c2.2), (C4.c2.5), and
(C4.c2.10) we have

(C4.c2.11) fs'=fs0.
```

Hence, we need to prove

```
[C4.a.6] next(TA1(X,p2,Ft,fs0)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) Ftf,
```

We prove [C4.a.6] by case distinction over Ftf.

```
Ftf = done(false)
-----------------
```

In this case, from (C4.c2.9) we get

```
(C4.c2.12) next(TA1(X,p2,Ft,fs1)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(false)
```

From (C4.c2.12), by the definition of → for forall we have

```
(C4.c2.13) ∃t∈ℕ,g∈TFormula,c∈Context:
              (t,g,c)∈fs1' ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
```

where

```
(C4.c2.14) fs1' =
             if pf+1 >∞ p2 then
                fs1
             else fs1 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}.
```

Take (t1,g1,c1) which is a witness for (C4.c2.13). That means, we have

```
(C4.c2.13') (t1,g1,c1)∈fs1' and
(C4.c2.13'') g1 →(pf+1,sf↓(pf+1),sf(pf+1),c1) done(false).
```

Assume first

```
(C4.c2.15) pf+1>∞ p2, which from (C4.c2.14) gives
---------------------------------------------------
(C4.c2.16) (t1,g1,c1)∈fs1.
```

To show [C4.a.6], we need to prove

```
[C4.a.7] ∃t∈ℕ,g∈TFormula,c∈Context:
            (t,g,c)∈fs0' ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
```

where

```
(C4.c2.17) fs0' =
              if pf+1 >∞ p2 then
                 fs0
              else fs0 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}.
```

From (C4.c2.15) and (C4.c2.17), we have

```
(C4.c2.18) fs0'=fs0.
```

from (C4.c2.16), by (C4.c2.7), there exists g0∈TFormula and fc1∈TFormulaCore such that

```
(C4.c2.19) g1=next(fc1)
(C4.c2.20) (t1,g0,c1)∈fs0
(C4.c2.21) ⊢ g0 →(pf,sf↓pf,sf(pf),c1) next(fc1)
```

From (C4.c2.21), by the definition of →, there exists fc0∈TFormulaCore such that

```
(C4.c2.22) g0=next(fc0).
```

From (C4.c2.13'), (C4.c2.19), and (C4.c2.13'') we know

```
(C4.c2.23) ⊢ next(fc1) →(pf+1,sf↓(pf+1),sf(pf+1),c1) done(false).
```

From (C4.c2.21), (C4.c2.22), (C4.c2.23), by the induction hypothesis, we get

```
(C4.c2.24) ⊢ g0 →(pf+1,sf↓(pf+1),sf(pf+1),c1) done(false).
```

From (C4.c2.18) and (C4.c2.20), we get

```
(C4.c2.25) (t1,g0,c1)∈fs0'.
```

From (C4.c2.25) and (C4.c2.24), we get [C4.a.7].

Now assume

```
(C4.c2.26) pf+1≤∞ p2, which from (C4.c2.14) gives
--------------------------------------------------
(C4.c2.27) (t1,g1,c1)∈ fs1 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}.
```

Recall:

To show [C4.a.6], we need to prove

```
[C4.a.7] ∃t∈ℕ,g∈TFormula,c∈Context:
            (t,g,c)∈fs0' ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
```

where

```
(C4.c2.17) fs0' =
             if pf+1 >∞ p2 then
                fs0
             else
                fs0 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}.
```

From (C4.c2.26) and (C4.c2.17), we have

```
(C4.c2.28) fc0'=fs0 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}.
```

If (t1,g1,c1)∈ fs1, the proof proceeds as for the case pf+1>∞ p2 above.

Consider

```
(C4.c2.29) (t1,g1,c1) = (pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)])).
```

From (C4.c2.28) and (C4.c2.29) we have

```
(C4.c2.30) (t1,g1,c1)∈fc0'
```

From (C4.c2.30) and (C4.c2.13'') we get [C4.a.7].

This finishes the proof of the case Ftf=done(false).

Ftf = done(true). The case p1=∞ is excluded due to (C4.5), and Def. of →.

Hence, we need to prove

```
[C4.a.true.1] next(TA1(X,p2,Ft,fs0)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true),
```

which by Def.→ means, we need to prove

```
[C4.a.true.2] ¬∃t∈ℕ,g∈TFormula,c∈Context:
                 (t,g,c)∈fs00 ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
[C4.a.true.3] fs01= ∅ ∧ pf+1 ≥∞ p2,
```

where

```
(C4.c2.true.1) fs00 =
                  if pf+1 >∞ p2 then
                     fs0
                  else fs0 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],c.2[X↦sf(pf+1)]))}
(C4.c2.true.2) fs01 =
        { (t,next(fc),c) ∈ TInstance |
              ∃g∈TFormula: (t,g,c)∈fs00 ∧
                             ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) next(fc) }
```

On the other hand, from (C4.c2.9) we know

```
(C4.c2.true.3) next(TA1(X,p2,Ft,fs1)) →(pf+1,sf↓(pf+1),sf(pf+1),cf) done(true).
```

From (C4.c2.true.3), by Def. →, we know

```
(C4.c2.true.4) ¬∃t∈ℕ,g∈TFormula,c∈Context:
                 (t,g,c)∈fs10 ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
(C4.c2.true.5) fs11 = ∅ ∧ pf+1 ≥∞ p2
```

where

```
(C4.c2.true.6) fs10 =
                 if pf+1 >∞ p2 then
                    fs1
                 else fs1 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],c.2[X↦sf(pf+1)]))}
(C4.c2.true.7) fs11 =
        { (t,next(fc),c) ∈ TInstance |
             ∃g∈TFormula: (t,g,c)∈fs10 ∧
                           ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) next(fc) }
```

Recall the relationshsip between fs0 and fs1:

```
(C4.c2.7) fs1 =
             { (t,next(fc),c) ∈ TInstance |
                 ∃g∈TFormula: (t,g,c)∈fs0 ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) next(fc) }
```

From (C4.c2.true.6), (C4.c2.true.7), and (C4.c2.true.5) we know that

