2012-2013-06 June 2013

Universiteit Leiden
Opleiding Informatica

Typing Streams

Stefan Buijsman

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Typing Streams

Stefan Buijsman

1 Introduction

One of the more common datastructures in use today is a list of elements. These lists are characterized by being finite and are usually defined element by element. Lists are fairly well understood nowadays and almost every programmer can use them effectively. Recently however, computer scientists have looked for ways to create datastructures that are very much like lists except that they're infinite. This means that these datastructures, which are called streams, cannot be defined element by element but instead require a different kind of definition. Streams are usually defined by coinduction, which means that the first element is defined along with a way to calculate the next element from that first element. Coinduction is further clarified in [Rutten, 2001, p.5-7] and it is used in giving the semantics for streams in section 2.

In order to be able to say more about streams than the above, it is first necessary to introduce a simple language for the manipulation of streams. The syntax of this language can be found at the beginning of section 2, after which the semantics is explained. The syntax chosen here has the same basic operators as the syntax in Rutten [2001], but has fewer operators that build on these basic operators. On the other hand, this syntax does have an if-statement and a variable stream, which are not present in Rutten [2001]. The system presented here is thus in no way complete nor is it necessary to adopt all of its elements, rather it has been chosen in such a way as to show as many aspects of typing streams as possible.

Next to a simple language for the manipulation of streams, it is also necessary to have a language which can be used to type these streams. This language is presented in section 3 and is constructed in such a way as to fit the language for streams in section 2. Because there is an if-statement in section 2, it is necessary to also introduce a union type in section 3. Intuitively, this is because when I have an if-statement either the then-clause or the else-clause will be executed. This can be typed by saying that the if-statement has either the type of the stream in the then-clause or the type of the stream in the else-clause. This solution in particular, the use of union types, has not yet been observed in the literature and thus forms an essential part of the language presented in section 3.

With the two languages in hand, all that is missing is some way to use the typing language to say something about streams. In order to be able to do this, two steps are taken in this paper. First, an ordering between types is introduced. This allows one to speak about the inclusion of one type in another and can be used later on to be more flexible in deriving the kind of types a stream falls under. This apparatus is defined in section 4. Second, a derivation system is given which can be used to derive which types a stream falls under. This system also allows the use of variables, whose range can be restricted by adding assumptions about which types a variable must satisfy. This system is defined in section 5.

In order to illustrate how the formal system works, a few example derivations are provided in section 6. Then, in section 7, there is a discussion on a possible extension of the type language from section 3. There, some challenges regarding the coinductive definition of streams are laid out for further study.

2 Streams: Syntax and Semantics

The syntactic definition of streams (which will be denoted by the, possibly subsripted, letters s and t) is as follows:

$$s ::= [r]|X|(s+s)|(s \times s)|(s^{-1})|\text{IF } s \text{ THEN } s \text{ ELSE } s|v$$

Streams are to be interpreted as elements of $\mathbb{R}^\omega$ with one interpretation function defined on streams, namely $(\!|s|\!)_\mu$ which goes from the syntax (s) to the interpretation of that syntactical statement; the stream. Often it is usefull to split the resulting interpretation in two parts using the split function: $\text{Split}(s) : \mathbb{R}^\omega \to \mathbb{R} \times \mathbb{R}^\omega$. The $\mathbb{R}$ is the first element of the stream (the head), whereas the $\mathbb{R}^\omega$ is the rest of the stream (the tail). μ is the interpretation of the variable stream v, if any occur in s. The interpretation function is a function that assigns a certain stream to all occurences of a certain variable, or similarly for more variables.
Streams are built up out of one set of primitive streams, which are all streams with a real number as first element and zeros as all the other elements. Based on these primitive streams one further stream is defined directly, namely the X stream which is the stream with 0 as first element and the primitive 1 stream as the tail of the stream. From now on streams will be defined using the interpretation function and the Split function. In the following cases the interpretation is the unique stream that satisfies the equation specified using the Split function.

$$\text{Split}((\!|[r]|\!)_\mu) = \langle r, (\!|[0]|\!)_\mu \rangle$$

$$\text{Split}((\!|X|\!)_\mu) = \langle 0, (\!|[1]|\!)_\mu \rangle$$

Using these two streams the rest of the streams are built up using the operators which are given in the syntactic definition. First of all some additional functions and abbreviations are defined:

$$\text{inl}(\langle x, y \rangle) = x \qquad \text{inr}(\langle x, y \rangle) = y$$

$$s_0 = \text{inl}(\text{Split}((\!|s|\!)_\mu)) \quad s' = \text{inr}(\text{Split}((\!|s|\!)_\mu))$$

The stream operators are defined corecursively, using the interpretation function. This leads to the following definitions, where a subscript $\mathbb{R}$ indicates that the functions is that defined on real numbers:

$$\text{Split}((\!|(s+t)|\!)_\mu) = \langle s_0 +_\mathbb{R} t_0, s' + t' \rangle$$

Note that the corecursion takes place in the second element of the ordered pair, which is possible because s' is only a part of $\text{Split}((\!|s_1|\!)_\mu)$.

$$\text{Split}((\!|(s \times t)|\!)_\mu) = \langle s_0 \times_\mathbb{R} t_0, (s' \times (\!|t|\!)_\mu) + (s_0 \times t') \rangle$$

$$\text{Split}((\!|(s^{-1})|\!)_\mu) = \langle \frac{1}{s_0}, -\frac{1}{s_0} \times (s' \times (\!|s^{-1}|\!)_\mu) \rangle$$

In the last case it is necessary that $\text{inl}(\text{Split}((\!|s|\!)_\mu)) \neq 0$. The definition for the IF statement is:

$$(\!|\text{IF } s_1 \text{ THEN } s_2 \text{ ELSE } s_3|\!)_\mu = \begin{cases} (\!|s_2|\!)_\mu, & (\!|s_1|\!)_\mu = [1] \\ (\!|s_3|\!)_\mu, & \text{Otherwise} \end{cases}$$

Finally there is the variable stream v, which is assigned a meaning through the function μ:

$$(\!|v|\!)_\mu = \mu(v)$$

Thus, a variable stream is assigned a value through the μ function. The value of $\mu(v)$ is a stream with no occurences of the variable v and thus a subset of $\mathbb{R}^\omega$.

