Connecting Fixpoints of Computations with Strict Progress

Walter Guttmann

Department of Computer Science and Software Engineering,
University of Canterbury, New Zealand
walter.guttmann@canterbury.ac.nz

Abstract. We study the semantics of recursion for computations that make strict progress. The underlying unified computation model has an abstract notion of progress, which instantiates in ways such as longer traces, passing of real time, or counting the number of steps. Recursion is given by least fixpoints in a unified approximation order. Other time-based models define the semantics of recursion by greatest fixpoints in the implication order. We give sufficient criteria for when least fixpoints in the approximation order coincide with greatest fixpoints in the implication order.

1 Introduction

A recursive computation has the form $x = f(x)$, where the function f specifies what happens in one unfolding step of the recursion in terms of x, which captures recursive invocations. For example, unfolding the loop $\mathsf{while}\ b\ \mathsf{do}\ a$ results in the recursion $\mathsf{while}\ b\ \mathsf{do}\ a = \mathsf{if}\ b\ \mathsf{then}\ a\ ;\ \mathsf{while}\ b\ \mathsf{do}\ a\ \mathsf{else\ skip}$. Hence $\mathsf{while}\ b\ \mathsf{do}\ a$ is a solution of the equation $x = \mathsf{if}\ b\ \mathsf{then}\ a\ ;\ x\ \mathsf{else\ skip}$ using a suitable semantics for the conditional. Solutions of the equation $x = f(x)$ are fixpoints of the function f. It is therefore not surprising that fixpoints have been widely used to define the semantics of recursive computations [4, 5, 12, 16, 32, 37, 39].

In general, a function may have several fixpoints. For example, every x is a fixpoint of the identity function. The corresponding recursion equation $x = x$ may seem contrived, but it gives the semantics of the endless loop $\mathsf{while\ true\ do\ skip}$ since $\mathsf{if\ true\ then\ skip}\ ;\ x\ \mathsf{else\ skip} = \mathsf{skip}\ ;\ x = x$ in many computation models. In this and similar cases, the question arises which fixpoint to choose as the semantics of recursion. Different computation models give different answers.

A common solution is to define an order on the computations and choose the least or greatest fixpoint in this order. Two orders are relevant for computation models: the implication order and the approximation order [5]. Conceptually they serve different purposes. The implication order establishes when a program implements a specification and is typically concerned with the amount of non-determinism exhibited by computations. The approximation order deals with the semantics of recursion; successive unfoldings of a recursion yield better approximations to its semantics, which emerges as the limit. Confusion sometimes occurs because in some computation models the approximation order happens to coincide with the implication order or its converse (the refinement order). For example, in the Unifying Theories of Programming (UTP) recursion is defined by least fixpoints in the refinement order [28]. In general, however, a better approximation is not tied to either a decrease or an increase in non-determinism.

Justification of a particular choice of fixpoints for a proposed computation model is usually left implicit (though explicit connections have sometimes been made [8]). Specifically, we note that apparently different choices are made for computation models with different notions of progress. A dedicated approximation order (different from implication) has been used in Boolean-time models, which are models that distinguish between termination and non-termination but do not have a finer notion of elapsing time [4, 5]. Greatest fixpoints with respect to implication have been used in models that measure time using natural numbers or real numbers [24].

The aim of this paper is to provide additional confidence that the choices of fixpoints made for several computation models make sense. To this end, we study the relationship between the greatest fixpoint in the implication order and the least fixpoint in the approximation order. We do this in a unified computation model that supports an abstract notion of progress, developed in previous work [21]. Computations in this model are sequential and may be non-deterministic; single executions may be finite, aborting, incomplete or infinite. Progress can be measured in terms of Boolean-time, abstract time, real time, traces and other instances. The model supports a unified approximation order in addition to the implication order. It can therefore be used to investigate how extremal fixpoints in these orders are connected.

We give sufficient conditions for when least fixpoints in the approximation order coincide with greatest fixpoints in the implication order. We interpret these conditions for various notions of progress. The findings of this paper can be summarised as follows. The availability of real time in the model is not sufficient for the fixpoints to coincide even if strict progress is assumed, that is, if time cannot stand still. However, the two studied fixpoints coincide if a uniform lower bound can be given for the progress made in each unfolding of the recursion.

Knowing circumstances in which the implication order can be used to define recursion is helpful since it is arguably simpler than the approximation order. Moreover the implication order is fundamental for reasoning about the correctness of computations, so a direct connection with the approximation order facilitates this task.

Section 2 describes the setup for our study including basic algebraic structures and the unified computation model. The following sections comprise new results, which form the contributions of this paper. We outline the overall strategy in Section 3. Section 4 contains results that can be derived algebraically, based on complete lattices, semirings and an iteration operation. In particular, we prove upper bounds for greatest fixpoints in the implication order by capturing the excess over corresponding least fixpoints in the same order. Section 5 contains results that are specific to the unified computation model referring to various notions of progress. In particular, we give further bounds on the excess so that the greatest fixpoint in the implication order coincides with the least fixpoint in the approximation order. We discuss what the conditions that enable these bounds mean for different kinds of progress.

2 Basic Definitions

This section presents the algebraic structures and the unified computation model used in the remainder of the paper. The discussion is based on [21].

2.1 Algebraic Structures

A *bounded distributive lattice* is an algebraic structure $(S, \sqcup, \sqcap, \bot, \top)$ with the following axioms:

$$\begin{array}{rlrl}
x \sqcup (y \sqcup z) &= (x \sqcup y) \sqcup z & \qquad x \sqcap (y \sqcap z) &= (x \sqcap y) \sqcap z \\
x \sqcup y &= y \sqcup x & x \sqcap y &= y \sqcap x \\
x \sqcup x &= x & x \sqcap x &= x \\
\bot \sqcup x &= x & \top \sqcap x &= x \\
x \sqcup (y \sqcap z) &= (x \sqcup y) \sqcap (x \sqcup z) & x \sqcap (y \sqcup z) &= (x \sqcap y) \sqcup (x \sqcap z) \\
x \sqcup (x \sqcap y) &= x & x \sqcap (x \sqcup y) &= x
\end{array}$$

The *lattice order* $x \sqsubseteq y \Leftrightarrow x \sqcup y = y \Leftrightarrow x \sqcap y = x$ has least element $\bot$, greatest element $\top$, least upper bound operation $\sqcup$ and greatest lower bound operation $\sqcap$. The operations $\sqcup$ and $\sqcap$ are $\sqsubseteq$-isotone.

A bounded distributive lattice S is *MID-complete* if each $A \subseteq S$ has a meet $\bigsqcap A$ with the following axioms, including meet-infinite distributivity (MID):

$$\forall a \in A : \bigsqcap A \sqsubseteq a \qquad (\forall a \in A : x \sqsubseteq a) \Rightarrow x \sqsubseteq \bigsqcap A \qquad x \sqcup \bigsqcap A = \bigsqcap_{a \in A} x \sqcup a$$

We use $\bigsqcap_{a \in A} f(a) = \bigsqcap\{f(a) \mid a \in A\}$ and similar abbreviations.