```
(C4.c2.true.8) ¬∃fc∈TFormulaCore:
                    Ft →(pf+1,sf↓(pf+1),sf(pf+1),cf.1[X↦pf+1]) next(fc).
```

Now assume by contradiction that for some (t0,g0,c0)∈fs0 we have

(C4.c2.true.9) ∃fc∈TFormulaCore: g0 →(pf+1,sf↓(pf+1),sf(pf+1),c0) next(fc)

From (C4.c2.true.9), by Lemma 6, there exist fc0∈TFormulaCore such that

(C4.c2.true.10) g0 →(pf,sf↓(pf),sf(pf),c0) next(fc0)

From (C4.c2.true.9) by (C4.c2.7) we have that there exists fc0∈TFormulaCore
such that

(C4.c2.true.11) (t0,next(fc0),c0)∈fs1.

From (C4.c2.true.11) by (C4.c2.true.6) we get

(C4.c2.true.12) (t0,next(fc0),c0)∈fs10.

From (C4.c2.true.12) by (C4.c2.true.7), (C4.c2.true.5), (C4.c2.true.4), we get

(C4.c2.true.13) next(fc0) →(pf+1,sf↓(pf+1),sf(pf+1),c0) done(true)

From (C4.c2.true.10) and (C4.c2.true.13), by the induction hypothesis, we get

(C4.c2.true.14) g0 →(pf+1,sf↓(pf+1),sf(pf+1),c0) done(true)

But (C4.c2.true.14) contradicts (C4.c2.true.9). Hence, we know that for all
(t,g,c)∈fs0

(C4.c2.true.15) ¬∃fc∈TFormulaCore: g →(pf+1,sf↓(pf+1),sf(pf+1),c) next(fc)

From (C4.c2.true.8) and (C4.c2.true.15) we know that for all
(t,g,c)∈fs00

(C4.c2.true.16) ¬∃fc∈TFormulaCore: g →(pf+1,sf↓(pf+1),sf(pf+1),c) next(fc).

From (C4.c2.true.16) we get

(C4.c2.true.17) fs01= ∅

From (C4.c2.true.17) and the second conjunct of (C4.c2.true.5) we get [C4.a.true.3].

To prove [C4.a.true.2] note that from (C4.c2.true.4) and (C4.c2.true.6) we have

```
(C4.c2.true.18) Ft →(pf+1,sf↓(pf+1),sf(pf+1),cf.1[X↦pf+1]) done(false) does not hold.
```

Recall that in (C4.c2.6) we have

```
(C4.c2.6) ¬∃t∈ℕ,g∈TFormula,c∈Context:
              (t,g,c)∈fs0 ∧ ⊢ g →(pf,sf↓pf,sf(pf),c) done(false)
```

Hence, for no (t,g,c)∈fs00 we have g →(pf,sf↓pf,sf(pf),c) done(false).
It proves [C4.a.true.2].

Ftf is a 'next' formula.

Let Ftf = next(TA1(X,p2,Ft,fs2)) for some fs2. Then from [C4.a.6] and (C4.c2.11), we need to prove

```
[C4.a.next.1] next(TA1(X,p2,Ft,fs0)) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
                next(TA1(X,p2,Ft,fs2))
```

To prove [C4.a.next.8], we define

```
(C4.c2.next.1) fs00 :=
                 if pf+1 >∞ p2 then
                    fs0
                 else fs0 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}

(C4.c2.next.2) fs01 :=
                  { (t,next(fc),c) ∈ TInstance |
                        ∃g∈TFormula: (t,g,c)∈fs00 ∧
                          ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) next(fc) }
```

and prove

```
[C4.a.next.2] ¬∃t∈ℕ,g∈FormulaStep,c∈Context:
                 (t,g,c)∈fs00 ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
[C4.a.next.3] ¬(fs01 = ∅ ∧ pf+1 ≥∞ p2)
```

On the other hand, from (C4.c2.9) we know

```
(C4.c2.next.3) next(TA1(X,p2,Ft,fs1)) →(pf+1,sf↓(pf+1),sf(pf+1),cf)
                 next(TA1(X,p2,Ft,fs2)).
```

From (C4.c2.next.3), by Def.→, we know

```
(C4.c2.next.4) ¬∃t∈ℕ,g∈FormulaStep,c∈Context:
                    (t,g,c)∈fs10 ∧ ⊢ g →(pf+1,sf↓(pf+1),sf(pf+1),c) done(false)
(C4.c2.next.5) ¬(fs11 = ∅ ∧ pf+1 ≥∞ p2)
```

where

```
(C4.c2.next.6) fs10 =
```

```
                 if pf+1 >∞ p2 then
                    fs1
                 else fs1 ∪ {(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))}
(C4.c2.next.7) fs11 =
                  { (t,next(fc),c) ∈ TInstance |
                          ∃g∈TFormula: (t,g,c)∈fs10 ∧
                              ⊢ g →((pf+1,sf↓(pf+1),sf(pf+1),c) next(fc) }
```

Recall the relation between fs0 and fs1:

```
(C4.c2.7) fs1 = { (t,next(fc),c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈fs0 ∧
                    ⊢ g →(pf,sf↓pf,sf(pf),c) next(fc) }
```

By (C4.c2.6) and (C4.c2.next.1), to prove [C4.a.next.2], it suffices to prove that

```
[C4.a.next.4] ⊢ Ft →(pf+1,sf↓(pf+1),sf(pf+1),(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)]))
                  done(false) does not hold.
```

But this directly follows from (C4.c2.next.6) and (C4.c2.next.4).
Hence, [C4.a.next.4] is proved.

To prove [C4.a.next.3], we assume

(C4.c2.next.8) pf+1 ≥∞ p2

and prove

[C4.a.next.5] fs01 ≠ ∅.

From (C4.c2.next.8) and (C4.c2.next.5) we know

(C4.c2.next.9) fs11 ≠ ∅.

From (C4.c2.next.9), there exist (t1,g1,c1)∈fs10 and fc1∈TFormulaCore such that

(C4.c2.next.9) ⊢ g1 →(pf+1,sf↓(pf+1),sf(pf+1),c1) next(fc1).

According to (C4.c2.next.6), (t1,g1,c1)∈fs10 means either (t1,g1,c1)∈fs1 or
(t1,g1,c1)=(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)])

First assume (t1,g1,c1)∈fs1.

By (C4.c2.7), it means that there exist (t0,g0,c0)∈fs0 and fc0∈TFormulaCore
such that

(C4.c2.next.10) ⊢ g0 →(pf,sf↓pf,sf(pf),c0) next(fc0)
(C4.c2.next.11) g1=next(fc0)

Moreover, g0 is a 'next' formula.

(C4.c2.next.12) g0 = next(fc) for some fc∈TFormulaCore.

Besides, from (C4.c2.7) one can see that

(C4.c2.next.13) c0=c1.

Hence, from (C4.c2.next.9--13) we have

(C4.c2.next.14) next(fc) →(pf,sf↓pf,sf(pf),c0) next(fc0)
(C4.c2.next.15) next(fc0)→(pf+1,sf↓(pf+1),sf(pf+1),c0) next(fc1)

From (C4.c2.next.14) and (C4.c2.next.15), by the induction hypothesis,
we obtain that

(C4.c2.next.16) next(fc)→(pf+1,sf↓(pf+1),sf(pf+1),c0) next(fc1)

Hence, we got that for (t0,g0,c0)∈fs0 and fc1∈TFormulaCore

(C4.c2.next.17) g0→(pf+1,sf↓(pf+1),sf(pf+1),c0) next(fc1).

By definition (C4.c2.next.1) of fs00, we have (t0,g0,c0)∈fs00.

Now assume (t1,g1,c1)=(pf+1,Ft,(cf.1[X↦pf+1],cf.2[X↦sf(pf+1)])

Trivially, by definition (C4.c2.next.1) of fs00, we have (t1,g1,c1)∈fs00.

Hence, in both cases we found a triple

(C4.c2.next.18) (t,g,c)∈fs00

such that

(C4.c2.next.19) g→(pf+1,sf↓(pf+1),sf(pf+1),c) next(fc1)

holds. (C4.c2.next.18), (C4.c2.next.19), and (C4.c2.next.2) imply [C4.a.next.5].

This finishes the proof of the case Ftf is a 'next' formula.

This finishes the proof of C4.c2.

This finishes the proof of C4.

This finishes the proof of Lemma 8.

A.11 Lemma 9: Soundness of Bound Analysis