Lastly, it is necessary to introduce a syntactic definition for s_0 and s', which is done by defining it on all the operators. This definition the becomes:

$$[r]_0 = r \quad X_0 = 0 \quad (s+t)_0 = s_0 +_{\mathbb{R}} t_0 \quad (s \times t)_0 = s_0 \times_{\mathbb{R}} t_0 \quad (s^{-1})' = -\frac{1}{s_0} \times (s' \times s^{-1})$$

$$(\text{IF } s_1 \text{ THEN } s_2 \text{ ELSE } s_3)_0 = \begin{cases} (s_2)_0, & (\!|s_1|\!)_\mu = [1] \\ (s_3)_0, & \text{Otherwise} \end{cases}$$

$$v_0 = \mu(v)_0$$

$$[r]' = [0] \quad X' = [1] \quad (s+t)' = s' + t' \quad (s \times t)' = (s' \times t) + (s_0 \times t') \quad (s^{-1})_0 = \frac{1}{s_0}$$

$$(\text{IF } s_1 \text{ THEN } s_2 \text{ ELSE } s_3)' = \begin{cases} (s_2)', & (\!|s_1|\!)_\mu = [1] \\ (s_3)', & \text{Otherwise} \end{cases}$$

$$v' = \mu(v)'$$

Two streams s and t are equal if both $s_0 = t_0$ and $s' = t'$, which is explained in more detail in [Rutten, 2001, p.5-6] using the notion of a bisimulation.

3 Stream-types: Syntax and Semantics

The syntax for stream-types (denoted by the possibly subscripted greek letters τ and σ, preferring the first) is given as follows:

$$\tau ::= top|int|bool|Nonzero|X\tau|\tau_0|(\tau+\tau)|(\tau \times \tau)|(\tau^{-1})|(\tau \cup \tau)|F\tau$$

A stream-type is then to be interpreted as a subset of $\mathbb{R}^\omega$ with the following interpretation function defined on each type τ, going from syntax to a subset:

$$[\![\tau]\!] \subseteq \{s | s \in \mathbb{R}^\omega\}$$

Meaning that it takes the syntax to the collection of streams falling under that stream-type.[1] The primitive types, *top*, *int* and *bool* are defined directly by:

$$[\![top]\!] = \{s | s \in \mathbb{R}^\omega\}$$

$$[\![int]\!] = \{s | s_0 \in \mathbb{Z} \wedge s' = [0]\}$$

$$[\![bool]\!] = \{s | s_0 \in \{0,1\} \wedge s' = [0]\}$$

$$[\![Nonzero]\!] = \{s | s_0 \neq 0\}$$

Where [0] is the stream defined in such a way that it consists of only zero's. The first stream-type operator is then:

$$[\![X\tau]\!] = \{s \,|\, (\!|s'|\!)_\mu \in [\![\tau]\!]\}$$

So this stream is satisfied by all the streams that satisfy τ and are shifted one place. Now the rest of the operators can be defined:

$$\tau_0 = \{[s_0] \,|\, (\!|s|\!)_\mu \in [\![\tau]\!]\}$$

τ_0 is also defined syntactically, by the following rules:

$$top_0 = top \quad int_0 = int \quad bool_0 = bool \quad Nonzero_0 = top \backslash \{[0]\} \quad (X\tau)_0 = top \quad (\tau_0)_0 = \tau_0 \quad (\tau+\sigma)_0 = \tau_0 + \sigma_0$$

[1] The falling-under, or satisfies, relation will be defined more precisely later on

Here $top \setminus \{[0]\}$ is the usual complement operator as defined on sets.

$$(\tau \times \sigma)_0 = \tau_0 \times \sigma_0 \quad (\tau^{-1})_0 = (\tau_0)^{-1} \quad (\tau \cup \sigma)_0 = \tau_0 \cup \sigma_0 \quad (G\tau)_0 = \tau_0 \quad (F\tau)_0 = top$$

The rest op the operators are only defined semantically here, their syntactic behaviour is defined by the derivation rules for satisfaction of a type.

$$[\![\sigma + \tau]\!] = \{s | s_0 = p_0 +_{\mathbb{R}} q_0 \wedge s' = p' +_s q' \wedge (\!|p|\!)_\mu \in [\![\sigma]\!] \wedge (\!|q|\!)_\mu \in [\![\tau]\!]\}$$

Where $+_{\mathbb{R}}$ is the + operator defined for real numbers and $+_s$ is the + operator defined for streams.

$$[\![\sigma \times \tau]\!] = \{s | s_0 = p_0 \times_{\mathbb{R}} q_0 \wedge s' = (p' \times_s q) + (p_0 \times_s q') \wedge (\!|p|\!)_\mu \in [\![\sigma]\!] \wedge (\!|q|\!)_\mu \in [\![\tau]\!]\}$$