An *idempotent semiring* without a right annihilator is an algebraic structure $(S, \sqcup, \cdot, \bot, 1)$ with the following axioms:

$$\begin{array}{rlrlrl}
x \sqcup (y \sqcup z) &= (x \sqcup y) \sqcup z & \quad x \cdot (y \cdot z) &= (x \cdot y) \cdot z & \quad x \cdot (y \sqcup z) &= (x \cdot y) \sqcup (x \cdot z) \\
x \sqcup y &= y \sqcup x & 1 \cdot x &= x & (x \sqcup y) \cdot z &= (x \cdot z) \sqcup (y \cdot z) \\
x \sqcup x &= x & x \cdot 1 &= x & \bot \cdot x &= \bot \\
\bot \sqcup x &= x & & & &
\end{array}$$

Note that $x \cdot \bot = \bot$ is not an axiom. The operation $\cdot$ is $\sqsubseteq$-isotone. We assume the operation $\cdot$ has higher precedence than $\sqcup$ and $\sqcap$ and abbreviate $x \cdot y$ as xy. Powers in semirings are defined by $x^0 = 1$ and $x^{i+1} = xx^i$ for $x \in S$ and $i \in \mathbb{N}$. It follows that $x^{i+1} = x^i x$.

A *lattice-ordered semiring* is an algebraic structure $(S, \sqcup, \cdot, \sqcap, \bot, 1, \top)$ such that $(S, \sqcup, \sqcap, \bot, \top)$ is a bounded distributive lattice and $(S, \sqcup, \cdot, \bot, 1)$ is an idempotent semiring without a right annihilator.

A *lattice-ordered itering* adds to a lattice-ordered semiring a unary operation $^\circ$ with higher precedence than $\cdot$ and satisfying the sumstar and productstar equations of [7] and two simulation axioms [17]:

$$\begin{aligned} (x \sqcup y)^\circ &= (x^\circ y)^\circ x^\circ & \qquad zx \sqsubseteq yy^\circ z \sqcup w &\Rightarrow zx^\circ \sqsubseteq y^\circ (z \sqcup wx^\circ) \\ (xy)^\circ &= 1 \sqcup x(yx)^\circ y & \qquad xz \sqsubseteq zy^\circ \sqcup w &\Rightarrow x^\circ z \sqsubseteq (z \sqcup x^\circ w) y^\circ \end{aligned}$$

The operation $^\circ$ is $\sqsubseteq$-isotone. Moreover the unfold property $x^\circ = 1 \sqcup xx^\circ$ and the related sumstar property $(x \sqcup y)^\circ = x^\circ (yx^\circ)^\circ$ follow. The itering operation generalises the Kleene star but has many other instances, for example, in omega algebras [6] and demonic refinement algebras [40]. In order to cover these instances, the simulation axioms of iterings generalise simpler simulation properties $zx \sqsubseteq yz \Rightarrow zx^\circ \sqsubseteq y^\circ z$ and $xz \sqsubseteq zy \Rightarrow x^\circ z \sqsubseteq zy^\circ$ suggested in [7] for Kleene algebras [31] and with applications in program transformation and data refinement [3, 6].

In many computation models, the operation $\sqcup$ represents non-deterministic choice, the operation $\cdot$ sequential composition, the operation $\sqcap$ conjunction, $\bot$ the computation with no executions, 1 the computation that does not change the state, $\top$ the computation with all executions, and $\sqsubseteq$ the implication order.

In particular, UTP designs [28] form instances of the above algebras, as do prescriptions, extended designs, conscriptions, extended conscriptions and other variants of designs discussed in the literature [10, 11, 18, 23]. Further works modelling computations based on lattices, semirings and related algebras include [2, 3, 6, 22, 29, 31, 33, 36, 40].

A (binary, homogeneous) *relation* $R : A \leftrightarrow A$ on a set $A \neq \emptyset$ is a set $R \subseteq A \times A$. Important constants are the empty relation $\mathsf{O} = \emptyset$, the identity relation $\mathsf{I} = \{(x, x) \mid x \in A\}$ and the universal relation $\mathsf{T} = A \times A$. The composition of relations Q and R is defined by $Q \cdot R = \{(x, z) \mid \exists y : (x, y) \in Q \wedge (y, z) \in R\}$. Relations on A form a lattice-ordered semiring $(2^{A \times A}, \cup, \cdot, \cap, \mathsf{O}, \mathsf{I}, \mathsf{T})$, which is MID-complete and satisfies right annihilation $Q\mathsf{O} = \mathsf{O}$. See [35] for further details about relations.

These definitions generalise to heterogeneous relations $R : A \leftrightarrow B$, which are sets $R \subseteq A \times B$ where A and B may differ. Heterogeneous relations form categories and their operations apply to arguments of suitable types [14, 34].

Let S be a set partially ordered by $\sqsubseteq$ and let $f : S \to S$. Provided they exist, the $\sqsubseteq$-least and $\sqsubseteq$-greatest *fixpoints* of f are denoted by μf and νf, respectively:

$$\begin{aligned} f(\mu f) &= \mu f & \qquad f(x) = x &\Rightarrow \mu f \sqsubseteq x \\ f(\nu f) &= \nu f & \qquad f(x) = x &\Rightarrow \nu f \sqsupseteq x \end{aligned}$$

For functions $f, g : S \to S$ such that g is $\sqsubseteq$-isotone and the involved fixpoints exist, the two rolling properties $\mu(g \circ f) = g(\mu(f \circ g))$ and $\nu(g \circ f) = g(\nu(f \circ g))$ follow [1, 9]. If S is a complete lattice and f is $\sqsubseteq$-isotone, both μf and νf exist by Tarski's fixpoint theorem [38].

2.2 A Unified Model for Computations with Progress

We discuss a model that represents computations by four relations and describes different notions of progress in a uniform way. Computations are sequential and may be non-deterministic; a computation comprises single executions, each of which may be finite, aborting, incomplete or infinite. We explain this model to the extent necessary for this paper; for further details and related work see [21].

The state space A of a computation is partitioned into two sets A_{fin} and A_∞ comprising the finite and infinite parts. The following are examples of such a separation, where D is the set of values that program variables can take:

A_{fin}	A_∞	model
D	$\{\infty\}$	Boolean time; computation terminates or does not terminate
$D \times \mathbb{N}$	$\{\infty\}$	abstract time; steps are counted
$D \times \mathbb{R}$	$\{\infty\}$	real time; a clock is used
D^+	D^ω	traces; finite and infinite sequences over D

Behavioural aspects of computations are modelled by relations on the state space. Two relations $\mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ and $\mathsf{F}_\infty : A_{\text{fin}} \leftrightarrow A_\infty$ describe finite progress and progress from the finite to the infinite. We assume that F is a preorder, that is, $\mathsf{I} \subseteq \mathsf{F} = \mathsf{F}^2$, and that the related transitivity property $\mathsf{F}\mathsf{F}_\infty = \mathsf{F}_\infty$ holds. For the above examples, these constants are:

$\mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$	A_{fin}	$\mathsf{F}_\infty : A_{\text{fin}} \leftrightarrow A_\infty$	A_∞	model
T	D	T	$\{\infty\}$	Boolean time
$\{((v,t),(v',t')) \mid t \le t'\}$	$D \times \mathbb{N}$	T	$\{\infty\}$	abstract time
$\{((v,t),(v',t')) \mid t \le t'\}$	$D \times \mathbb{R}$	T	$\{\infty\}$	real time
$\preceq$	D^+	$\preceq$	D^ω	traces

Traces use the prefix relation $\preceq$ on finite and infinite sequences.

In a computation, four relations N, P, Q and R are used to describe different kinds of execution. The relation R represents the finite executions, which terminate successfully. The relation P represents executions that abort due to an error. The relation N represents incomplete executions, which are unproductive and used in approximation. The relation Q represents actually infinite executions.