```
∀ re∈RangeEnv, e∈Environment, p∈ℕ, s∈Stream, B∈Bound, l,u∈ℤ∞:
  re ⊢ B : (l,u) ∧ dom(e) = dom(re) ∧
  (∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p) ⇒
    let c := (e,{X, s(e(X)) | X∈dom(e)}):
    l +i p ≤i T(B)(c) ≤i u +i p
```

Proof

Denote

```
Φ(B) :⇔
  ∀ re∈RangeEnv, e∈Environment, p∈ℕ, s∈Stream, l,u∈ℤ∞:
  re ⊢ B : (l,u) ∧ dom(e) = dom(re) ∧
  (∀Y∈dom(e): re(Y).1 +i p ≤i e(Y) ≤i re(Y).2 +i p) ⇒
    let c := (e,{X, s(e(X)) | X∈dom(e)}):
    l +i p ≤i T(B)(c) ≤i u +i p
```

Then we need to prove

```
[1] ∀B∈Bound: Φ(B).
```

We prove [1] by structural induction over B.

(a). B=0.

We take ref, ef, pf, sf, Bf, lf, uf arbitrary but fixed, assume

```
(a1) ref ⊢ B : (lf,uf)
(a2) dom(ef) = dom(ref)
(a3) ∀Y∈dom(e): ref(Y).1 +i pf ≤i ef(Y) ≤i ref(Y).2 +i pf,
(a4) c = (ef,{X, sf(ef(X)) | X∈dom(ef)})
```

and prove

```
[a5] lf +i pf ≤i T(B)(c) ≤i uf +i pf
```

By the translation, we have

```
(a6) T(B)(c) = 0, when B=0.
```

Therefore, we need to prove

```
[a7] lf +i pf ≤i 0 ≤i uf +i pf
```

By the analysis rules, we have

```
(a8) ref ⊢ B : (-∞,0), when B=0.
```

That means, from (a8) and (a1) we need to consider the case, when

(a9) lf = $-\infty$
(a10) uf = 0.

From the definition of +i, we have $-\infty$ + n = $-\infty$. Hence, from (a9,a10) we need to prove

[a11] $-\infty \leq$i 0 $\leq$i 0

which obviously holds. Hence, the case (a) is proved.

(b). B = ∞.

We take ref, ef, pf, sf, Bf, lf, uf arbitrary but fixed, assume

(b1) ref $\vdash$ B : (lf,uf)
(b2) dom(ef) = dom(ref)
(b3) $\forall$Y$\in$dom(e): ref(Y).1 +i pf $\leq$i ef(Y) $\leq$i ref(Y).2 +i pf,
(b4) c = (ef,{X, sf(ef(X)) | X$\in$dom(ef)})

and prove

[b5] lf +i pf $\leq$i T(B)(c) $\leq$i uf +i pf

By the translation, we have

(b6) T(B)(c) = ∞, when B=∞.

Therefore, we need to prove

[b7] lf +i pf $\leq$i ∞ $\leq$i uf +i pf

By the analysis rules, we have

(b8) ref $\vdash$ B : (∞,∞), when B=∞.

That means, from (b7) and (b1) we need to consider the case, when

(b9) lf = ∞
(b10) uf = ∞.

Hence, from (b9,b10) we need to prove

[b11] $\infty \leq$i $\infty \leq$i ∞

which obviously holds. Hence, the case (b) is proved.

(c). B=X.

We take ref, ef, pf, sf, Bf, lf, uf arbitrary but fixed, assume

(c1) ref $\vdash$ B : (lf,uf)
(c2) dom(ef) = dom(ref)

(c3) $\forall$Y$\in$dom(e): ref(Y).1 +i pf $\leq$i ef(Y) $\leq$i ref(Y).2 +i pf,
(c4) c = (ef,{X, sf(ef(X)) | X$\in$dom(ef)})

and prove

[c5] lf +i pf $\leq$i T(B)(c) $\leq$i uf +i pf

By the analysis rules, we have two subcases:

(c.case1) X$\notin$dom(ref)

In this case, by (c2) and (c3) we have X$\notin$dom(ef)=dom(c.1).

By the translation, we have

(c.case1.1) T(X)(c) = 0, when B=X and X$\notin$dom(c.1).

Therefore, we need to prove

[c.case1.2] lf +i pf $\leq$i 0 $\leq$i uf +i pf.

By the analysis rules, in this subcase we have

(c.case1.3) ref $\vdash$ X : (-∞,0), when B=X and X $\notin$ dom(ref).

From (c.case1.3) and (c1) we get

(c.case1.4) lf = -∞
(c.case1.5) uf = 0.

Therefore, to prove [c.case1.2], we need to prove

[c.case1.3] -∞ +i pf $\leq$i 0 $\leq$i 0 +i pf,

which holds, because -∞ +i pf = -∞. It proves the subcase (c.case1).

(c.case2) X$\in$dom(ref)

In this case, by (c2) and (c3) we have X$\in$dom(ef)=dom(c.1).

By the translation, we have

(c.case2.1) T(X)(c) = c.1(X)=ef(X), when B=X and X$\in$dom(c.1).

Therefore, we need to prove

[c.case2.2] lf +i pf $\leq$i ef(X) $\leq$i uf +i pf

By the analysis rules, in this subcase we have

(c.case2.3) ref $\vdash$ X : ref(X), when B=X and X$\in$dom(ref).

From (c.case2.3) and (c1) we get

(c.case2.4) lf = ref(X).1
(c.case2.5) uf = ref(X).2

Therefore, to prove [c.case2.2], we need to prove

[c.case2.3] ref(X).1 +i pf $\leq$i ef(X) $\leq$i ref(X).2 +i pf,

which follows from (c3). The case (c.case2) is proved.

d. B=B0+N, for B0$\in$Bound and N$\in\mathbb{N}$

We take ref, ef, pf, sf, Bf, lf, uf arbitrary but fixed, assume

(d1) ref $\vdash$ B : (lf,uf)
(d2) dom(ef) = dom(ref)
(d3) $\forall$Y$\in$dom(e): ref(Y).1 +i pf $\leq$i ef(Y) $\leq$i ref(Y).2 +i pf,
(d4) c = (ef,{X, sf(ef(X)) | X$\in$dom(ef)})

and prove

[d5] lf +i pf $\leq$i T(B)(c) $\leq$i uf +i pf.

By the translation, we have

(d6) T(B)(c) = T(B0)(c)+$[\![$N$]\!]$, when B=B0+N.

Assume that

(d7) ref $\vdash$ B0 : (l0,u0).

Then, by the analysis rules, since ref $\vdash$ B+N : (l0 +i $[\![$N$]\!]$, u0 +i $[\![$N$]\!]$), we have
from (d7) and (d1):

(d8) lf = l0 +i $[\![$N$]\!]$
(d9) uf = u0 +i $[\![$N$]\!]$

and we need to prove

[d10] l0 +i $[\![$N$]\!]$ +i pf $\leq$i T(B0)(c)+$[\![$N$]\!]$ $\leq$i u0 +i $[\![$N$]\!]$ +i pf.

By the induction hypothesis for B0 we have

(d11) l0 +i pf $\leq$i T(B0)(c) $\leq$i u0 +i pf,

which implies [d10]. It proves the case (d).

(e) B=B0-N, for B0$\in$Bound and N$\in\mathbb{N}$.

Similar to the case (d).

A.12 Lemma 10: Invariant Lemma for Universal Formulas

