# $f$-Generated Kleene Algebra

Peter Höfner

Institut für Informatik, Universität Augsburg
D-86135 Augsburg, Germany
hoefner@informatik.uni-augsburg.de

**Abstract.** When describing iterations or loops it is well known and common to use the Kleene star. We first show an example for iteration, where the star operation is not adequate, since it just iterate and do not modify the iterated element. Therefore we introduce, as a generalisation of Kleene algebra, the structure of $f$-generated Kleene algebra, that have an iteration operation which depends on a function $f$ and modify the iterated element in each step.

## 1 Introduction and Motivation

The use of Kleene star for describing iterations or loops is well known and common (see e.g. [3, 5]). From a theoretical point of view, $a^*$ is the least fixed point of $\lambda x.1 + a \cdot x$ and therefore characterises finite iteration. Nevertheless, as we will see, in some situations it is useful to have an additional function, which modifies an iterated element $a$ in each step; i.e., instead of calculating $a \cdot a \cdot \ldots \cdot a$, we want to get $a \cdot f(a) \cdot \ldots \cdot f^{n-1}(a)$. More precisely, $\lambda x.1 + a \cdot x$ is replaced by $\lambda x.1 + a \cdot f(x)$. As far as we know this generalisation of Kleene star has not been discussed before. Let us motivate our idea by a concrete example.

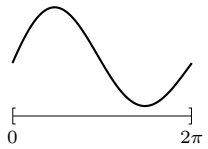
**Example 1.1** In [4] we presented an algebra of hybrid systems, which is based on (lazy) Kleene algebra and uses sets of trajectories as elements.[1] A *trajectory* $t$ is a pair $(d, g)$, where $d \in \mathbb{R}_0^+$ and $g : [0, d] \to V$, where $d$ is the *duration* of the trajectory and $V$ a set of (possible) *values*. We define composition of trajectories $(d_1, g_1)$ and $(d_2, g_2)$ as

$$(d_1, g_1) \cdot (d_2, g_2) =_{df} \begin{cases} (d_1 + d_2, g) \text{ if } g_1(d_1) = g_2(0) \\ \text{undefined} \quad \text{otherwise} \end{cases}$$

with $g(x) = g_1(x)$ for all $x \in [0, d_1]$ and $g(x + d_1) = g_2(x)$ for all $x \in [0, d_2]$. Since the algebra uses sets of trajectories as elements, the composition is lifted pointwise to those sets.

[1] For lack of space, we only recapitulate the definition and composition of trajectories and not the whole algebra. Furthermore, we restrict the set of durations to $\mathbb{R}_0^+$, which simplifies the structure and excludes trajectories with infinite length.

We assume the set, which only includes the trajectory $(2\pi, \frac{\sin x}{5x})$. This single set is called $t_{\sin}$. It describes a single swing of a pendulum. The element $t^*_{\sin}$ describes sequences of swings, but does not consider the fact that the pendulum gets slower by gravity, friction and so on. □



Sketch of $t_{\sin}$

## 2 $f$-Generated Kleene Algebra

Motivated by the previous example, we now define an iteration operator w.r.t. to a function $f$. In the remainder we will use $a, b, c \ldots$ for arbitrary elements of an idempotent semiring $S$.

**Definition 2.1** Let $f : S \to S$ be a homomorphism w.r.t. addition and multiplication, i.e., $f(a+b) = f(a) + f(b)$, $f(0) = 0$ and $f(a \cdot b) = f(a) \cdot f(b)$, $f(1) = 1$.[2] An *$f$-generated Kleene algebra* is a structure $(S, +, \cdot, 0, 1, {}^{f*})$, such that $(S, +, \cdot, 0, 1)$ is an idempotent semiring and ${}^{f*}$ satisfies

$$1 + a \cdot f(a^{f*}) \leq a^{f*} \qquad (f1) \qquad\qquad 1 + a \cdot f(b) \leq b \Rightarrow a^{f*} \leq b \qquad (f2)$$

Similarly to the Kleene star, ${}^{f*}$ is the least prefixed point of the function $\lambda x.1 + a \cdot f(x)$ and $(f1)$ can be strengthened to an equation. But in contrast to the Kleene star, we do not postulate the symmetrical laws of $(f1)$ and $(f2)$, since this would imply $a \cdot a^{f*} = a^{f*} \cdot a$. Using fixpoint iteration, we calculate for $\lambda x.1 + a \cdot f(x)$:

$$\begin{aligned}
x_0 &= 0 \\
x_1 &= 1 + a \cdot f(x_0) = 1 \\
x_2 &= 1 + a \cdot f(x_1) = 1 + a \\
x_3 &= 1 + a \cdot f(x_2) = 1 + a + a \cdot f(a) \\
x_4 &= 1 + a \cdot f(x_3) = 1 + a + a \cdot f(a) + a \cdot f(a) \cdot f(f(a))
\end{aligned}$$

In general, we get

$$x_n = 1 + a \cdot f(x_{n-1}) = \sum_{i=0}^{n-1} \prod_{j=0}^{i-1} f^j(a) \ , \qquad (1)$$

where, as usual, $\prod_{i=0}^{-1} x = 1$. Before returning to our example of Section 1, we give some other simple examples, which illustrate that there are more applications of the theory than trajectories.