Formally, a computation $(N|P|Q|R)$ comprises relations $N, P, R : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ and $Q : A_{\text{fin}} \leftrightarrow A_\infty$ satisfying the *progress requirements* $N, P, R \subseteq \mathsf{F}$ and $Q \subseteq \mathsf{F}_\infty$ and the *closure requirements* $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_\infty \subseteq Q$. It is defined by the following matrix:

$$(N|P|Q|R) = \begin{pmatrix} \mathsf{I} & \mathsf{O} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} & \mathsf{I} & \mathsf{O} \\ N & P & Q & R \end{pmatrix}$$

In the trace model, the progress requirements correspond to UTP's healthiness condition R1, which specifies that traces can only get longer [28]. More details about this and the progress and closure requirements are given in [21].

The matrix definition of a computation helps to derive basic operations, which elaborate as follows.

– Non-deterministic choice is the componentwise union of the matrices:

$$(N_1|P_1|Q_1|R_1) \sqcup (N_2|P_2|Q_2|R_2) = (N_1 \cup N_2|P_1 \cup P_2|Q_1 \cup Q_2|R_1 \cup R_2)$$

– Sequential composition is given by the matrix product, where union and relational composition replace addition and multiplication:

$$(N_1|P_1|Q_1|R_1) \cdot (N_2|P_2|Q_2|R_2) = (N_1 \cup R_1N_2|P_1 \cup R_1P_2|Q_1 \cup R_1Q_2|R_1R_2)$$

– Conjunction is the componentwise intersection of the matrices:

$$(N_1|P_1|Q_1|R_1) \sqcap (N_2|P_2|Q_2|R_2) = (N_1 \cap N_2|P_1 \cap P_2|Q_1 \cap Q_2|R_1 \cap R_2)$$

– The implication order is the componentwise set inclusion order:

$$(N_1|P_1|Q_1|R_1) \sqsubseteq (N_2|P_2|Q_2|R_2) \Leftrightarrow N_1 \subseteq N_2 \wedge P_1 \subseteq P_2 \wedge Q_1 \subseteq Q_2 \wedge R_1 \subseteq R_2$$

– The computation with no executions is

$$\bot = (\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O})$$

– The computation that does not change the state is

$$1 = (\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{I})$$

– The computation with all executions is

$$\top = (\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F})$$

– The computation with all incomplete and infinite executions is

$$\mathsf{L} = (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})$$

– The approximation order is:

$$\begin{aligned}(N_1|P_1|Q_1|R_1) \sqsubseteq\!\!\sqsubseteq (N_2|P_2|Q_2|R_2) \Leftrightarrow N_2 \subseteq N_1 \wedge P_1 \subseteq P_2 \subseteq P_1 \cup N_1 \wedge \\ Q_2 \subseteq Q_1 \wedge R_1 \subseteq R_2 \subseteq R_1 \cup N_1\end{aligned}$$

With these operations, the set S of computations forms a lattice-ordered semiring $(S, \sqcup, \cdot, \sqcap, \bot, 1, \top)$, which has the lattice order $\sqsubseteq$ and is MID-complete. This can be extended to a lattice-ordered itering; details including the definition of $^\circ$ for computations are given in [21].

As usual, the implication order reflects the amount of non-determinism of computations and the notion of a computation refining another if the former contains a subset of the executions of the latter. The four subset relationships $N_1 \subseteq N_2$ and $P_1 \subseteq P_2$ and $Q_1 \subseteq Q_2$ and $R_1 \subseteq R_2$ state this for the four kinds of execution represented in our model.

The above approximation order has been derived in [21] using algebraic techniques. Intuitively it states that a more precise approximation can add finite and aborting executions ($P_1 \subseteq P_2 \wedge R_1 \subseteq R_2$) provided they extend incomplete executions ($P_2 \subseteq P_1 \cup N_1 \wedge R_2 \subseteq R_1 \cup N_1$), and can remove only incomplete and infinite executions ($N_2 \subseteq N_1 \wedge Q_2 \subseteq Q_1$).

Let S be the computations in our model and let $f : S \to S$. Provided it exists, the $\sqsubseteq$-least fixpoint of f is denoted by κf:

$$f(\kappa f) = \kappa f \qquad\qquad f(x) = x \Rightarrow \kappa f \sqsubseteq x$$

The following result is a consequence of [21, Theorems 2 and 4]. Item 4 uses the abbreviation $c(x) = n(\mathsf{L})\top \sqcap x$ and the operation n, which elaborates to $n(N|P|Q|R) = (\mathsf{O}|\mathsf{O}|Q|N)$ in our computation model.

Proposition 1. *Let S be the computations in our model.*

1. *The relation $\sqsubseteq$ is a partial order with least element L.*
2. *The operations $\sqcup$ and $\cdot$ and $^\circ$ are $\sqsubseteq$-isotone.*

Let $f : S \to S$ be $\sqsubseteq$- and $\sqsubseteq$-isotone. Then the following are equivalent:

3. *κf exists and $\kappa f = (\nu f \sqcap \mathsf{L}) \sqcup \mu f$.*
4. *$c(\nu f) \sqsubseteq (\nu f \sqcap \mathsf{L}) \sqcup \mu f \sqcup n(\nu f)\top$.*

Items 1 and 2 state basic properties of the approximation order and operations used for defining program constructs. The equivalence of items 3 and 4 gives a condition for the existence of κf in terms of μf and νf and reduces calculation of κf to that of μf and νf. This is already helpful as μf and νf are based on the implication order and therefore often easier to obtain than κf directly. In the following we aim to establish an even closer connection, namely $\kappa f = \nu f$, under suitable conditions.

3 Overall Strategy

In the remainder of this paper we further study how the $\sqsubseteq$-least fixpoint κf and the $\sqsubseteq$-greatest fixpoint νf of a function f are related when f represents computations with progress. We will give sufficient conditions for $\kappa f = \nu f$. Corollary 11 will establish this equality by using Proposition 1. To this end, we will derive conditions under which $\nu f \sqsubseteq \mu f \sqcup \mathsf{L}$.

To establish $\nu f \sqsubseteq \mu f \sqcup \mathsf{L}$, the main idea is to bound the excess of νf over μf by a meet of the form $\bigsqcap a_i$ for suitable elements a_i. That is, we prove $\nu f \sqsubseteq \mu f \sqcup \bigsqcap a_i$ and $\bigsqcap a_i \sqsubseteq \mathsf{L}$. Corollary 10 will show the latter. The former is established by the general Theorem 3 and a series of specialisations in Corollaries 4, 5 and 7. The specialisations instantiate $a_i = b^i\top$ for a suitable element b.

While our unified computation model supports a notion of progress, we still need a way to study different kinds of progress and to ensure a given computation actually makes progress. The idea for this is to separate the two concerns of progress and computation, which is based on having a separate time variable as described in [25–27]. To achieve this separation algebraically, we split the function f representing a recursion into a composition $f = g \circ h$. Here, the function h represents the computation done in the body of a recursion (without further recursive invocations) and the function g represents the progress made by this computation.

By imposing different conditions on the function g we can ensure actual progress and study different kinds of progress; this is shown in Theorem 9 and subsequently discussed in Section 5. Theorem 6 shows that the function h can represent computations carried out by programs constructed from choice, sequence, iteration and similar constructs. Specifically, we will consider the function $f(x) = a \cdot h(x)$, where progress is modelled by a suitable element a that satisfies $a \sqsubseteq b$ and is sequentially composed with the computation $h(x)$.