Where the $\times_s$ is the $\times$ operator as it is defined for streams and $\times_{\mathbb{R}}$ is the $\times$ operator as defined on the real numbers.

$$[\![\tau^{-1}]\!] = \{s | s_0 = \frac{1}{p_0} \wedge s' = -\frac{1}{p_0} \times_s (p' \times_s p^{-1}) \wedge (\!|p|\!)_\mu \in [\![\tau]\!]\}$$

Note that this stream is only defined if the first element is not equal to zero, that is if the stream is a subset of $Nonzero$. In the notation of the next section, this type is only defined if $\tau <: Nonzero$.

$$[\![\sigma \cup \tau]\!] = \{s | (\!|s|\!)_\mu \in [\![\sigma]\!] \vee (\!|s|\!)_\mu \in [\![\tau]\!]\}$$

The last stream-type operator cannot be defined directly, but is defined via a fixed point. It uses the set τ_0 defined above.

$F\tau$ is the least fixed point of the relation $f_\tau(S)$, where S is the set of streams that satisfy $F\tau$:

$$f_\tau(S) = \{s \in \mathbb{R}^\omega | (\!|[s_0]|\!)_\mu \in [\![\tau_0]\!] \vee s' \in S\}$$

After which the satisfies relation : which obtains iff a stream falls under a stream-type can be defined as the greatest fixed point of the function F where S is the set of all streams and T the set of all stream-types.

$$F(S,T) = \{\langle s, \tau \rangle | s \in S \wedge \tau \in T \wedge (\!|s|\!)_\mu \in [\![\tau]\!]\}$$

Both fixed points exist, since both functions are monotonic. If $S \subseteq T$ then $f_\tau(S) \subseteq f_\tau(T)$ since whenever $s' \in S$ is fulfilled, so is $s' \in T$ and $s_0 \in \tau_0$ is unaffected by S and T. For the function F it holds that when $S \subseteq S'$ then $F(S,T) \subseteq F(S',T)$ since $s \in S'$ whenever $s \in S$ and possibly more often. Similarly, when $T \subseteq T'$ then $F(S,T) \subseteq F(S,T')$ since $t \in T'$ whenever $t \in T$ and possibly more often. Thus, both functions are monotonic and because of that the fixed points exist.

4 Internal ordering of stream-types

Using the stream-type definitions it is possible to define an internal (partial) ordering between various of these types via the *subtype* operator <:, which is defined as follows:

$$\tau_1 <: \tau_2 = [\![\tau_1]\!] \subseteq [\![\tau_2]\!]$$

Two stream-types σ and τ are equal, which is written as $\tau =: \sigma$, when $\sigma <: \tau$ and $\tau <: \sigma$. For the subtyping relation various theorems and natural deduction style rules can be proved. I start here with the theorems.

$$\tau <: \tau$$

Since $[\![\tau]\!] = [\![\tau]\!]$, it is true that $[\![\tau]\!] \subseteq [\![\tau]\!]$ and thus that $\tau <: \tau$.

$$\tau_1 \cup \tau_2 <: \tau_2 \cup \tau_1$$

Since the set defined by $\tau_1 \cup \tau_2$ is equivalent to that defined by $\tau_2 \cup \tau_1$, something which follows from the commutative properties of the logical $\vee$ and the definition of the $\cup$ operator the above theorem holds by virtue of the definition of $\subseteq$. Similarly, the theorem $\tau_2 \cup \tau_1 <: \tau_1 \cup \tau_2$ holds.

$$\sigma \cup (\tau_1 \cup \tau_2) <: (\sigma \cup \tau_1) \cup \tau_2$$

Is a valid theorem, in virtue of the transitivity of the logical $\vee$, the definition of the $\cup$ operator and the properties of $\subseteq$. For the same reasons the converse theorem $(\sigma \cup \tau_1) \cup \tau_2 <: \sigma \cup (\tau_1 \cup \tau_2)$ also holds.

$$\sigma <: \sigma \cup \tau$$

Since the $\cup$ operator does not alter any of the streams in σ but only adds those in τ to them, all the streams that are in σ are also in $\sigma \cup \tau$. Thus, $[\![\sigma]\!] \subseteq [\![\sigma \cup \tau]\!]$ and the theorem holds.

$$\tau \cup \tau <: \tau$$

Is a valid theorem, since the $\cup$ operator does not alter the stream-types that it takes, except that the resultant stream-type is satisfied by all streams that satisfy either one of the argument stream-types. Since τ is satisfied by the same streams as itself, $\tau \cup \tau$ is equivalent to τ[2]. Thus, because of the properties of $\subseteq$ the theorem holds.

$$\sigma + \tau <: \tau + \sigma$$

This is true since $+_{\mathbb{R}}$ is associative and $+_s$ is associative[3]. Therefore, by the definition of $+$, $\sigma + \tau$ is equivalent to $\tau + \sigma$ and thus $\sigma + \tau <: \tau + \sigma$. For the primitive types there are some further valid rules:

$$int + int =: int \quad int \times int =: int \quad bool \times bool =: bool \quad bool <: int$$

Since int is the type containing all streams with an integer as first element that consist further of nothing but zero's, nothing happens to the streams when added except that the first element of a stream changes. Since $\mathbb{Z}$ is closed under addition no new streams will be formed by adding two streams in *int*. Nor will any streams in *int* disappear when added together, for each is duplicated by addition with the [0] stream. Therefore $int + int$ satisfies exactly the same streams as *int* and is thus a subtype of *int*. A similar argument can be made for the second and third cases, because neither $+$ nor $\times$ changes the tail of the streams satisifed by *bool* and *int* - so it remains [0]. Next to that, the first element stays within either $\mathbb{Z}$ or $\{0, 1\}$ and no elements are lost. For the last rule all one has to do is note that $\{0, 1\} \subset \mathbb{Z}$ and that $[0] = [0]$. There are also some valid theorems for *top*:

$$top + top =: top \quad top \times top =: top \quad top^{-1} = top \quad Ftop =: top$$