```
∀X∈Variable,b1∈BoundValue,b2∈BoundValue,f∈TFormulaCore:
  ∀n∈ℕ: n≥1 ⇒ forall(n,X,b1,b2,next(f))

Predicates
----------

forall ⊆ ℕ × Variable × BoundValue × BoundValue × TFormula:
forall(n,X,b1,b2,f):⇔
  ∀p∈ℕ,s∈Stream,e∈Environment,g∈TFormula:
    (⊢ next(TA(X,b1,b2,f))→*(n,p,s,e) g) ⇒
      let c = (e,{(Y,s(e(Y))) | Y ∈ dom(e)}) :
      let p0 = p+n, p1 = b1(c), p2 = b2(c) :
      (
        n = 1 ∧ (p1 = ∞ ∨ p1 >∞ p2) ∧ g=done(true)
      )
      ∨
      (
        n ≥ 1 ∧ p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p0 ≤ p1 ∧ g=next(TA0(X,p1,p2,f))
      )
      ∨
      (
        n ≥ 1 ∧ p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p0 > p1 ∧
        (
          (∃b∈Bool: g=done(b)) ∨
          (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n ≤∞ p2) ∧
             forallInstances(X,p,p0,p1,p2,f,s,e,gs) ∧
             g = next(TA1(X,p2,f,gs)))
        )
      )

forallInstances ⊆
  Variable × ℕ × ℕ × ℕ × ℕ∞ × TFormula ×
  Stream × Environment × ℙ(TInstance):
forallInstances(X,p,p0,p1,p2,f,s,e,gs) :⇔
  ∀t∈ℕ,g∈TFormula,c0∈Context: (t,g,c0)∈gs ⇒
    (∀t1∈ℕ,g1∈TFormula,c1∈Context:
      (t1,g1,c1)∈gs ∧ t=t1 ⇒ (t,g,c0)=(t1,g1,c1)) ∧
    (∃gc∈TFormulaCore: g=next(gc)) ∧
    c0.1=e[X ↦ t] ∧ c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X} ∧
    p1 ≤ t ≤∞ min∞(p0-1,p2) ∧
    ⊢ f →*(p0-max(p,t),max(p,t),s,c0.1) g

Proof
-----

Let X∈Variable,b1∈BoundValue,b2∈BoundValue,f∈TFormulaCore be arbitrary fixed.
We prove

  ∀n∈ℕ: n≥1 ⇒ forall(n,X,b1,b2,next(f))

by induction on n≥1.
```

```
Let n∈ℕ be arbitrary but fixed and assume

(0) n≥1.

Induction Base
--------------

We show

[1]  forall(1,X,b1,b2,next(f))

i.e. by the definition of "forall" for arbitrary but fixed
p∈ℕ, s∈Stream, e∈Environment, g∈TFormula under the assumptions

(1) ⊢ next(TA(X,b1,b2,f))→*(1,p,s,e) g
(2) c := (e,{(Y,s(e(Y))) | Y ∈ dom(e)})
(3) p0 := p+1
(4) p1 := b1(c)
(5) p2 := b2(c)

the goal

[2]   (
        (p1 = ∞ ∨ p1 >∞ p2) ∧ g=done(true)
      )
      ∨
      (
        p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p0 ≤ p1 ∧ g=next(TA0(X,p1,p2,next(f)))
      )
      ∨
      (
        p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p0 > p1 ∧
        (
          (∃b∈Bool: g=done(b)) ∨
          (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n ≤∞ p2) ∧
             forallInstances(X,p,p0,p1,p2,next(f),s,e,gs) ∧
             g = next(TA1(X,p2,next(f),gs)))
        )
      )

From (1), (2) and the rules for →*, we know for some Ft'∈TFormula

(6) ⊢ next(TA(X,b1,b2,next(f))) →(p,s↓p,s(p),c) Ft'
(7) ⊢ Ft' →*(0,p+1,s,e) g

From (6), (7) and the rules for →*, we know

(8) ⊢ next(TA(X,b1,b2,next(f))) →(p,s↓p,s(p),c) g

From (4),(5),(8) and the rules for →, we have two cases.

Case 1
------
```

```
(20) p1 = ∞ ∨ p1 >∞ p2
(21) g = done(true)

From (20) and (21), we have [2].

Case 2
------
(30) p1 ≠ ∞ ∧ p1 ≤∞ p2
(31) ⊢ next(TA0(X,p1,p2,next(f))) →(p,s↓p,s(p),c) g

We proceed by case distinction.

Case 2.1
--------
(40) p0 ≤∞ p1

From (30) and (40), to show [2] it suffices to show

[2.a] g=next(TA0(X,p1,p2,next(f)))

From (31) and the fact that the rule system for → is deterministic,
to show [2.a], it suffices to show

[2.b] ⊢ next(TA0(X,p1,p2,next(f))) →(p,s↓p,s(p),c) next(TA0(X,p1,p2,next(f)))

which holds from (3), (40) and the rules for →.

Case 2.2
--------
(70) p0 >∞ p1

From (30) and (70), to show [2] it suffices to show

[2.a] (∃b∈Bool: g=done(b)) ∨
      (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+1 ≤∞ p2) ∧
         forallInstances(X,p,p0,p1,p2,next(f),s,e,gs) ∧
         g = next(TA1(X,p2,next(f),gs)))

We define

(72) fs := {(px,next(f),(c.1[X↦px],c.2[X↦s↓p(px+p-|s↓p|)])) |
              p1 ≤ px <∞ min∞(p,p2+∞1)}

From (3), (31), (70), (72), and the rules for →, we know

(73) ⊢ next(TA1(X,p2,next(f),fs)) →(p,s↓p,s(p),c) g

From (72), we know with |s↓p|=p

(74) fs = {(px,next(f),(c.1[X↦px],c.2[X↦s↓p(px)])) | p1 ≤ px <∞ min∞(p,p2+∞1)}

and thus

(74') fs = {(px,next(f),(c.1[X↦px],c.2[X↦s(px)])) | p1 ≤ px <∞ min∞(p,p2+∞1)}
```

To show [2.a], we assume

```
(75) ¬(∃b∈Bool: g=done(b))
```

and show

```
[2.b] ∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+1 ≤∞ p2) ∧
        forallInstances(X,p,p0,p1,p2,next(f),s,e,gs) ∧
        g = next(TA1(X,p2,next(f),gs))
```

From (73), (75), and the rules for →, we know

```
(76) fs0 := if p >∞ p2 then fs else fs ∪ {(p,next(f),(c.1[X↦p],c.2[X↦s(p)]))}
(77) ¬∃t∈ℕ,g∈TFormula,c∈Context: (t,g,c)∈fs0 ∧ ⊢ g →(p,s↓p,s(p),c) done(false)
(78) fs1 := { (t,next(fc),c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈fs0 ∧
               ⊢ g →(p,s↓p,s(p),c) next(fc) }
(79) ¬(fs1 = ∅ ∧ p ≥∞ p2)
(80) g = next(TA1(X,p2,next(f),fs1))
```

From (80), to show [2.b], it suffices to show

```
[2.b.1] fs1 ≠ ∅ ∨ p+1 ≤∞ p2
[2.b.2] forallInstances(X,p,p0,p1,p2,next(f),s,e,fs1)
```

From p ∈ ℕ and (79), we have [2.b.1].

To show [2.b.2], we proceed by case distinction.