The development is split into two sections. We first cover results that can be derived algebraically in Section 4. In Section 5 we look closer into the computation model to discuss progress.

4 Connecting Fixpoints Algebraically

In this section we derive general results relating the $\sqsubseteq$-least and $\sqsubseteq$-greatest fixpoints of a function in the algebraic setting of Section 2.1.

We start with a simple upper bound for $\sqsubseteq$-greatest fixpoints. Kleene's recursion theorem gives the representation $\nu f = \bigsqcap_{i \in \mathbb{N}} f^i(\top)$ if the function f satisfies certain continuity requirements [9, 30, 32]. It is evident from previous proofs that one of the two inequalities comprising this equation only needs that f is $\sqsubseteq$-isotone.

Lemma 2. *Let S be a complete lattice. Let $f : S \to S$ be $\sqsubseteq$-isotone. Then $\nu f \sqsubseteq \bigsqcap_{i \in \mathbb{N}} f^i(\top)$.*

Proof. The claim follows if $\nu f \sqsubseteq f^i(\top)$ for each $i \in \mathbb{N}$. We show this by induction over i. The base case $i = 0$ follows since $\nu f \sqsubseteq \top$. The inductive case follows since for $i \in \mathbb{N}$ we have

$$\nu f = f(\nu f) \sqsubseteq f(f^i(\top)) = f^{i+1}(\top)$$

using that f is $\sqsubseteq$-isotone. □

Next comes a general result that relates νf and μf by capturing the excess of $f^i(\top)$ over μf in the element a_i. It also splits the function f into a composition $g \circ h$ where an application of h does not increase the excess and an application of g may increase the excess from a_i to a_{i+1}. When we apply this result later, the component g will represent the progress of the computation.

Theorem 3. *Let S be a MID-complete lattice. Let $g : S \to S$ distribute over $\sqcup$ and let $h : S \to S$ be $\sqsubseteq$-isotone. Let $a_i \in S$ such that $g(a_i) \sqsubseteq a_{i+1}$ for each $i \in \mathbb{N}$. Assume $h(\top) \sqsubseteq a_0$ and $h(a_i \sqcup x) \sqsubseteq a_i \sqcup h(x)$ for each $i \geq 1$ and $x \in S$. Consider $f = g \circ h$. Then $\nu f \sqsubseteq \mu f \sqcup \bigsqcap_{i \geq 1} a_i$.*

Proof. We show by induction over i that $f^i(\top) \sqsubseteq a_i \sqcup \mu f$ for each $i \geq 1$. The base case $i = 1$ follows by

$$f(\top) = g(h(\top)) \sqsubseteq g(a_0) \sqsubseteq a_1 \sqsubseteq a_1 \sqcup \mu f$$

using that g is $\sqsubseteq$-isotone since it distributes over $\sqcup$. The inductive case follows since for $i \geq 1$ we have

$$\begin{aligned} f^{i+1}(\top) &= f(f^i(\top)) = g(h(f^i(\top))) \sqsubseteq g(h(a_i \sqcup \mu f)) \sqsubseteq g(a_i \sqcup h(\mu f)) \\ &= g(a_i) \sqcup g(h(\mu f)) \sqsubseteq a_{i+1} \sqcup f(\mu f) = a_{i+1} \sqcup \mu f \end{aligned}$$

using the assumption $h(a_i \sqcup x) \sqsubseteq a_i \sqcup h(x)$ with $x = \mu f$. The overall claim then follows by

$$\nu f \sqsubseteq \textstyle\bigsqcap_{i \in \mathbb{N}} f^i(\top) \sqsubseteq \bigsqcap_{i \geq 1} f^i(\top) \sqsubseteq \bigsqcap_{i \geq 1} (a_i \sqcup \mu f) = \mu f \sqcup \bigsqcap_{i \geq 1} a_i$$

using Lemma 2. □

The following corollaries specialise the previous result. We first instantiate the excess to $a_i = b^i c$ for suitable elements b and c, and the function g to sequential composition of an element a such that $a \sqsubseteq b$. The latter condition guarantees that an application of g increases the excess from a_i to at most a_{i+1}.

The intuition is that b captures progress made by the body of the recursion (without further invocations). Hence $b^i c$ represents progress up to the ith unfolding of the recursion where c contains potential progress by further unfoldings.

Like b, the element a captures progress of the computation. Distinguishing a and b in the following results makes it possible later to establish $h(b^i c \sqcup x) \sqsubseteq b^i c \sqcup h(x)$, which separates the progress part $b^i c$ from the remainder of the computation $h(x)$. Setting $a = b$ would work in many models, but in trace models the element a represents specific progress that does not commute with other parts of the computation, so cannot be separated this way. However, we are able to choose an upper bound b on the progress that does commute and therefore can be collected separately. This will be demonstrated in more detail in Section 5.

Corollary 4. *Let S be a MID-complete lattice-ordered semiring. Let $h : S \to S$ be $\sqsubseteq$-isotone. Let $a, b, c \in S$ such that $a \sqsubseteq b$ and $h(\top) \sqsubseteq c$ and $h(b^i c \sqcup x) \sqsubseteq b^i c \sqcup h(x)$ for each $i \geq 1$ and $x \in S$. Consider $f(x) = a \cdot h(x)$. Then $\nu f \sqsubseteq \mu f \sqcup \bigsqcap_{i \geq 1} b^i c$.*

Proof. Let $g(x) = ax$ and $a_i = b^i c$ for each $i \in \mathbb{N}$. Then g distributes over $\sqcup$ and $h(\top) \sqsubseteq c = a_0$ and

$$g(a_i) = g(b^i c) = ab^i c \sqsubseteq bb^i c = b^{i+1} c = a_{i+1}$$

for each $i \in \mathbb{N}$. Moreover $h(a_i \sqcup x) \sqsubseteq a_i \sqcup h(x)$ for each $i \geq 1$ and $x \in S$. Finally $f = g \circ h$. Thus the claim follows by Theorem 3. □

We next instantiate $c = \top$, which simplifies the statement of the previous result and is sufficient for the development in this paper. The intuition for setting $c = \top$ is to make no assumptions in the ith unfolding of the recursion about progress that may happen in further recursive invocations.

Corollary 5. *Let S be a MID-complete lattice-ordered semiring. Let $h : S \to S$ be $\sqsubseteq$-isotone. Let $a, b \in S$ such that $a \sqsubseteq b$ and $h(b^i\top \sqcup x) \sqsubseteq b^i\top \sqcup h(x)$ for each $i \geq 1$ and $x \in S$. Consider $f(x) = a \cdot h(x)$. Then $\nu f \sqsubseteq \mu f \sqcup \bigsqcap_{i\geq 1} b^i\top$.*

Proof. Let $c = \top$. Then $h(\top) \sqsubseteq c$ and $h(b^i c \sqcup x) \sqsubseteq b^i c \sqcup h(x)$ for each $i \geq 1$ and $x \in S$. Thus the claim follows by Corollary 4. $\square$

The elements $b^i\top$ for $i \geq 1$ form a chain since $b^{i+1}\top = b^i b\top \sqsubseteq b^i\top$.

We still need to ensure that an application of h does not increase the excess, that is, $h(a_i \sqcup x) \sqsubseteq a_i \sqcup h(x)$ in Theorem 3. The next result gives sufficient conditions for an element a_i to satisfy this property.