For the first three, all one has to note is that $\mathbb{R}^\omega$ is closed under addition, multiplication and inverse. This means that when all streams are added to all streams, no streams disappear. Similarly for multiplication and inverse, no streams disappear as long as one applies the operator to all streams. The last equality is simpler, if at some time I encounter an element of a stream then the stream satisfies $Ftop$. Since all streams satisfy this requirement, every stream falls under $Ftop$ and thus $Ftop =: top$. There is one more valid theorem for the primitive type *top*:

$$\tau <: top$$

This holds for any τ, since *top* is the type under which all streams fall, thus all the other types are subsets of *top* by the fact that only streams fall under those types. Thus, for any τ it is true that

[2]This can also be argued for via the properties of $\vee$ and the definition of the $\cup$ operator, but the current proof seems to me to be more insightfull

[3]This is proved in [Rutten, 2001, p.11]

$[\![\tau]\!] \subseteq [\![top]\!]$ and the theorem holds. Finally, there are also two theorems regarding the primitive type $Nonzero$:

$$Nonzero \times Nonzero =: Nonzero \quad Nonzero^{-1} =: Nonzero$$

The first holds because it is impossible to get a zero by multiplying two nonzero numbers and the second holds because it is impossible to get zero by taking the inverse of nonzero numbers. Because both operations don't reduce the amount of streams that satisfy the type, the types are equivalent.

$$\frac{\sigma_1 <: \tau_1 \qquad \sigma_2 <: \tau_2}{(\sigma_1 + \sigma_2) <: (\tau_1 + \tau_2)} +O$$

By assumption, $[\![\sigma_1]\!] \subseteq [\![\tau_1]\!]$ and $[\![\sigma_2]\!] \subseteq [\![\tau_2]\!]$. This means that every ordered pair that is in the interpretation of σ_1 is also in τ_1 and the same for σ_2 and τ_2. Thus, every combination of streams that can be made using the streams in the interpretation of σ_1 and σ_2 can also be made using the streams in the interpretation of τ_1 and τ_2. Since it are these combinations of streams that determine the elements of $[\![\sigma_1 + \sigma_2]\!]$ and $[\![\tau_1 + \tau_2]\!]$ and nothing else[4], every stream that is in the first is also in the second. Therefore, $[\![\sigma_1 + \sigma_2]\!] \subseteq [\![\sigma_1 + \sigma_2]\!]$ and thus $(\sigma_1 + \sigma_2) <: (\tau_1 + \tau_2)$.

$$\frac{\sigma_1 <: \tau_1 \qquad \sigma_2 <: \tau_2}{(\sigma_1 \times \sigma_2) <: (\tau_1 \times \tau_2)} \times O$$

Here again it holds that there is a function f such that $[\![\sigma_1 \times \sigma_2]\!] = f([\![\sigma_1]\!], [\![\sigma_2]\!])$ and the same (with the *same function*) for τ_1 and τ_2. Since by assumption $[\![\sigma_1]\!] \subseteq ([\![\tau_1]\!]$ and $[\![\sigma_2]\!] \subseteq [\![\tau_2]\!]$ it also holds that $[\![\sigma_1 \times \sigma_2]\!] \subseteq [\![\tau_1 \times \tau_2]\!]$. Hence, it holds that $(\sigma_1 \times \sigma_2) <: (\tau_1 \times \tau_2)$.

$$\frac{\tau_1 <: \tau_2 \qquad \tau_1 <: Nonzero \qquad \tau_2 <: Nonzero}{(\tau_1)^{-1} <: (\tau_2)^{-1}} -1O$$

By assumption, $[\![\tau_1]\!] \subseteq [\![\tau_2]\!]$ and from the definition of the $^{-1}$ operator (which is defined for these types as ensured by the other two assumptions), it follows that there is a function f such that $[\![(\tau_1)^{-1}]\!] = f([\![\tau_1]\!])$ and the same is true, with the same function f, for τ_2. Since f has to produce the same values for the same arguments (this is mandatory for all functions), it is true that $f([\![\tau_1]\!]) \subseteq f([\![\tau_2]\!])$ and thus that $[\![(\tau_1)^{-1}]\!] \subseteq [\![(\tau_2)^{-1}]\!]$. From this it follows that $(\tau_1)^{-1} <: (\tau_2)^{-1}$.

$$\frac{\sigma <: \tau}{F\sigma <: F\tau} F\tau O$$

When $\sigma <: \tau$ it holds that $[\![\sigma]\!] \subseteq [\![\tau]\!]$ and thus that $\sigma_0 \subseteq \tau_0$. Thus, whenever a stream s fulfills $s_0 \in \sigma_0$ it also fulfills $s_0 \in \tau_0$. And, since $[\![\sigma]\!] \subseteq [\![\tau]\!]$ if s fulfills $s' \in S$ with S the set of streams that satisfy $F\sigma$, s also fulfills $s' \in T$ with T the set of streams that satisfy $F\tau$. Thus, in both cases $[\![F\sigma]\!] \subseteq [\![F\tau]\!]$ and therefore $F\sigma <: F\tau$, whence the inference is valid.

$$\frac{\sigma <: \tau \qquad \tau <: \rho}{\sigma <: \rho} Trans$$

This rule holds because by assumption $[\![\sigma]\!] \subseteq [\![\tau]\!]$ and $[\![\tau]\!] \subseteq [\![\rho]\!]$. Since $\subseteq$ is transitive, it thus follows that $[\![\sigma]\!] \subseteq [\![\rho]\!]$ and hence that $\sigma <: \rho$.
Using the above proofs it is possible to infer a **Soundness Theorem**, namely that if the above proof rules prove $\sigma <: \tau$ then $[\![\sigma]\!] \subseteq [\![\tau]\!]$. This follows by induction on the three proof rules (+, $\times$ and -1) and the theorems, since for each of those it has been shown that there is no mismatch between the syntactic proofs that can be given and the interpretation behind those proofs.