```
Case 2.2.1
----------
(100) p >∞ p2
```

From (76) and (100), we know

```
(101) fs0 = fs
```

From (74), (78), (101), we know

```
(102) fs1 = { (t,next(fc),(c.1[X↦t],c.2[X↦s↓p(t)])) |
               p1 ≤ t <∞ min∞(p,p2+∞1) ∧
               ⊢ next(f) →(p,s↓p,s(p),(c.1[X↦t],c.2[X↦s↓p(t)])) next(fc) }
```

To show [2.b.2], from the definition of "forallInstances", we have to show
for arbitrary but fixed t∈ℕ,g∈TFormula,c0∈Context such that

```
(120) (t,g,c0)∈fs1
```

the following:

```
[2.b.2.1] ∀t1∈ℕ,g1∈TFormula,c1∈Context:
              (t1,g1,c1)∈fs1 ∧ t=t1 ⇒ (t,g,c0)=(t1,g1,c1)
[2.b.2.2] ∃gc∈TFormulaCore: g=next(gc)
[2.b.2.3] c0.1=e[X ↦ t]
```

[2.b.2.4] c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X}
[2.b.2.5] p1 ≤ t ≤∞ min∞(p0-1,p2)
[2.b.2.6] ⊢ next(f) →*(p0-max(p,t),max(p,t),s,c0.1) g

From (102) and the fact that the rule system for → is deterministic,
we have [2.b.2.1].

From (102) and (120), we have for some fc∈TFormulaCore

(121) g=next(fc)
(122) c0=(c.1[X↦t],c.2[X↦s↓p(t)])
(123) p1 ≤ t <∞ min∞(p,p2+∞1)
(124) ⊢ next(f) →(p,s↓p,s(p),c0) next(fc)

From (121), we have [2.b.2.2].

From (122) and (2), we have [2.b.2.3].

To show [2.b.2.5], from (3), it suffices to show

[2.b.2.5.1] p1 ≤ t
[2.b.2.5.2] t ≤ p
[2.b.2.5.3] t ≤∞ p2

which all three follow from (123).

We now show [2.b.2.4]. From (122), we know

(125) c0.1 = c.1[X↦t]
(126) c0.2 = c.2[X↦s↓p(t)]

From (123), we know

(127) t < p

From (126) and (127), we have

(128) c0.2 = c.2[X↦s(t)]

From (125) and (128), to show [2.b.2.4], it suffices to show

[2.b.2.4.a] c.2[X↦s(t)]={(Y,s(c.1[X↦t](Y))) | Y ∈ dom(e) ∨ Y = X}

For this it suffices to show for arbitrary Y with Y ∈ dom(e) ∨ Y = X

[2.b.2.4.b] c.2[X↦s(t)](Y) = s(c.1[X↦t](Y))

Case Y=X:

We have

(130) c.2[X↦s(t)](Y) = s(t)
(131) s(c.1[X↦t](Y)) = s(t)

and thus [2.b.2.4.b].

Case Y ≠ X:

We have

(132) Y ∈ dom(e)
(133) c.2[X↦s(t)](Y) = c.2(Y)
(134) s(c.1[X↦t](Y)) = s(c.1(Y))

From (2) and (132), we have

(135) c.1 = e
(136) c.2(Y) = s(e(Y))

From (133), (134), (135), (136), we have [2.b.2.4.b].

To show [2.b.2.6], by (3), it suffices to show

[2.b.2.6.a] ⊢ next(f) →*(p0-max(p,t),max(p,t),s,c0.1) g

From (123), we know

(140) max(p,t) = p

From (3) and (140), it suffices to show

[2.b.2.6.b] ⊢ next(f) →*(1,p,s,c0.1) g

From (2), (125), (128), we know

(141) c0 = (c0.1, {(Y,s(c0.1(Y))) | Y ∈ dom(c0.1)})

From (141) and the definition of →*, it suffices to show

[2.b.2.6.c] ⊢ next(f) →(p,s↓p,s(p),c0) g

which follows from (121) and (124).

Case 2.2.2
----
(200) p ≤∞ p2

To show [2.b.2], from the definition of "forallInstances", we have to show
for arbitrary but fixed t∈ℕ,g∈TFormula,c0∈Context such that

(201) (t,g,c0)∈fs1

the following:

[2.b.2.1] ∀t1∈ℕ,g1∈TFormula,c1∈Context:
(t1,g1,c1)∈fs1 ∧ t=t1 ⇒ (t,g,c0)=(t1,g1,c1)
[2.b.2.2] ∃gc∈TFormulaCore: g=next(gc)
[2.b.2.3] c0.1=e[X ↦ t]

[2.b.2.4] c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X}
[2.b.2.5] p1 ≤ t ≤∞ min∞(p0-1,p2)
[2.b.2.6] ⊢ next(f) →*(p0-max(p,t),max(p,t),s,c0.1) g

We define

(202) c1 := (c.1[X↦p],c.2[X↦s(p)])

From (76), (200), (202), we know

(203) fs0 = fs ∪ {(p,next(f),c1)}

From (78) and (203), we know

```
(204) fs1 = { (t,next(fc),c) ∈ TInstance |
                (∃g∈TFormula: (t,g,c)∈fs ∧
                   ⊢ g →(p,s↓p,s(p),c) next(fc)) ∨
                (t = p ∧ c = c1 ∧ ⊢ next(f) →(p,s↓p,s(p),c1) next(fc)) }
```

From (74’), (204), and the fact that the rule system is deterministic,
we have [2.b.2.1].

From (201) and (204), we have [2.b.2.2].

It thus remains to show [2.b.2.3-6].

From (201), (202) and (204) we have two cases:

Case 2.2.2.1

There exists some fc∈TFormulaCore such that
(220) t=p
(221) g=next(fc)
(222) ⊢ next(f) →(p,s↓p,s(p),c0) next(fc)
(223) c0.1=c.1[X↦p]
(224) c0.2=c.2[X↦s(p)]

From (2), (223), (224), we have [2.b.2.3].

From (2), (222), (223), (224), we have [2.b.2.4].

From (3) and (70) and (220), we have

(230) p1 ≤∞ t

From (200) and (220), we have

(231) t ≤∞ p2

From (3) and (220), we have

(232) t <∞ p0

From (230), (231), (232), we have [2.b.2.5].

To show [2.b.2.6], from (3) and (220), it suffices to show

[2.b.2.6.a] ⊢ next(f) →*(1,p,s,c0.1) g

From the definition of →*, (2), (223), (224), it suffices to show

[2.b.2.6.b] ⊢ next(f) →(p,s↓p,s(p),c0) g

which follows from (221) and (222).

Case 2.2.2.2

There exist some fc∈TFormulaCore and g0∈TFormula such that
(240) g = next(fc)
(241) (t,g0,c0)∈fs
(242) ⊢ g0 →(p,s↓p,s(p),c0) g

From (74') and (241), we know

(243) g0 = next(f)
(244) c0.1 = c.1[X ↦ t]
(245) c0.2 = c.2[X ↦ s(t)]
(246) p1 ≤ t
(247) t < p
(248) t ≤∞ p2

From (2) and (244), we know [2.b.2.3].

From (2), (244) and (245), we know [2.b.2.4].

From (3), (246), (247), and (248), we know [2.b.2.5].

From (247), we know

(249) max(p,t) = p

From (3) and (249), to show [2.b.2.6], we have to show

[2.b.2.6.a] ⊢ next(f) →*(1,p,s,c0.1) g

From the definition of →*, (2), (244), (245), it suffices to show

[2.b.2.6.b] ⊢ next(f) →*(p,s↓p,s(p),c0) g

which follows from (242) and (243).

Induction Step

We assume

(1) forall(n,X,b1,b2,next(f))

and show