The ideas underlying the conditions $a_i x \sqsubseteq a_i$ and $x a_i \sqsubseteq a_i \sqcup x\bot$ are as follows. The element $a_i = b^i\top$ represents the progress b^i made by the first i unfoldings of the recursion followed by arbitrary behaviour $\top$ in further unfoldings. The composition $a_i x$ makes another computation x at the end, but this does not add anything new to the arbitrary behaviour $\top$ already available at the end of the ith unfolding, which results in $a_i x \sqsubseteq a_i$.

In contrast, the composition $x a_i$ makes another computation x at the start, which can have an overall effect. Observe that among the executions contained in x, only the finite ones will reach the computation a_i. The infinite, aborting and incomplete executions of x absorb any subsequent computation and are contained in $x\bot$. The element b is chosen to capture progress in a general way so that the finite executions of x commute with it. Hence this part of x can be postponed all the way until after the ith unfolding, where again it adds nothing new to the arbitrary behaviour $\top$. All executions together are therefore contained as per $x a_i \sqsubseteq a_i \sqcup x\bot$.

Theorem 6. *Let S be a lattice-ordered itering. Let $h : S \to S$ such that $h(x)$ is constructed from the parameter x, arbitrary constants and the operations $\sqcup$, $\sqcap$, $\cdot$ and $^\circ$. Let $c \in S$ such that $cx \sqsubseteq c$ and $xc \sqsubseteq c \sqcup x\bot$ for each $x \in S$. Then $h(c \sqcup x) \sqsubseteq c \sqcup h(x)$ for each $x \in S$.*

Proof. We prove the claim by induction over the structure of the expression defining h. The base case $h(x) = x$ follows by

$$h(c \sqcup x) = c \sqcup x = c \sqcup h(x)$$

The base case $h(x) = d$ for a constant $d \in S$ follows by

$$h(c \sqcup x) = d \sqsubseteq c \sqcup d = c \sqcup h(x)$$

The inductive case $h(x) = f(x) \sqcup g(x)$ follows by

$$h(c \sqcup x) = f(c \sqcup x) \sqcup g(c \sqcup x) \sqsubseteq c \sqcup f(x) \sqcup c \sqcup g(x) = c \sqcup f(x) \sqcup g(x) = c \sqcup h(x)$$

The inductive case $h(x) = f(x) \sqcap g(x)$ follows by

$$h(c \sqcup x) = f(c \sqcup x) \sqcap g(c \sqcup x) \sqsubseteq (c \sqcup f(x)) \sqcap (c \sqcup g(x)) = c \sqcup (f(x) \sqcap g(x)) = c \sqcup h(x)$$

The inductive case $h(x) = f(x)g(x)$ follows by

$$\begin{aligned} h(c \sqcup x) &= f(c \sqcup x)g(c \sqcup x) \sqsubseteq (c \sqcup f(x))(c \sqcup g(x)) \\ &= c(c \sqcup g(x)) \sqcup f(x)c \sqcup f(x)g(x) \\ &\sqsubseteq c \sqcup c \sqcup f(x)\bot \sqcup f(x)g(x) = c \sqcup f(x)g(x) = c \sqcup h(x) \end{aligned}$$

using the assumption $cy \sqsubseteq c$ with $y = c \sqcup g(x)$ and the assumption $zc \sqsubseteq c \sqcup z\bot$ with $z = f(x)$.

The inductive case $h(x) = f(x)^\circ$ follows by

$$\begin{aligned} h(c \sqcup x) &= f(c \sqcup x)^\circ \sqsubseteq (c \sqcup f(x))^\circ = f(x)^\circ(c \cdot f(x)^\circ)^\circ \sqsubseteq f(x)^\circ c^\circ \\ &= f(x)^\circ(1 \sqcup cc^\circ) \sqsubseteq f(x)^\circ(1 \sqcup c) = f(x)^\circ \sqcup f(x)^\circ c \\ &\sqsubseteq f(x)^\circ \sqcup c \sqcup f(x)^\circ\bot = c \sqcup f(x)^\circ = c \sqcup h(x) \end{aligned}$$

using sumstar and unfold properties of $^\circ$, the assumption $cy \sqsubseteq c$ with $y = f(x)^\circ$ and with $y = c^\circ$ and the assumption $zc \sqsubseteq c \sqcup z\bot$ with $z = f(x)^\circ$. □

This means that the results of this section apply to any recursion whose characteristic function h is composed of constants, $\sqcup$, $\sqcap$, $\cdot$ and $^\circ$. In particular, the body of the recursion can use sequential compositions, conditionals and while-loops, the semantics of which are based on $\sqcup$, $\cdot$ and $^\circ$.

The following result combines Corollary 5 and Theorem 6. It provides the interface to the model-based discussion in Section 5.

Corollary 7. *Let S be a MID-complete lattice-ordered itering. Let $h : S \to S$ such that $h(x)$ is constructed from the parameter x, arbitrary constants and the operations $\sqcup$, $\sqcap$, $\cdot$ and $^\circ$. Let $a, b \in S$ such that $a \sqsubseteq b$ and $xb^i\top \sqsubseteq b^i\top \sqcup x\bot$ for each $i \geq 1$ and $x \in S$. Consider $f(x) = a \cdot h(x)$. Then $\nu f \sqsubseteq \mu f \sqcup \bigsqcap_{i \geq 1} b^i\top$.*

Proof. We apply Theorem 6 with $c = b^i\top$ for each $i \geq 1$. For this, first observe that $cx = b^i\top x \sqsubseteq b^i\top = c$ for each $x \in S$ since $\top x \sqsubseteq \top$ and composition is $\sqsubseteq$-isotone. Second, $xc \sqsubseteq c \sqcup x\bot$ for each $x \in S$ by the assumption of the present corollary. Hence $h(c \sqcup x) \sqsubseteq c \sqcup h(x)$ for each $x \in S$ by Theorem 6. Moreover, the function h is $\sqsubseteq$-isotone since it is composed of $\sqsubseteq$-isotone constructs. Thus the claim follows by Corollary 5. □

We conclude this section with the following version of Theorem 3, which applies to functions of the form $f' = h \circ g$ instead of $f = g \circ h$. Other than this swap, the assumptions and conclusions of the two theorems are the same. The proof is by rolling fixpoints.

Theorem 8. *Let S be a MID-complete lattice. Let $g : S \to S$ distribute over $\sqcup$ and let $h : S \to S$ be $\sqsubseteq$-isotone. Let $a_i \in S$ such that $g(a_i) \sqsubseteq a_{i+1}$ for each $i \in \mathbb{N}$. Assume $h(\top) \sqsubseteq a_0$ and $h(a_i \sqcup x) \sqsubseteq a_i \sqcup h(x)$ for each $i \geq 1$ and $x \in S$. Consider $f' = h \circ g$. Then $\nu f' \sqsubseteq \mu f' \sqcup \bigsqcap_{i \geq 1} a_i$.*

Proof. Let $f = g \circ h$. Then $\nu f' = \nu(h \circ g) = h(\nu(g \circ h)) = h(\nu f)$ by the rolling property of ν. Similarly $\mu f' = h(\mu f)$ by the rolling property of μ. Moreover for each $i \geq 1$ we obtain

$$h(\mu f \sqcup \textstyle\bigsqcap_{j \geq 1} a_j) \sqsubseteq h(\mu f \sqcup a_i) \sqsubseteq a_i \sqcup h(\mu f)$$

using the assumption $h(a_i \sqcup x) \sqsubseteq a_i \sqcup h(x)$ with $x = \mu f$. Hence

$$\begin{aligned}\nu f' &= h(\nu f) \sqsubseteq h(\mu f \sqcup \textstyle\bigsqcap_{j \geq 1} a_j) \sqsubseteq \bigsqcap_{i \geq 1}(a_i \sqcup h(\mu f)) = h(\mu f) \sqcup \bigsqcap_{i \geq 1} a_i \\ &= \mu f' \sqcup \textstyle\bigsqcap_{i \geq 1} a_i\end{aligned}$$

using Theorem 3. □

5 Connecting Fixpoints of Computations with Progress

In this section we work in the more detailed setting of the computation model presented in Section 2.2. This allows us to discuss progress more precisely.