[4]This holds since there is a function f such that $[\![\sigma_1 + \sigma_2]\!] = f([\![\sigma_1]\!], [\![\sigma_2]\!])$ and similarly, with the same function f, for τ_1 and τ_2

5 Satisfaction of a stream-type by a stream

Before staring with the deduction rules, it is important to define exactly what will stand in these deduction rules. Each term will be of the form $\Gamma \vdash t : \tau$. Where Γ is a set of assumptions of the form $v : \sigma$ for variables v that may (but do not have to) occur in t. A variable v may occur at most once in Γ. The semantics of these terms is as follows: $\Gamma \vdash t : \tau$ holds iff $\forall \mu : Var \to \mathbb{R}^\omega$ such that $(\forall v_i : \sigma_i \in \Gamma)\, \mu(v_i) \in [\![\tau_i]\!]$ implies that $(\!|s|\!)_\mu \in [\![\tau]\!]$. One obvious consequence of these definitions is that

$$v : \sigma \vdash v : \sigma$$

Which holds for any variable v and any type σ and basically says that if I assume something, then that same thing is derivable from the assumption.

Using the definition of the satisfies relation and those of the stream-types, it is possible to prove the validity of a set of natural deduction rules that specify how a proof that a certain stream satisfies a certain type can be constructed. The goal here is to prove that the stream-types, except for the $\cup$ and F operator, match the streams in that when a stream syntactically mirrors a stream-type it also satisfies that stream-type. The $\cup$ operator then serves to group various streams under one stream-type and is used to deal with the if-statement for streams. The F operator is used to be able to group streams according to the kind of elements that are in those streams.

$$\frac{r \in \mathbb{Z}}{\emptyset \vdash [r] : int}\, int$$

The proof for this rule is fairly simple since $\text{Split}((\!|[r]|\!)_\mu) = \langle r, [0]\rangle$ and $[\![int]\!] = \{s | s_0 \in \mathbb{Z} \wedge s' = [0]\}$. Since $r \in \mathbb{Z}$, $(\!|[r]|\!)_\mu \in [\![int]\!]$ and thus that $[r] : int$.

$$\frac{r \in \{0, 1\}}{\emptyset \vdash [r] : bool}\, bool$$

The proof for this rule is fairly simple since $\text{Split}((\!|[r]|\!)_\mu) = \langle r, [0]\rangle$ and $[\![bool]\!] = \{s | s_0 \in \{0, 1\} \wedge s' = [0]\}$. Since $r \in \{0, 1\}$, $(\!|[r]|\!)_\mu \in [\![bool]\!]$ and thus that $[r] : bool$.

$$\frac{}{\emptyset \vdash s : top}\, top$$

Where s can be either a concrete stream or a stream with variables. This holds because s has to be an element of $\mathbb{R}^\omega$ because s is a stream and all elements of $\mathbb{R}^\omega$ are in top.

$$\frac{s_0 \neq 0}{\emptyset \vdash s : Nonzero}\, Nonzero$$

By assumption $s_0 \neq 0$, thus $(\!|s|\!)_\mu \in [\![Nonzero]\!]$ according to the definition of $Nonzero$ and thus $s : Nonzero$. Hence, the inference rule is valid.

$$\frac{t : \tau}{\emptyset \vdash X \times t : X\tau}\, X\tau$$

It is true that $[\![X\tau]\!] = \{s \,|\, (\!|s'|\!)_\mu \in [\![\tau]\!]\}$, from which it easily follows with $(\!|t|\!)_\mu = (X \times t)'$ and $(\!|t|\!)_\mu \in [\![\tau]\!]$ that $(\!|X \times t|\!)_\mu \in [\![X\tau]\!]$ and thus $X \times t : X\tau$.

$$\frac{\Gamma \vdash s : \sigma \qquad \Gamma \vdash t : \tau}{\Gamma \vdash (s + t) : (\sigma + \tau)}\, +$$

It is true that $(\!|s|\!)_\mu \in [\![\sigma]\!]$ and $(\!|t|\!)_\mu \in [\![\tau]\!]$. Given that, the definition of the + operator for streams can be seen to operate in exactly the same way on two given ordered pairs as the definition of + for stream-types. Therefore there exists a function f (namely, exactly that from the stream definition) such that $f((\!|s|\!)_\mu, (\!|t|\!)_\mu) = (\!|s + t|\!)_\mu \in [\![\sigma + \tau]\!] = \{s \,|\, \text{Split}(s) = f(a, b) \wedge a \in [\![\sigma]\!] \wedge b \in [\![\tau]\!]\}$. Which yields $(\!|s + t|\!)_\mu \in [\![\sigma + \tau]\!]$ and thus $\Gamma \vdash (s + t) : (\sigma + \tau)$.