**Example 2.2**

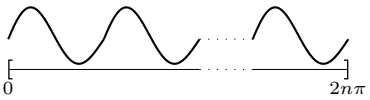
1. The standard Kleene star is subsumed by ${}^{f*}$ when choosing $f$ as identity.
2. Using infinite lists, the $f$-generated Kleene star corresponds to

   $$\texttt{sum scan } [\texttt{1}, \texttt{a}, \texttt{f(a)}, \texttt{f}^2\texttt{(a)}, \ldots]$$

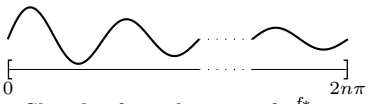
   in functional programming, which is discussed by Bird in [1]. □

[2] In the setting of semirings (monoids) $f(0) = 0$ does not follow from $f(a+b) = f(a) + f(b)$; this implication only holds in groups, rings, ....

**Example 1.1 continued** By setting $f((d,g)) =_{df} (d, \frac{g}{2})$, we are able to illustrate the difference between $t^*_{\sin}$ and $t^{f*}_{\sin}$. A characteristic element of $t^*_{\sin}$ just repeats $t_{\sin}$ for a finite number of times; whereas a characteristic element of $t^{f*}_{\sin}$ repeats and modifies $t_{\sin}$.



Sketch of an element of $t^*_{\sin}$



Sketch of an element of $t^{f*}_{\sin}$

Therefore we are able to describe a pendulum considering changes in time, like gravity, without changing the trajectory itself. Of course the function of our example as well as the trajectory are freely chosen and do not reflect reality. If one wants to describe a real pendulum, the trajectory and the function will getting much more complex; but do not change the idea of $^{f*}$. □

Of course, it is also possible to change the duration using another homomorphism $f$; e.g., to simulate Zeno effects one can set the function $f((d, g(x))) = (\frac{d}{2}, g(\frac{x}{2}))$.

## 3 Properties

In this section we discuss some properties of $^{f*}$. Since the Kleene star is a very special $f$-generated star, we cannot expect to get all the (well-known) properties of star in our setting. Therefore we also discuss those properties, which hold for Kleene star but not for the $f$-generated one.

First we give some useful properties of homomorphisms.

**Lemma 3.1** *Let $(M, +, 0)$ be a monoid and $f : M \to M$ be a homomorphism w.r.t. addition.*

1. *$f$ preserves idempotency.*
2. *$f$ is isotone, i.e, $a \leq b \Rightarrow f(a) \leq f(b)$.*

Following Kozen's approach to Kleene algebra with tests [5], we say that a *$f$-generated Kleene algebra with tests* is a $f$-generated Kleene algebra $S$ with a distinguished Boolean subalgebra $\mathsf{test}(S)$ of $[0, 1]$ with greatest element 1 and least element 0. Using tests yields an interesting result concerning $f$ which follows directly from the homomorphism properties.

**Lemma 3.2** *If $\mathsf{test}(S)$ is maximal, i.e., there is no Boolean subalgebra $B$ of $[0, 1]$ with $\mathsf{test}(S) \subset B$, then $f(p) \in \mathsf{test}(S)$ for all $p \in \mathsf{test}(S)$.*

In the remainder of the section we discuss some properties of $^{f*}$. First, we get immediately from $(f1)$ by lattice algebra

$$1 \leq a^{f*} \quad \text{and} \quad a \cdot f(a^{f*}) \leq a^{f*} \ . \tag{2}$$

Furthermore, we get laws similar to the standard Kleene star.

**Lemma 3.3**

1. $\prod_{j=0}^{n} f^j(a) \leq a^{f*}$ *for all* $n \in \mathbb{N}$,
2. $a^{f*} \leq a^{f*} \cdot a^{f*}$,
3. $a \leq 1 \Rightarrow a^{f*} = 1$,
4. $a \leq b \Rightarrow a^{f*} \leq b^{f*}$,
5. $a^{f*} \leq (a^{f*})^{f*}$.

*Proof.*

1. By (1) we get $\prod_{j=0}^{n-1} f^j(a) \leq x_n$ and $x_{n-1} \leq x_n$. Then the claim follows by induction.
2. By (2) and isotony, we get $1 \leq a^{f*} \Rightarrow a^{f*} \leq a^{f*} \cdot a^{f*}$.
3. By lattice algebra, homomorphism and $(f2)$, we get
   $a \leq 1 \Leftrightarrow 1 + a \leq 1 \Leftrightarrow 1 + a \cdot f(1) \leq 1 \Rightarrow a^{f*} \leq 1$.
   The other direction is by (2).
4. By $(f2)$, assumption, isotony and $(f1)$
   $a^{f*} \leq b^{f*} \Leftarrow 1 + a \cdot f(b^{f*}) \leq b^{f*} \Leftarrow 1 + b \cdot f(b^{f*}) \leq b^{f*} \Leftrightarrow \mathsf{true}$.
5. By 4., homomorphism and (2) (twice), we get
   $a^{f*} \leq (a^{f*})^{f*} \Leftarrow a \leq a^{f*} \Leftrightarrow a \cdot f(1) \leq a^{f*} \Leftarrow a \cdot f(a^{f*}) \leq a^{f*} \Leftarrow \mathsf{true}$.