In order to apply Corollary 7 we need to choose b such that $xb^i\top \sqsubseteq b^i\top \sqcup x\bot$ for each $x \in S$ and $i \geq 1$. As the following result shows, this condition holds for each computation $b = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B)$ such that $\mathsf{F}B \subseteq B\mathsf{F}$.

Because we have separated the progress part from the actual computation in the recursion, we can now focus solely on the progress part. The intuition is that b is a general computation which captures a finite amount of progress. In particular, b does not contain any incomplete, aborting or infinite executions, which gives the form $(\mathsf{O}|\mathsf{O}|\mathsf{O}|B)$ for a relation B. Typically B will affect only the part of the state representing progress (such as a clock or a trace). The relation F describes an arbitrary amount of finite progress and allows arbitrary changes to the remaining part of the state. In most instances discussed below, B and F actually commute, but this is not necessary for the following result.

Theorem 9. *Let S be the computations in our model. Let $b = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B)$ for a relation $B \subseteq \mathsf{F}$ with $\mathsf{F}B \subseteq B\mathsf{F}$. Then $xb^i\top \sqsubseteq b^i\top \sqcup x\bot$ for each $x \in S$ and $i \geq 1$.*

Proof. We first prove $b^i = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B^i)$ for each $i \geq 1$ by induction over i. The base case $i = 1$ follows immediately. The inductive case $i \geq 1$ holds by

$$b^{i+1} = bb^i = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B)(\mathsf{O}|\mathsf{O}|\mathsf{O}|B^i) = (\mathsf{O}|\mathsf{O}|\mathsf{O}|BB^i) = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B^{i+1})$$

Hence for each $i \geq 1$ we have

$$b^i\top = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B^i)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) = (B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F})$$

We next prove that $\mathsf{F}B \subseteq B\mathsf{F}$ implies $\mathsf{F}B^i \subseteq B^i\mathsf{F}$ for each $i \geq 0$ by induction over i. The base case $i = 0$ follows by

$$\mathsf{F}B^0 = \mathsf{F}\mathsf{I} = \mathsf{F} = \mathsf{I}\mathsf{F} = B^0\mathsf{F}$$

The inductive case $i \geq 0$ follows by

$$\mathsf{F}B^{i+1} = \mathsf{F}B^iB \subseteq B^i\mathsf{F}B \subseteq B^iB\mathsf{F} = B^{i+1}\mathsf{F}$$

Thus for arbitrary $x = (N|P|Q|R) \in S$ and $i \geq 1$ we obtain

$$\begin{aligned}
xb^i\top &= (N|P|Q|R)(B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F}) \\
&\sqsubseteq (N|P|Q|\mathsf{F})(B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F}) \\
&= (N \cup \mathsf{F}B^i\mathsf{F}|P \cup \mathsf{F}B^i\mathsf{F}|Q \cup \mathsf{F}B^i\mathsf{F}_\infty|\mathsf{F}B^i\mathsf{F}) \\
&\sqsubseteq (N \cup B^i\mathsf{F}\mathsf{F}|P \cup B^i\mathsf{F}\mathsf{F}|Q \cup B^i\mathsf{F}\mathsf{F}_\infty|B^i\mathsf{F}\mathsf{F}) \\
&= (N \cup B^i\mathsf{F}|P \cup B^i\mathsf{F}|Q \cup B^i\mathsf{F}_\infty|B^i\mathsf{F}) \\
&= (B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F}) \cup (N|P|Q|\mathsf{O}) \\
&= (B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F}) \cup (N \cup R\mathsf{O}|P \cup R\mathsf{O}|Q \cup R\mathsf{O}|R\mathsf{O}) \\
&= (B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F}) \cup (N|P|Q|R)(\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O}) \\
&= b^i\top \sqcup x\bot
\end{aligned}$$

using the progress requirement $R \subseteq \mathsf{F}$ and $\mathsf{F}B^i \subseteq B^i\mathsf{F}$. □

We next consider if the expression $\bigsqcap_{i\geq 1} b^i\top$, which appears in Corollary 7, can be bounded by L. The following result gives a sufficient criterion based on the intersection $B' = \bigcap_{i\geq 1} B^i\mathsf{F}$.

Corollary 10. *Let S be the computations in our model. Let $h : S \to S$ such that $h(x)$ is constructed from the parameter x, arbitrary constants and the operations $\sqcup$, $\sqcap$, $\cdot$ and $^\circ$. Let $a \sqsubseteq b = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B)$ for a relation $B \subseteq \mathsf{F}$ with $\mathsf{F}B \subseteq B\mathsf{F}$ and $\bigcap_{i\geq 1} B^i\mathsf{F} = \mathsf{O}$. Consider $f(x) = a \cdot h(x)$. Then $\nu f \sqsubseteq \mu f \sqcup \mathsf{L}$.*

Proof. By Corollary 7 and Theorem 9 we obtain $\nu f \sqsubseteq \mu f \sqcup \bigsqcap_{i\geq 1} b^i\top$. From this the claim follows by

$$\begin{aligned}
\textstyle\bigsqcap_{i\geq 1} b^i\top &= \textstyle\bigsqcap_{i\geq 1}(B^i\mathsf{F}|B^i\mathsf{F}|B^i\mathsf{F}_\infty|B^i\mathsf{F}) \sqsubseteq \bigsqcap_{i\geq 1}(B^i\mathsf{F}|B^i\mathsf{F}|\mathsf{F}_\infty|B^i\mathsf{F}) \\
&= \textstyle(\bigcap_{i\geq 1} B^i\mathsf{F}|\bigcap_{i\geq 1} B^i\mathsf{F}|\bigcap_{i\geq 1} \mathsf{F}_\infty|\bigcap_{i\geq 1} B^i\mathsf{F}) = (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) \sqsubseteq \mathsf{L}
\end{aligned}$$

using that $B^i\mathsf{F}_\infty \subseteq \mathsf{F}_\infty$ by the progress requirement on that component. □

The value of B' depends on the kind of progress represented by the computation model. We discuss several examples. We first consider examples which satisfy $a = b$ and then examples where $a = (\mathsf{O}|\mathsf{O}|\mathsf{O}|A)$ for $A \subset B$.

In the Boolean-time model, $\mathsf{F} = \mathsf{T}$ holds, so the assumption $\mathsf{F}B \subseteq B\mathsf{F}$ is equivalent to $\mathsf{T}B \subseteq B\mathsf{T}$, which implies $\mathsf{T}B\mathsf{T} \subseteq B\mathsf{T}\mathsf{T} = B\mathsf{T}$. By the Tarski property of relations [35], either $B = \mathsf{O}$ or $\mathsf{T}B\mathsf{T} = \mathsf{T}$, in which case $B\mathsf{T} = \mathsf{T}$ follows. If $B = \mathsf{O}$, we have $a = b = \bot$ and therefore $f(x) = \bot$, whence Corollary 10 is vacuous. If $B\mathsf{T} = \mathsf{T}$, we have $B^i\mathsf{T} = \mathsf{T}$ for each $i \geq 1$ by induction, so $B' = \bigcap_{i\geq 1} B^i\mathsf{T} = \bigcap_{i\geq 1} \mathsf{T} = \mathsf{T} \neq \mathsf{O}$, whence Corollary 10 does not apply. The notion of progress in the Boolean-time model is too coarse for this result.