```
[1] forall(n+1,X,b1,b2,next(f))
```

i.e. by the definition of "forall" for arbitrary but fixed
p∈ℕ, s∈Stream, e∈Environment, g∈TFormula, c∈Context, p1∈ℕ∞, p2∈ℕ∞
under the assumptions

```
(2) ⊢ next(TA(X,b1,b2,f))→*(n+1,p,s,e) g
(3) c = (e,{(Y,s(e(Y))) | Y ∈ dom(e)})
(4) p1 = b1(c)
(5) p2 = b2(c)
```

the goal

```
[2]   (
        n+1 = 1 ∧ (p1 = ∞ ∨ p1 >∞ p2) ∧ g=done(true)
      )
      ∨
      (
        n+1 ≥ 1 ∧ p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p+n+1 ≤ p1 ∧
        g=next(TA0(X,p1,p2,next(f)))
      )
      ∨
      (
        n+1 ≥ 1 ∧ p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p+n+1 > p1 ∧
        (
          (∃b∈Bool: g=done(b)) ∨
          (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n+1 ≤∞ p2) ∧
             forallInstances(X,p,p+n+1,p1,p2,next(f),s,e,gs) ∧
             g = next(TA1(X,p2,next(f),gs)))
        )
      )
```

which with (0) can be simplified to

```
[3]   (
        p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p+n+1 ≤ p1 ∧ g=next(TA0(X,p1,p2,next(f)))
      )
      ∨
      (
        p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p+n+1 > p1 ∧
        (
          (∃b∈Bool: g=done(b)) ∨
          (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n+1 ≤∞ p2) ∧
             forallInstances(X,p,p+n+1,p1,p2,next(f),s,e,gs) ∧
             g = next(TA1(X,p2,next(f),gs)))
        )
      )
```

From (2) and Lemma 2 "Equivalence of Left- and Right-Recursive Definitions
of n-Step Reductions", we know

```
(6) ⊢ next(TA(X,b1,b2,f))→l*(n+1,p,s,e) g
```

From (6) and the definition of →l*, we know for some Ft'∈TFormula

```
(7) ⊢ next(TA(X,b1,b2,next(f))) →l*(n,p,s,e) Ft'
(8) ⊢ Ft' →(p+n,s↓(p+n),s(p+n),c) g
```

From (7) and Lemma 2 "Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions", we know

```
(9) ⊢ next(TA(X,b1,b2,next(f))) →*(n,p,s,e) Ft'
```

From (1),(3),(4),(5),(9), and the definition of "forall", we know

```
(10)  (
        n = 1 ∧ (p1 = ∞ ∨ p1 >∞ p2) ∧ Ft'=done(true)
      )
      ∨
      (
        n ≥ 1 ∧ p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p+n ≤ p1 ∧ Ft'=next(TA0(X,p1,p2,next(f)))
      )
      ∨
      (
        n ≥ 1 ∧ p1 ≠ ∞ ∧ p1 ≤∞ p2 ∧ p+n > p1 ∧
        (
          (∃b∈Bool: Ft'=done(b)) ∨
          (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n ≤∞ p2) ∧
             forallInstances(X,p,p+n,p1,p2,next(f),s,e,gs) ∧
             Ft' = next(TA1(X,p2,next(f),gs)))
        )
      )
```

From (10), we proceed by case distinction.

Case 1

```
(20) n = 1
(21) p1 = ∞ ∨ p1 >∞ p2
(22) Ft'=done(true)
```

By the definition of →, (22) contradicts (8).

Case 2

```
(50) n ≥ 1
(51) p1 ≠ ∞
(52) p1 ≤∞ p2
(53) p+n ≤ p1
(54) Ft'=next(TA0(X,p1,p2,next(f)))
```

By the definition of →, from (8) and (54), we have two subcases.

Subcase 2.1

```
(60) p+n < p1
```

```
(61) g = Ft'

From (60), we know

(62) p+n+1 ≤ p1

From (51), (52), (54), (61), (62), we have [3] (first disjunct).

Subcase 2.2
-----------
There exists fs such that
(70) p+n ≥ p1
(71) fs = {(px,next(f),(c.1[X↦px],c.2[X↦s↓(p+n)(px+p+n-|s↓(p+n)|)])) |
             p1 ≤ px <∞ min∞(p+n,p2+∞1)}
(72) ⊢ next(TA1(X,p2,next(f),fs)) →(p+n,s↓(p+n),s(p+n),c) g

From (71), we know

(73) fs = {(px,next(f),(c.1[X↦px],c.2[X↦s(px)])) | p1 ≤ px <∞ min∞(p+n,p2+∞1)}

From (51), (52), (70), to show [3], it suffices to show

[4]        (∃b∈Bool: g=done(b)) ∨
           (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n+1 ≤∞ p2) ∧
              forallInstances(X,p,p+n+1,p1,p2,next(f),s,e,gs) ∧
              g = next(TA1(X,p2,next(f),gs)))

To show [4], we assume

(74) ∀b∈Bool: g≠done(b)

and show

[5] (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n+1 ≤∞ p2) ∧
       forallInstances(X,p,p+n+1,p1,p2,next(f),s,e,gs) ∧
       g = next(TA1(X,p2,next(f),gs)))

From (72) and (74), we know by the definition of → for some fs0 and fs1

(75) fs0 = if p+n >∞ p2 then fs else
              fs ∪ {(p+n,next(f),(c.1[X↦p+n],c.2[X↦s(p+n)]))}
(76) ¬∃t∈ℕ,g∈TFormula,c∈Context: (t,g,c)∈fs0 ∧
        ⊢ g →(p+n,s↓(p+n),s(p+n),c) done(false)
(77) fs1 = { (t,next(fc),c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈fs0 ∧
              ⊢ g →(p+n,s↓(p+n),s(p+n),c) next(fc) }
(78) ¬(fs1 = ∅ ∧ p+n ≥∞ p2)
(79) g = next(TA1(X,p2,next(f),fs1))