$$\frac{\Gamma \vdash s : \sigma \qquad \Gamma \vdash t : \tau}{\Gamma \vdash (s \times t) : (\sigma \times \tau)} \times$$

It holds that $(\!|s|\!)_\mu \in [\![\sigma]\!]$ and $(\!|t|\!)_\mu \in [\![\tau]\!]$. Given that, the definition of the $\times$ operator for streams can be seen to operate in exactly the same way on two given ordered pairs as the definition of $\times$ for stream-types. Therefore there exists a function f (namely, exactly that from the stream definition) such that $f((\!|s|\!)_\mu, (\!|t|\!)_\mu) = (\!|s \times t|\!)_\mu \in [\![\sigma \times \tau]\!] = \{s \,|\, \text{Split}(s) = f(a,b) \wedge a \in [\![\sigma]\!] \wedge b \in [\![\tau]\!]\}$. Which yields $(\!|s \times t|\!)_\mu \in [\![\sigma \times \tau]\!]$ and thus $\Gamma \vdash (s \times t) : (\sigma \times \tau)$.

$$\frac{\Gamma \vdash t : \tau \qquad \Gamma \vdash t : Nonzero \qquad \tau <: Nonzero}{\Gamma \vdash (t^{-1}) : (\tau^{-1})} -1$$

It holds that $(\!|t|\!)_\mu \in [\![\tau]\!]$ and because $(\!|t|\!)_\mu \in [\![Nonzero]\!]$ it is certain that t^{-1} is defined and thanks to $\tau <: Nonzero$ it is certain that τ^{-1} is defined. Given that, the definition of the $^{-1}$ operator for streams can be seen to operate in exactly the same way on two given ordered pairs as the definition of $^{-1}$ for stream-types. Therefore there exists a function f (namely, exactly that from the stream definition) such that $f((\!|t|\!)_\mu) = (\!|t^{-1}|\!)_\mu \in [\![\tau^{-1}]\!] = \{s \,|\, \text{Split}(s) = f(a) \wedge a \in [\![\tau]\!]\}$. Which yields $(\!|t^{-1}|\!)_\mu \in [\![\tau^{-1}]\!]$ and thus $\Gamma \vdash (t^{-1}) : (\tau^{-1})$.

$$\frac{\Gamma \vdash s_2 : \sigma \qquad \Gamma \vdash s_3 : \tau}{\Gamma \vdash \text{IF } s_1 \text{ THEN } s_2 \text{ ELSE } s_3 : \sigma \cup \tau} \text{ IF}$$

It is clear from the definition of IF that it denotes either the stream s_2 or the stream s_3. If $(\!|s_1|\!)_\mu = [1]$ then it denotes s_2. Then, by assumption, $s_2 : \sigma$. Hence, via W (see below) and $\sigma <: \sigma \cup \tau$,[5] $s_2 : \sigma \cup \tau$. In this case then IF s_1 THEN s_2 ELSE $s_3 : \sigma \cup \tau$. In the other case, when IF denotes s_3, the assumption that $s_3 : \tau$ is used. For, from that assumption, W and $\tau <: \sigma \cup \tau$ it follows that $s_3 : \sigma \cup \tau$ and thus that IF s_1 THEN s_2 ELSE $s_3 : \sigma \cup \tau$. Hence, since all cases lead to the same result it holds in general that $\Gamma \vdash$ IF s_1 THEN s_2 ELSE $s_3 : \sigma \cup \tau$.

$$\frac{\Gamma \vdash [s_0] : \tau_0}{\Gamma \vdash s : F\tau} F\tau$$

By assumption $[s_0] : \tau_0$, which means that $(\!|[s_0]|\!)_\mu \in [\![\tau_0]\!]$ and thus, by the definition of $F\tau$ that $(\!|s|\!)_\mu \in [\![F\tau]\!]$ and $\Gamma \vdash s : F\tau$. Therefore, the inference is valid.

$$\frac{\Gamma \vdash s' : F\tau}{\Gamma \vdash s : F\tau} F\tau$$

By assumption, $(s')_0 \in \tau_0$, which means that in the function $f_\tau(S)$ $s' \in S$ and thus that $s \in S$. By virtue of the definition of $F\tau$ as the least fixed point of $f_\tau(S)$ it thus holds that $(\!|s|\!)_\mu \in [\![F\tau]\!]$ and $\Gamma \vdash s : F\tau$. Thus, the inference is valid.

$$\frac{\Gamma \vdash t : \sigma \qquad \sigma <: \tau}{\Gamma \vdash t : \tau} \text{ W}$$

This inference rule allows one to weaken the inferenced satisfaction by choosing a larger type (a weaker consequence for $\vdash$), hence W. The validity is established by noting that $(\!|t|\!)_\mu \in [\![\sigma]\!]$ and $[\![\sigma]\!] \subseteq [\![\tau]\!]$ by assumption. Therefore, by the properties of $\subseteq$, $(\!|t|\!)_\mu \in [\![\tau]\!]$. Thus, $\Gamma \vdash t : \tau$.

$$\frac{\Gamma, v : \tau \vdash s : \sigma \qquad \tau_1 <: \tau}{\Gamma, v : \tau_1 \vdash s : \sigma} \text{ S}$$

Here it is presumed that the assignment function μ makes it true that $(\!|v|\!)_\mu \in [\![\tau]\!]$. Since μ assigns to v a specific stream, of which nothing is demanded here than that it falls in τ, the $\vdash$ remains valid when the demand on μ is strengthened. After all, v still satisfies τ when it satisfies τ'.