The abstract-time model counts steps and here $\mathsf{F} = \{((v,t),(v',t')) \mid t \leq t'\}$ specifies that the counter, which is a natural number, does not decrease. To achieve strict progress, we consider

$$B = \{((v,t),(v,t')) \mid t' - t > 0\} = \{((v,t),(v,t')) \mid t' - t \geq 1\}$$

specifying that the state remains unchanged but the counter must increase. It follows that

$$\mathsf{F}B = B\mathsf{F} = \{((v,t),(v',t')) \mid t' - t \geq 1\}$$

specifying that the counter must increase. Moreover,

$$B^i = \{((v,t),(v,t')) \mid t' - t \geq i\}$$

for each $i \geq 1$ by induction, specifying that the state remains unchanged and the counter increases by at least i. Hence

$$B^i\mathsf{F} = \{((v,t),(v',t')) \mid t' - t \geq i\}$$

for each $i \geq 1$, specifying that the counter increases by at least i. Therefore $B' = \bigcap_{i \geq 1} B^i\mathsf{F} = \mathsf{O}$ since any finite difference $t' - t$ will be exceeded after sufficiently many steps. Thus Corollary 10 applies.

In the real-time model we again have $\mathsf{F} = \{((v,t),(v',t')) \mid t \leq t'\}$, but the previous argument fails. Specifically, strict progress using

$$B = \{((v,t),(v,t')) \mid t' - t > 0\}$$

implies that

$$\mathsf{F}B = B\mathsf{F} = \{((v,t),(v',t')) \mid t' - t > 0\}$$

and $B^2 = B$, whence $B^i = B$ for each $i \geq 1$ by induction. Hence $B^i\mathsf{F} = B\mathsf{F}$ for each $i \geq 1$. It follows that $B' = B\mathsf{F} \neq \mathsf{O}$, so Corollary 10 does not apply. More precisely, we obtain only $\bigsqcap_{i \geq 1} b^i\top \sqsubseteq (B\mathsf{F}|B\mathsf{F}|\mathsf{F}_\infty|B\mathsf{F})$, which is not below L.

However, if strict progress means at least c units of time pass for an arbitrary $c > 0$, that is,

$$B = \{((v,t),(v,t')) \mid t' - t \geq c\}$$

we obtain

$$B^i = \{((v,t),(v,t')) \mid t' - t \geq ic\}$$

for each $i \geq 1$ by induction. Moreover,

$$\mathsf{F}B = B\mathsf{F} = \{((v,t),(v',t')) \mid t' - t \geq c\}$$

and

$$B^i\mathsf{F} = \{((v,t),(v',t')) \mid t' - t \geq ic\}$$

reflecting that at least ic units of time pass after i steps. In this case, we obtain $B' = \bigcap_{i \geq 1} B^i\mathsf{F} = \mathsf{O}$ again, whence Corollary 10 applies.

So, in the real-time model the argument fails because there is no lower bound on the progress in each step. A particular instance of this is the Zeno effect where each step takes half the time of the preceding step. The argument works if we assume a uniform lower bound for all steps.

We remark that it is also possible to specify an upper bound on the amount of progress. One way to see this is to consider for an arbitrary $d \geq c$ the relation

$$A = \{((v,t),(v,t')) \mid d \geq t' - t \geq c\}$$

whence $A \subset B$, and use the previous argument for B. This includes the case $c = d$ where the amount of progress in each step is exactly specified.

In the trace model, $\mathsf{F} = \preceq$ is the prefix relation on traces. We can achieve strict progress, for example, by

$$A = \{(tr, tr') \mid tr' = tr +\!\!+ [last(tr)]\}$$

which appends to the sequence tr its last element. Hence $A \subset B$ for the strict prefix relation $B = \prec$, which specifies that the trace gets longer by at least one element. It follows that $\mathsf{F}B = B = B\mathsf{F}$. Moreover B^i specifies that the trace gets longer by at least i elements. It follows that $B^i\mathsf{F} = B^i$. Thus $B' = \bigcap_{i\geq 1} B^i = \mathsf{O}$ since any finite length will be exceeded after sufficiently many steps. Therefore Corollary 10 applies.

The previous example also demonstrates why Corollaries 4, 5, 7 and 10 distinguish between a and b. Namely, to apply these results if $a = b$ we would need the condition $\mathsf{F}A \subseteq A\mathsf{F}$, which does not hold in this case. This is because F may extend the trace in an arbitrary way while A appends a specific element. However, A refines the more general B, which commutes with F.

We finally use Corollary 10 to connect least fixpoints in the approximation order with greatest fixpoints in the implication order.

Corollary 11. *Let S be the computations in our model. Let $h : S \to S$ such that $h(x)$ is constructed from the parameter x, arbitrary constants and the operations $\sqcup$, $\cdot$ and $^\circ$. Let $a \sqsubseteq b = (\mathsf{O}|\mathsf{O}|\mathsf{O}|B)$ for a relation $B \subseteq \mathsf{F}$ with $\mathsf{F}B \subseteq B\mathsf{F}$ and $\bigcap_{i\geq 1} B^i\mathsf{F} = \mathsf{O}$. Consider $f(x) = a \cdot h(x)$. Then κf exists and $\kappa f = \nu f$.*

Proof. By item 2 of Proposition 1, the functions f and h are $\sqsubseteq$-isotone. We show the condition in item 4 of Proposition 1. By Corollary 10, we have $\nu f \sqsubseteq \mu f \sqcup \mathsf{L}$. Hence

$$\begin{aligned} c(\nu f) = n(\mathsf{L})\top \sqcap \nu f \sqsubseteq \nu f &= \nu f \sqcap (\mu f \sqcup \mathsf{L}) = (\nu f \sqcap \mu f) \sqcup (\nu f \sqcap \mathsf{L}) \\ &= (\nu f \sqcap \mathsf{L}) \sqcup \mu f \sqsubseteq (\nu f \sqcap \mathsf{L}) \sqcup \mu f \sqcup n(\nu f)\top \end{aligned}$$

By item 3 of Proposition 1, it follows that κf exists and $\kappa f = (\nu f \sqcap \mathsf{L}) \sqcup \mu f$. Hence $\kappa f = \nu f$ using the previous calculation. $\square$

The assumptions of Corollary 11 are the same as those of Corollary 10 except the operation $\sqcap$ cannot be used in the construction of h, since it is not $\sqsubseteq$-isotone. The discussion following Corollary 10 applies in the same way.

6 Conclusion

In this paper we have studied when least fixpoints in the approximation order coincide with greatest fixpoints in the implication order. To achieve this, we separately captured the progress part of a recursive computation, an idea rooted in the time variables of [25–27]. The separation was achieved using algebraic means in a setting based on lattices and semirings, which unifies many computation models. We were then able to focus on the elements representing progress in a suitable unified computation model. This allowed us to state conditions under which the studied fixpoints coincide and to discuss the meaning of these conditions for various kinds of progress.

Two anonymous referees asked about necessary conditions for $\kappa f = \nu f$. Since Corollary 11 makes several assumptions, this question can be considered in many different ways. We briefly discuss a negative answer leaving further investigation to future work.