To show [5], it suffices to show (gs:=fs1)

[5.1] fs1 ≠ ∅ ∨ p+n+1 ≤∞ p2
[5.2] forallInstances(X,p,p+n+1,p1,p2,next(f),s,e,fs1)
[5.3] g = next(TA1(X,p2,next(f),fs1))
```

To show [5.1], we assume

(80) fs1 = ∅

and show

[5.1.a] p+n+1 ≤∞ p2

From (78) and (80), we know

(81) p+n <∞ p2

From (81), we know [5.1.a].

From (79), we know [5.3].

It remains to show [5.2], i.e., by the definition of "forallInstances",
for arbitrary t∈ℕ,g0∈TFormula,c0∈Context, that under the assumption

(82) (t,g0,c0)∈fs1

the following holds:

```
[5.2.1] (∀t1∈ℕ,g1∈TFormula,c1∈Context:
          (t1,g1,c1)∈fs1 ∧ t=t1 ⇒ (t,g0,c0)=(t1,g1,c1)
[5.2.2] ∃gc∈TFormulaCore: g0=next(gc)
[5.2.3] c0.1=e[X ↦ t]
[5.2.4] c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X}
[5.2.5] p1 ≤ t
[5.2.6] t ≤ p+n
[5.2.7] t ≤∞ p2
[5.2.8] ⊢ next(f) →*(p+n+1-max(p,t),max(p,t),s,c0.1) g0
```

From (77) and (82), we know for some fc0∈TFormulaCore, g1∈TFormula

```
(83) g0=next(fc0)
(84) (t,g1,c0)∈fs0
(85) ⊢ g1 →(p+n,s↓(p+n),s(p+n),c0) g0
```

From (53) and (70), we know

(86) p+n = p1

From (73) and (86), we know

(87) fs = ∅

From (84), we know

(88) fs0 ≠ ∅

From (75), (87), and (88), we know

(89) fs0 = {(p+n,next(f),(c.1[X↦p+n],c.2[X↦s(p+n)]))}

From (84) and (89), we know

(100) t = p+n
(101) g1 = next(f)
(102) c0.1 = c.1[X↦p+n]
(103) c0.2 = c.2[X↦s(p+n)]

From (77), (89), and the fact that the rule system is deterministic,
we know [5.2.1].

From (83), we know [5.2.2].

From (3), (100), (102), and (103) we know [5.2.3] and [5.2.4].

From (86) and (100), we know [5.2.5] and [5.2.6].

From (52), (86), and (100), we know [5.2.7].

From (0) and (100), we know

(104) max(p,t) = t

From (100), (101) and (104), to show [5.2.8], it suffices to show

[5.2.8.a] ⊢ g1 →*(1,p+n,s,c0.1) g0

From the definition of →, (85), (3), (102), and (103), we have [5.2.8.a].

Case 3
------
(200) n ≥ 1
(201) p1 ≠ ∞
(202) p1 ≤∞ p2
(203) p+n > p1
(204) (∃b∈Bool: Ft'=done(b)) ∨
      (∃gs∈ℙ(TInstance): (gs ≠ ∅ ∨ p+n ≤∞ p2) ∧
         forallInstances(X,p,p+n,p1,p2,next(f),s,e,gs) ∧
         Ft' = next(TA1(X,p2,next(f),gs)))

From (204), we proceed by case distinction.

Subcase 3.1

We have some b∈Bool such that
(210) Ft'=done(b)

By the definition of →, (210) contradicts (8).

Subcase 3.2

We have some gs∈ℙ(TInstance) such that
(301) gs ≠ ∅ ∨ p+n ≤∞ p2
(302) forallInstances(X,p,p+n,p1,p2,next(f),s,e,gs)

```
(303) Ft' = next(TA1(X,p2,next(f),gs))

We define

(304) fs0 = if p+n >∞ p2 then gs else gs ∪
              {(p+n,next(f),(c.1[X↦p+n],c.2[X↦s(p+n)]))}

From (8), (303), and (304), we have by the definition of → three cases.

Subsubcase 3.2.1
----------------
We have some t0∈ℕ,g0∈TFormula,c0∈Context such that
(310) (t0,g0,c0)∈fs0
(311) ⊢ g0 →(p+n,s↓(p+n),s(p+n),c) done(false)
(312) g = done(false)

From (201), (202), (203), and (312), we have [3] (second disjunct, first case).

Subsubcase 3.2.2
----------------
We have some fs1 such that
(320) ¬∃t∈ℕ,g∈TFormula,c∈Context: (t,g,c)∈gs ∧
         ⊢ g →(p+n,s↓(p+n),s(p+n),c) done(false)
(321) fs1 = { (t,next(fc),c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈fs0 ∧
        ⊢ g →(p+n,s↓(p+n),s(p+n),c) next(fc) }
(322) fs1 = ∅
(323) p+n ≥∞ p2
(324) g = done(true)

From (201), (202), (203), and (324), we have [3] (second disjunct, first case).

Subsubcase 3.2.3
----------------
We have some fs1 such that
(330) ¬∃t∈ℕ,g∈TFormula,c∈Context: (t,g,c)∈gs ∧
         ⊢ g →(p+n,s↓(p+n),s(p+n),c) done(false)
(331) fs1 = { (t,next(fc),c) ∈ TInstance | ∃g∈TFormula: (t,g,c)∈fs0 ∧
        ⊢ g →(p+n,s↓(p+n),s(p+n),c) next(fc) }
(332) ¬(fs1 = ∅ ∧ p+n ≥∞ p2)
(333) g = next(TA1(X,p2,next(f),fs1))

From (201), (202), (203), and (333), to show [3], it suffices to show
(second disjunct, second case, gs:=fs1):

[3.1] fs1 ≠ ∅ ∨ p+n+1 ≤∞ p2
[3.2] forallInstances(X,p,p+n+1,p1,p2,next(f),s,e,fs1)
[3.3] g = next(TA1(X,p2,next(f),fs1)))

From (332), we have [3.1].
From (333), we have [3.3].

To show [3.2], by the definition of "forallInstances", we take
arbitrary t,g0,c0 such that
```

```
(340) (t,g0,c0)∈fs1

and show

[3.2.1] ∀t1∈ℕ,g1∈TFormula,c1∈Context:
          (t1,g1,c1)∈fs1 ∧ t=t1 ⇒ (t,g0,c0)=(t1,g1,c1)
[3.2.2] ∃gc∈TFormulaCore: g0=next(gc)
[3.2.3] c0.1=e[X ↦ t]
[3.2.4] c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X}
[3.2.5] p1 ≤ t
[3.2.6] t ≤ p+n
[3.2.7] t ≤∞ p2
[3.2.8] ⊢ next(f) →*(p+n+1-max(p,t),max(p,t),s,c0.1) g0

From (331) and (340), we have some fc0∈TFormulaCore, g1∈TFormula with

(341) g0 = next(fc0)
(342) (t,g1,c0)∈fs0
(343) ⊢ g1 →(p+n,s↓(p+n),s(p+n),c0) next(fc0)

From (341), we have [3.2.2].

It remains to show [3.2.1] and [3.2.3-8].