□

Due to Example 2.2.1 the definition of $^{f*}$ is a generalisation of the standard Kleene star. Therefore we cannot expect that all properties of $^*$, like leapfrog, hold for it. More precisely, we have

**Lemma 3.4**

1. $a^{f*} \cdot a^{f*} \not\leq a^{f*}$.
2. $(a^{f*})^{f*} \not\leq a^{f*}$.
3. $a^{f*} \cdot (b \cdot a^{f*})^{f*} \not\leq (a + b)^{f*}$ *and* $(a + b)^{f*} \not\leq a^{f*} \cdot (b \cdot a^{f*})^{f*}$.
4. $a \cdot (b \cdot a)^{f*} \not\leq (a \cdot b)^{f*} \cdot a$ *and* $(a \cdot b)^{f*} \cdot a \not\leq a \cdot (b \cdot a)^{f*}$.

The proof is straightforward by choosing explicit elements. E.g., $a \cdot a \leq a^{f*} \cdot a^{f*}$ and $a \cdot a \not\leq a^{f*}$ implies the first item.

Since the function $f$ often simulates physical behaviours, it will explicitly occur in algebraic expressions. Therefore we briefly discuss the interaction between $f$ and $^{f*}$.

**Lemma 3.5** *Assume a $f$-generated Kleene algebra. Then $f(a)^{f*} \leq f(a^{f*})$. If $f$ is even universally disjunctive, then*

$$f(a)^{f*} = f(a^{f*}).$$

*Proof.* The first claim ($f(a)^{f*} \leq f(a^{f*})$) is immediate by induction $(f2)$ and the assumed properties of $f$. Using $\mu$-fusion together with $a^{f*} = \mu_x(1 + a \cdot f(x))$ and $f(a)^{f*} = \mu_x(1 + f(a) \cdot f(x))$, we get

$$f(1 + a \cdot f(x)) \leq 1 + f(a) \cdot f(f(x)) \Rightarrow f(a^{f*}) \leq a^{f*}.$$

The antecedent holds since $f$ is a homomorphism. □

## 4 Conclusion and Outlook

Sumarising, we have showed that the Kleene star is not the best in some situations. Therefore we introduced an $f$-generated Kleene star, which modifies the iterated element in each step. The new operator can be used e.g. in describing physical behaviours like a pendulum. We have also presented some basic properties of ${}^{f*}$.

Since this research is still ongoing, there is a lot of further work. First it will be interesting to find more properties of the homomorphism $f$ and the $f$-generated Kleene star. Especially the interaction of both should be discussed in more detail. Also the connection to functional programming (Example 2.2.2) will help to find more useful properties. This will lead to a better understanding of $f$-generated Kleene algebras.

In the paper we showed how to weaken the finite iteration operator of a Kleene algebra. But in the same way it should easily be possible to weaken the finite iteration of lazy Kleene algebra [6], the infinite iteration operator of omega algebras [2, 6] and the strong iteration of refinement algebra [7].

Additionally, as already mentioned, we give an algebra for hybrid systems, which is based on lazy Kleene algebra [4]. Using this approach the homomorphism $f$ can be interpreted as changing behaviours in (continuous) time. Therefore the operator ${}^{f*}$ fits well in the development, specification and analysis of hybrid systems. As a first step, a case study will be done.

**Acknowledgements.** I am very grateful to Bernhard Möller and Kim Solin for valuable and fruitful discussions.

## References

1. R. Bird. Lectures on constructive functional programming. In B. Manfred, editor, *Constructive Methods in Computing*, Science, volume F55 of NATO ASI Series, pages 151–216, 1989.
2. E. Cohen. Seperation and Reduction. In R. Backhouse and J. Olivera, editors, *Mathematics of Program Construction*, Lecture Notes in Computer Science 1837, pages 45–59, 2000.
3. J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
4. P. Höfner and B. Möller. Towards an Algebra of Hybrid Systems. In W. MacCaull, M. Winter, and I. Duentsch, editors, *Relational Methods in Computer Science*, Lecture Notes in Computer Science 3929, pages 121–133, 2006. (in press).
5. D. Kozen. Kleene Algebra with Tests. *Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.
6. B. Möller. Kleene getting lazy. *Science of Computer Programming, Special issue on MPC 2004*, 2006. (to appear)
   Previous version: B. Möller: Lazy Kleene algebra. In D. Kozen (ed.): Mathematics of program construction. LNCS 3125. Springer 2004, 252–273.
7. J. von Wright. From Kleene Algebra to Refinement Algebra. In E. Boiten and B. Möller, editors, *Proc. of 6th Int. Conf. on Mathematics of Program Construction, MPC 2002*, Lecture Notes in Computer Science 2386, pages 233–262, 2002.