Section 5 shows that the assumption $\bigcap_{i \geq 1} B^i \mathsf{F} = \mathsf{O}$ of Corollary 11 fails if there is no lower bound for the progress B made by the body of the recursion in the real-time model. But even then, if the body of the recursion $f(x)$ makes arbitrary finite progress followed by an error (without a recursive invocation of x), the characteristic function f is constant and therefore has a unique fixpoint. Hence κf exists and $\kappa f = \nu f$ in this case.

We finally mention connections to a number of related works, which can also be explored further.

The semantics of iteration in a timed computation model (an extension of UTP designs by a time variable) is defined in [24] by inserting a statement that increments the clock before the tail-recursive call. This can be seen as an instance of separating the progress part of a computation. Inserting before the recursive call corresponds to the pattern of Theorem 8.

Corollary 11 relies on the property $\kappa f = (\nu f \sqcap \mathsf{L}) \sqcup \mu f$ of the $\sqsubseteq$-least fixpoint of f. This property holds not only in the computation model discussed in this paper, but in a number of other relational, matrix-based and multirelational computation models [19, 20].

The implication and approximation orders are related to the truth and knowledge/information orders used in the bilattice approach to the semantics of logic programs [13, 15].

Acknowledgement. I thank the anonymous referees for their helpful comments.

References

1. Aarts, C.J., Backhouse, R.C., Boiten, E.A., Doornbos, H., van Gasteren, N., van Geldrop, R., Hoogendijk, P.F., Voermans, E., van der Woude, J.: Fixed-point calculus. Inf. Process. Lett. 53(3), 131–136 (1995)

2. Alexandru, A., Ciobanu, G.: Abstract interpretations in the framework of invariant sets. Fundamenta Informaticae 144(1), 1–22 (2016)
3. Back, R.J.R., von Wright, J.: Reasoning algebraically about loops. Acta Inf. 36(4), 295–334 (1999)
4. de Bakker, J.W.: Semantics and termination of nondeterministic recursive programs. In: Michaelson, S., Milner, R. (eds.) ICALP 1976. pp. 435–477. Edinburgh University Press (1976)
5. Broy, M., Gnatz, R., Wirsing, M.: Semantics of nondeterministic and noncontinuous constructs. In: Bauer, F.L., Broy, M. (eds.) Program Construction. LNCS, vol. 69, pp. 553–592. Springer (1979)
6. Cohen, E.: Separation and reduction. In: Backhouse, R., Oliveira, J.N. (eds.) MPC 2000. LNCS, vol. 1837, pp. 45–59. Springer (2000)
7. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall (1971)
8. Cousot, P., Cousot, R.: An abstract interpretation framework for termination. In: POPL 2012. pp. 245–257. ACM (2012)
9. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, second edn. (2002)
10. Dunne, S.: Recasting Hoare and He's Unifying Theory of Programs in the context of general correctness. In: Butterfield, A., Strong, G., Pahl, C. (eds.) 5th Irish Workshop on Formal Methods. Electronic Workshops in Computing, The British Computer Society (2001)
11. Dunne, S.: Conscriptions: A new relational model for sequential computations. In: Wolff, B., Gaudel, M.C., Feliachi, A. (eds.) UTP 2012. LNCS, vol. 7681, pp. 144–163. Springer (2013)
12. Egli, H.: A mathematical model for non-deterministic computations. Tech. rep., Forschungsinstitut für Mathematik ETH Zürich (1975)
13. Fitting, M.: Bilattices and the semantics of logic programming. Journal of Logic Programming 11(2), 91–116 (1991)
14. Freyd, P.J., Ščedrov, A.: Categories, Allegories, North-Holland Mathematical Library, vol. 39. Elsevier Science Publishers (1990)
15. Ginsberg, M.L.: Multivalued logics: a uniform approach to reasoning in artificial intelligence. Computational Intelligence 4(3), 265–316 (1988)
16. Gordon, M.J.C.: The denotational description of programming languages. Springer, New York (1979)
17. Guttmann, W.: Algebras for iteration and infinite computations. Acta Inf. 49(5), 343–359 (2012)
18. Guttmann, W.: Extended conscriptions algebraically. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M.E. (eds.) RAMiCS 2014. LNCS, vol. 8428, pp. 139–156. Springer (2014)
19. Guttmann, W.: Multirelations with infinite computations. Journal of Logical and Algebraic Methods in Programming 83(2), 194–211 (2014)
20. Guttmann, W.: Infinite executions of lazy and strict computations. Journal of Logical and Algebraic Methods in Programming 84(3), 326–340 (2015)
21. Guttmann, W.: An algebraic approach to computations with progress. Journal of Logical and Algebraic Methods in Programming 85(4), 520–539 (2016)
22. Guttmann, W., Möller, B.: Normal design algebra. Journal of Logic and Algebraic Programming 79(2), 144–173 (2010)
23. Hayes, I.J., Dunne, S.E., Meinicke, L.: Unifying theories of programming that distinguish nontermination and abort. In: Bolduc, C., Desharnais, J., Ktari, B. (eds.) MPC 2010. LNCS, vol. 6120, pp. 178–194. Springer (2010)

24. Hayes, I.J., Dunne, S.E., Meinicke, L.A.: Linking Unifying Theories of Program refinement. Sci. Comput. Program. 78(11), 2086–2107 (2013)
25. Hehner, E.C.R.: Real-time programming. Inf. Process. Lett. 30(1), 51–56 (1989)
26. Hehner, E.C.R.: Termination is timing. In: van de Snepscheut, J.L.A. (ed.) MPC 1989. LNCS, vol. 375, pp. 36–47. Springer (1989)
27. Hehner, E.C.R.: Abstractions of time. In: Roscoe, A.W. (ed.) A Classical Mind, chap. 12, pp. 191–210. Prentice Hall (1994)
28. Hoare, C.A.R., He, J.: Unifying theories of programming. Prentice Hall Europe (1998)
29. Höfner, P., Möller, B.: An algebra of hybrid systems. Journal of Logic and Algebraic Programming 78(2), 74–97 (2009)
30. Kleene, S.C.: Introduction to Metamathematics. North-Holland Publishing Company (1952)
31. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Information and Computation 110(2), 366–390 (1994)
32. Manna, Z.: Mathematical Theory of Computation. McGraw-Hill (1974)
33. Möller, B.: Kleene getting lazy. Sci. Comput. Program. 65(2), 195–214 (2007)
34. Schmidt, G., Hattensperger, C., Winter, M.: Heterogeneous relation algebra. In: Brink, C., Kahl, W., Schmidt, G. (eds.) Relational Methods in Computer Science, chap. 3, pp. 39–53. Springer, Wien (1997)
35. Schmidt, G., Ströhlein, T.: Relationen und Graphen. Springer (1989)
36. Shinwell, M.R.: The Fresh Approach: functional programming with names and binders. Ph.D. thesis, University of Cambridge (2005), available as Technical Report UCAM-CL-TR-618, Computer Laboratory, University of Cambridge
37. Stoy, J.E.: Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory. MIT Press (1977)
38. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5(2), 285–309 (1955)
39. Vuillemin, J.: Correct and optimal implementations of recursion in a simple programming language. Tech. Rep. 24, IRIA (1973), also in *Journal of Computer and System Sciences* 9(3):332–354, 1974
40. von Wright, J.: Towards a refinement algebra. Sci. Comput. Program. 51(1–2), 23–45 (2004)