From (302) and the definition of "forallInstances", we know

(344)

  ∀t∈ℕ,g∈TFormula,c0∈Context: (t,g,c0)∈gs ⇒
    (∀t1∈ℕ,g1∈TFormula,c1∈Context:
      (t1,g1,c1)∈gs ∧ t=t1 ⇒ (t,g,c0)=(t1,g1,c1)) ∧
    (∃gc∈TFormulaCore: g=next(gc)) ∧
    c0.1=e[X ↦ t] ∧ c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X} ∧
    p1 ≤ t ≤∞ min∞(p+n-1,p2) ∧
    ⊢ next(f) →*(p+n-max(p,t),max(p,t),s,c0.1) g

We proceed by case distinction.

Subsubsubcase 3.2.3.1
---------------------
(350) p+n >∞ p2

From (304) and (350), we have

(351) fs0 = gs

From (342), (351), and (344), we know for some gc0∈TFormulaCore

(352) ∀t2∈ℕ,g2∈TFormula,c2∈Context:
        (t2,g2,c2)∈gs ∧ t=t2 ⇒ (t,g1,c0)=(t2,g2,c2)
(353) g1=next(gc0)
(354) c0.1=e[X ↦ t]
(355) c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X}
(356) p1 ≤ t
```

```
(357) t < p+n
(358) t ≤∞ p2
(359) ⊢ next(f) →*(p+n-max(p,t),max(p,t),s,c0.1) g1
```

From (331), (351), (352), and the fact that the rule system for → is deterministic, we know [3.2.1].

From (354), we know [3.2.3].
From (355), we know [3.2.4]
From (356), we know [3.2.5].
From (357), we know [3.2.6].
From (358), we know [3.2.7].

From (359) and Lemma 2 "Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions", we know

```
(360) ⊢ next(f) →l*(p+n-max(p,t),max(p,t),s,c0.1) g1
```

From (343) and (360), we know by the definition of →*

```
(361) ⊢ next(f) →l*(p+n+1-max(p,t),max(p,t),s,c0.1) next(fc0)
```

From (361) and Lemma 2 "Equivalence of Left- and Right-Recursive Definitions of n-Step Reductions", we know

```
(362) ⊢ next(f) →*(p+n+1-max(p,t),max(p,t),s,c0.1) next(fc0)
```

From (341) and (362), we know [3.2.8].

Subsubsubcase 3.2.3.2

```
(400) p+n ≤∞ p2
```

From (304) and (400), we know

```
(401) fs0 = gs ∪ {(p+n,next(f),(c.1[X↦p+n],c.2[X↦s(p+n)]))}
```

From (342) and (401), we have two cases.

Subsubsubsubcase 3.2.3.2.1

```
(410) (t,g1,c0)∈gs
```

From (344) and (410), we know for some gc0∈TFormulaCore

```
(412) ∀t2∈ℕ,g2∈TFormula,c2∈Context:
        (t2,g2,c2)∈gs ∧ t=t2 ⇒ (t,g1,c0)=(t2,g2,c2)
(413) g1=next(gc0)
(414) c0.1=e[X ↦ t]
(415) c0.2={(Y,s(c0.1(Y))) | Y ∈ dom(e) ∨ Y = X}
(416) p1 ≤ t
(417) t < p+n
(418) t ≤∞ p2
(419) ⊢ next(f) →*(p+n-max(p,t),max(p,t),s,c0.1) g1
```

To show [3.2.1], we take arbitrary t2∈ℕ,g2∈TFormula,c2∈Context
for which we assume

(420) (t2,g2,c2)∈fs1
(421) t=t2

and show

[3.2.1.a] (t,g0,c0)=(t2,g2,c2)

To show [3.2.1.a], from (421), it suffices to show

[3.2.1.a.1] g0 = g2
[3.2.1.a.2] c0 = c2

From (331) and (420), we have some g3∈TFormula, fc3∈TFormulaCore such that

(422) g2=next(fc3)
(423) (t,g3,c1)∈fs0
(424) ⊢ g3 →(p+n,s↓(p+n),s(p+n),c1) g2

From (401), (417), and (423), we know

(425) (t,g3,c1)∈gs

From (410), (412), and (425), we have

(426) g1 = g3
(427) c0 = c1

From (341), (343), (426), and (427), we have

(428) ⊢ g3 →(p+n,s↓(p+n),s(p+n),c1) g0

From (424), (428), and the fact that the rule system for → is deterministic,
we have [3.2.1.a.1].

From (427), we have [3.2.1.a.2].

From (414), we know [3.2.3].
From (415), we know [3.2.4]
From (416), we know [3.2.5].
From (417), we know [3.2.6].
From (418), we know [3.2.7].

From (419) and Lemma 2 "Equivalence of Left- and Right-Recursive Definitions
of n-Step Reductions", we know

(450) ⊢ next(f) →l*(p+n-max(p,t),max(p,t),s,c0.1) g1

From (343) and (450), we know by the definition of →*

(451) ⊢ next(f) →l*(p+n+1-max(p,t),max(p,t),s,c0.1) next(fc0)

From (451) and Lemma 2 "Equivalence of Left- and Right-Recursive Definitions
of n-Step Reductions", we know

(452) ⊢ next(f) →*(p+n+1-max(p,t),max(p,t),s,c0.1) next(fc0)

From (341) and (452), we know [3.2.8].

Subsubsubsubcase 3.2.3.2.2
----
(500) t=p+n
(501) g1=next(f)
(502) c0.1=c.1[X↦p+n]
(503) c0.2=c.2[X↦s(p+n)]

To show [3.2.1], we take arbitrary t2∈ℕ,g2∈TFormula,c2∈Context
for which we assume

(520) (t2,g2,c2)∈fs1
(521) t=t2

and show

[3.2.1.a] (t,g0,c0)=(t2,g2,c2)

To show [3.2.1.a], from (521), it suffices to show

[3.2.1.a.1] g0 = g2
[3.2.1.a.2] c0 = c2

From (331) and (520), we have some g3∈TFormula, fc3∈TFormulaCore such that

(522) g2=next(fc3)
(523) (t,g3,c2)∈fs0
(524) ⊢ g3 →(p+n,s↓(p+n),s(p+n),c2) g2

From (344) and (500), we know

(525) (t,g3,c2)∉gs

From (401), (523), (525), we know

(526) g3 = next(f)
(527) c2.1 = c.1[X↦p+n]
(528) c2.2 = c.2[X↦s(p+n)]

From (341), (343), (501), (524), (527), (528), we know

(529) ⊢ g3 →(p+n,s↓(p+n),s(p+n),c2) g0

From (524), (529), and the fact that the rule system for → is deterministic,
we have [3.2.1.a.1].

From (502), (503), (527), (528), we know [3.2.1.a.2].

From (2), (500), (502), (503), we know [3.2.3] and [3.2.4].
From (203) and (500), we know [3.2.5].
From (500), we know [3.2.6].
From (400) and (500), we know [3.2.7].

From (500), to show [3.2.8], it suffices to show

[3.2.8.a] ⊢ next(f) →*(1,p+n,s,c0.1) g0

From (526), (529), [3.2.1.a.2], and the definition of →*,
we know [3.2.8.a].

Q.E.D.