[5] This follows from $\sigma <: \sigma$ when put into $\cup O$

$$\frac{\Gamma \vdash t : \tau}{\Gamma, v : \sigma \vdash t : \tau}\ \mathrm{A}$$

This obviously holds, since the added assumption can only narrow the amount of streams that t can signify. Since $t : \tau$ was already valid with fewer assumptions, this extra assumption does nothing to the validity of $t : \tau$. For this it is necessary however that the extra assumption does not conflict with any of the assumptions that are already in Γ. Therefore, the inference is valid.

$$\frac{\Gamma \vdash s : \sigma \qquad \Gamma, v : \sigma \vdash t : \tau}{\Gamma \vdash t[s/v] : \tau}\ \mathrm{Inst}$$

Since s satisfies all the requirements laid upon the variable v, it is allowed to substitute s for v in virtue of the definition of $\vdash$ given at the beginning of this section. Basically, s is one of the values v is given by μ that satisfies the assumptions. Thus, $\Gamma \vdash t[s/v] : \tau$ is true because $\Gamma, v : \sigma \vdash t : \tau$ is true. Hence, the inference is valid.

Because of all the proofs of validity of the inferences, it can be concluded that whenever it is derivable that a stream satisfies a type under a certain assumption then the interpretation of that stream is also a member of the interpretation of that type. Thus, this inference system is **Sound**, just like the system presented in section 4.

6 Examples

A first example is provided by the following:

$$\emptyset \vdash \mathrm{IF}\, v\, \mathrm{THEN}\, X \times [3]\, \mathrm{ELSE}\, X \times X \times ([1] + [2])\ : Fint$$

The proof of this proceeds in three steps: First, prove that $\emptyset \vdash X \times [3] : Fint$. Second, prove that $\emptyset \vdash X \times X \times ([1] + [2]) : Fint$. Third, use the type ordering theorems to achieve the desired result. The first step goes as follows: $[[3]_0] = [3]$ and

$$\frac{3 \in \mathbb{Z}}{\emptyset \vdash [3] : int}\ \mathrm{int}$$

So, $\emptyset \vdash [[3]_0] : int$. Since also $(X \times [3])' = (X' \times [3]) + (X_0 \times [3]') = ([1] \times [3]) + (0 \times [0]) = [3]$ and $int_0 = int$ it holds that

$$\frac{\dfrac{\emptyset \vdash [[3]_0] : int_0}{\emptyset \vdash [3] : Fint}\ Fint}{\emptyset \vdash X \times [3] : Fint}\ Fint$$

The second step is somewhat more elaborate, but analogous. By the inference above, $\emptyset \vdash [3] : int$ and $\emptyset \vdash X \times [3] : Fint$. Now, $(X \times (X \times [3]))' = (X' \times (X \times [3])) + (X_0 \times (X \times [3])') = ([1] \times (X \times [3])) + (0 \times [3]) = X \times [3]$. Since we know that $\emptyset \vdash X \times [3] : Fint$ it is possible to conclude that

$$\frac{\emptyset \vdash X \times [3] : Fint}{\emptyset \vdash X \times X \times [3] : Fint}\ \mathrm{Fint}$$

Now, since $([1] + [2])_0 = ([1]_0 + [2]_0) = (1 + 2) = 3 = [3]_0$ and $([1] + [2])' = ([1]' + [2]') = ([0] + [0]) = [0] = [3]'$, $[1] + [2] = [3]$ and thus, by substitution of equals, $\emptyset \vdash X \times X \times ([1] + [2]) : Fint$. For the third step, one first needs to make the following inference

$$\frac{\emptyset \vdash X \times [3] : Fint \qquad \emptyset \vdash X \times X \times ([1] + [2]) : Fint}{\emptyset \vdash \mathrm{IF}\, v\, \mathrm{THEN}\, X \times [3]\, \mathrm{ELSE}\, X \times X \times ([1] + [2])\ : Fint \cup Fint}\ \mathrm{IF}$$

Finally, by the type ordering theorems, $Fint \cup Fint <: Fint$ and thus by W:

$$\frac{\emptyset \vdash \text{IF } v \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint \cup Fint \qquad Fint \cup Fint <: Fint}{\emptyset \vdash \text{IF } v \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint} \text{ W}$$

A second example that builds on the first is gained by instantiating the variable in the if-statement. For example, one can prove that

$$\emptyset \vdash \text{IF } (1-X)^{-1} \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint$$

This proceeds in two steps. First:

$$\frac{}{\emptyset \vdash (1-X)^{-1} : top} \ top$$

And second:

$$\frac{\emptyset \vdash \text{IF } v \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint}{v : top \vdash \text{IF } v \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint} \text{ A}$$

Then the result is obtained using Inst:

$$\frac{\emptyset \vdash (1-X)^{-1} : top \qquad v : top \vdash \text{IF } v \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint}{\emptyset \vdash \text{IF } (1-X)^{-1} \text{ THEN } X \times [3] \text{ ELSE } X \times X \times ([1]+[2]) \ : Fint} \text{ Inst}$$

7 A possible extension: $G\tau$

The system presented up until now works well, but misses one operator that would be useful. Whereas the $F\tau$ operator indicates that there is an element that eventually falls within a certain range, right now there is no operator to say that every element falls within a certain range. This can be done however, by introducing an operator $G\tau$. The only problem is that as soon as one tries to prove that a stream satisfies this operator one runs into difficulties. These are presented below using an example, after all the definitions and relevant proofs have been given.
$G\tau$ is the greatest fixed point of the function $g_\tau(S)$, where S is the set of streams that satisfy $G\tau$:

$$g_\tau(S) = \{s \in \mathbb{R}^\omega | (\!|[s_0]|\!)_\mu \in [\![\tau_0]\!] \wedge s' \in S\}$$

The fixed point exists, since the function is monotonic. If $S \subseteq T$ then $g_\tau(S) \subseteq g_\tau(T)$ since whenever $s' \in S$ is fulfilled, so is $s' \in T$ and $s_0 \in \tau_0$ is unaffected by S and T.
For this operator two rules hold, one with respect to type ordering and one with respect to stream satisfaction.

$$\frac{\sigma <: \tau}{G\sigma <: G\tau} \ G\tau O$$

When $\sigma <: \tau$ it holds that $[\![\sigma]\!] \subseteq [\![\tau]\!]$ and thus that $\sigma_0 \subseteq \tau_0$. Thus, whenever a stream s fulfills $s_0 \in \sigma_0$ it also fulfills $s_0 \in \tau_0$. And, since $[\![\sigma]\!] \subseteq [\![\tau]\!]$ if s fulfills $s' \in S$ with S the set of streams that satisfy $G\sigma$, s also fulfills $s' \in T$ with T the set of streams that satisfy $G\tau$. Thus, whenever a stream fulfills both conditions for satisfying $G\sigma$ it also fulfills both conditions for satisfying $G\tau$. Therefore, $[\![G\sigma]\!] \subseteq [\![G\tau]\!]$ and thus $G\sigma <: G\tau$, and the inference is valid.

$$\frac{\Gamma \vdash [s_0] : \tau_0 \qquad \Gamma \vdash s' : G\tau}{\Gamma \vdash s : G\tau} \ G\tau$$

The first demand for $s : G\tau$ is that $s_0 \in \tau_0$, which is met by virtue of the assumption that $[s_0] : C\tau$ and the definition of $C\tau$. The second demand, that $s' \in S$ with S the set of streams that satisfy $G\tau$ is met by the assumption that $s' : G\tau$. Therefore, $\Gamma \vdash s : G\tau$ and thus the inference is valid.

An example to illustrate the difficulty in reasoning with $G\tau$ is the stream $\frac{1}{1-X}$, or $(1-X)^{-1}$. First,

$$((1-X)^{-1})_0 = \frac{1}{(1-X)_0} = \frac{1}{[1]_0 - X_0} = \frac{1}{1-0} = 1$$

Here clearly $[((1-X)^{-1})_0] : bool$, since $1 \in \{0,1\}$ and via an application of $bool$. Second, it is necessary to calculate the tail of the stream:

$$(\frac{1}{1-X})' = \frac{-(1-X)'}{(1-X)_0 \times (1-X)} = \frac{-([0]+[-1])}{(1-0)\times(1-X)} = \frac{[1]}{1-X} = (1-X)^{-1}$$

This leads to the following inference according to the rule $G\tau$:

$$\frac{\emptyset \vdash [((1-X)^{-1})_0] : bool \qquad \emptyset \vdash (1-X)^{-1} : Gbool}{\emptyset \vdash (1-X)^{-1} : Gbool}\, G\tau$$

Clearly this inference is not permissible, since the second premiss has to be established before the conclusion can be established. However, this can only be done by an extra application of the $G\tau$ rule and this quickly leads to an infinite regress. Thus, the $G\tau$ rule leads to infinite grounding trees, something which is not standardly allowed. This means that if $G\tau$ is to be added to the typing system, work has to be done regarding the structure of proof trees. Unfortunately this issue is far too complicated to tacklle in this paper, so it has to be left as an open problem here. One thing that can be said though is that this problem arises because the current proof structure used is inductive, reducing elaborate cases on a step-by-step basis to a certain base-case. Streams however are defined co-inductively and thus work in the exact opposite direction. Next to that, a way has to be found to deal with the infinity of streams, for as the $G\tau$ operator shows finite proof trees are not sufficient.

8 Conclusion

This paper has presented a first step towards a full apparatus for typing streams. In order to do this, a simple language for stream manipulation was set up in section 2 with specifically an if-statement and a variable stream. Then, in section 3, a language for typing these streams was presented which incorporated among other types a union type in order to deal with the if-statement for streams.

In order to be able to use this language to actually group streams under types two additional steps were taken. First, a way of ordering the types using the subtyping operator $<:$ was given. This allows for more flexibility in proving the satisfaction of stream-types by streams when using the apparatus developed in section 5. This apparatus was the second step and provides a set of deduction rules which can be used to prove that a certain stream satisfies a certain stream-type. The use of these rules has been illustrated in section 6.

All in all the apparatus that has been developed in this paper works quite well and provides a good ground on which more elaborate systems for stream manipulation and stream typing can be build. One point where the system clearly misses the point though was illustrated in section 7, by the $G\tau$ operator. This showed that if one wants to say something about every single element of the stream, then this cannot be done with the standardly used finite proof trees. There are two problems here, one is the misfit between the coinductive definition of streams and the inductive way that proof trees work. The other is the misfit between the infinite character of streams and the finite character of proof trees. However, these problems are not so severe as to prevent any typing of streams, as this paper has tried to show. Finite proof trees can in fact be used to type streams, but their use is limited.

References

J.J.M.M. Rutten. Elements of Stream Calculus: (An Extensive Exercise in Coinduction). *Electronic Notes in Theoretical Computer Science*, 45(0):358–423, November 2001. ISSN 1571-0661. doi: http://dx.doi.org/10.1016/S1571-0661(04)80972-1. URL `http://www.sciencedirect.com/science/article/pii/S1571066104809721`.