Unique Solutions to Stream Specifications

A.E.M. Koenraadt (0567024) a.e.m.koenraadt@student.tue.nl

Department of Mathematics and Computer Science
University of Technology Eindhoven

February 27, 2010

Abstract

Streams are infinite sequences of data elements. Systems of recursive equations may or may not admit a unique solution. We study a limited signature for systems of recursive equations and propose a method to determine whether a system of recursive equations admits a unique solution.

Contents

1 Introduction

A stream over a data set $\mathbf{A}$ is an infinite sequence of elements from $\mathbf{A}$, i.e. an element from $\mathbf{A}^\omega$. Streams are defined via recursive equations. For example, the stream consisting of alternating 0's and 1's, starting with 0, for a data set $\mathbf{A} = \{0, 1\}$ can be defined by the equation $X = 0 : 1 : X$. Additional operators may be used to describe streams more succinctly or to describe streams that can not be described otherwise. A system of recursive equations may or may not admit a unique solution.

A system of recursive equations over some data set adheres to the following rules:

1. A system of recursive equations consists of a *finite* number of recursive equations.
2. A recursive equation is described by the signature below. The start-symbol for recursive equations is $\langle equation \rangle$.

$$\begin{aligned} \langle data \rangle &::= a, b, c, ... \\ \langle recvar \rangle &::= X, X_i, Y, Y_i, Z, Z_i ... (i \in \mathbb{N}) \\ \langle term \rangle &::= \langle recvar \rangle \mid zip(\langle term \rangle, \langle term \rangle) \mid \langle data \rangle : \langle term \rangle \\ \langle equation \rangle &::= \langle recvar \rangle = \langle term \rangle \end{aligned}$$

3. A recursion variable ($\langle recvar \rangle$) is defined by exactly one recursive equation.

Note that no operators on the data set will be used. We use greek letters $\sigma, \sigma_1, \sigma_2, \sigma_A, \sigma_B, ...$ to denote specific streams and $\mathcal{S}_*, \mathcal{T}_*$ to denote specific terms. Let E be a system of recursive equations over dataset $\mathbf{A}$. We denote the set of al recursion variables in E by $Vars(E)$ and the set of all terms that may be constructed with elements in $\mathbf{A}$ and variables in E according to $\langle term \rangle$ by $Terms(E)$. Note that $Vars(E) \subseteq Terms(E)$.

A system of recursive equations is intended to define for every recursion variable a stream. Streams can be interpreted as maps from the natural numbers $\mathbb{N}$ to $\mathbf{A}$, with $\mathbf{A}$ the set of elements that may occur in the system of recursive equations. A solution to a system of recursive equations is a mapping $\phi : Vars(E) \to \mathbf{A}^\omega$ that maps each recursion variable to a stream. Given that we interpret streams as maps from the natural numbers $\mathbb{N}$ to $\mathbf{A}$, we denote selection of the i-th element in the stream σ as follows: $\sigma(i)$. We extend the mapping ϕ to arbitrary terms as follows:

$$\begin{aligned} \phi(a : \mathcal{T}_x)(0) &= a \\ \phi(a : \mathcal{T}_x)(i+1) &= \phi(\mathcal{T}_x)(i) \\ \phi(zip(\mathcal{T}_x, \mathcal{T}_y))(2i) &= \phi(\mathcal{T}_x)(i) \\ \phi(zip(\mathcal{T}_x, \mathcal{T}_y))(2i+1) &= \phi(\mathcal{T}_y)(i) \end{aligned}$$

We now use this mapping to define the notion of well-definedness.

Definition 1. *(Equality) Two streams, σ_1, σ_2, are equal if and only if*

$$(\forall n \in \mathbb{N} : \sigma_1(n) \equiv \sigma_2(n))$$

Definition 2. *(Well-definedness) A system of recursive equations E over a* $\mathbf{A}$ *is well-defined if and only if it admits exactly one solution*

$$(\exists! \phi : Vars(E) \to \mathbf{A}^\omega :: (\forall X = \mathcal{T}_X \in E :: \phi(X) \equiv \phi(\mathcal{T}_X)))$$

The question we wish to answer: is a set of recursive equations well-defined by the equations given to describe it, i.e. is there a unique solution for every stream described by the recursive equations?

In the following sections, we will study systems of recursive equations over the action-prefix operator (:) and function *zip*. We introduce three properties, called `zip-cycle-freeness`, `guardedness` and `observability` and will show that these three properties are all equivalent to `well-definedness`. The structure is as follows:

- Section 2 introduces `zip-cycle-freeness` and shows that zip-cycle-freeness can be verified in linear time with respect to the number of recursive equations in the system (in Section 2.1).
- Section 3 introduces `guardedness` and shows that guardedness is equivalent to zip-cycle-freeness (in Section 3.1).
- Section 4 introduces `observability` and shows that observability is equivalent to guardedness (in Section 4.1) and that observability is equivalent to well-definedness (in Section 4.2), which concludes the proof of equivalence between the four properties of `zip-cycle-freeness`, `guardedness`, `observability` and `well-definedness`.

2 Zip-cycle-freeness

A zip-cycle is a certain type of cyclic dependency between the recursive equations.
`zip-cycle-freeness` is a property on the recursion in a system of recursive equations. The property expresses that a recursion variable X_i is defined such that it depends on only itself and no action prefixes. First, we introduce a shorthand notation for nested *zip*-applications:

$$\begin{aligned} zip^0(X) &= X \\ zip^1(X, Y_1) &= zip(X, Y_1) \\ zip^{n+1}(X, Y_1, .., Y_{n+1}) &= zip(zip^n(X, Y_1, ..Y_n), Y_{n+1}) \end{aligned}$$

Example: $zip^2(X, Y, Z) = zip(zip(X, Y), Z)$.

Next, we introduce the notion of a zip-cycle.

Definition 3. *(Zip-cycle) A zip-cycle is a set of recursive equations over a set of recursion variables $\{X_0, ..., X_n\} \subseteq Vars(E)$ such that the defining equations of this set are of the following shape:*

$$\begin{aligned} X_0 &= zip^{m_0}(X_1, \mathcal{T}_{(X_0,1)}, ..., \mathcal{T}_{(X_0,m_0)}) \\ X_1 &= zip^{m_1}(X_2, \mathcal{T}_{(X_1,1)}, ..., \mathcal{T}_{(X_1,m_1)}) \\ &... \\ X_{n-1} &= zip^{m_{n-1}}(X_n, \mathcal{T}_{(X_{n-1},1)}, ..., \mathcal{T}_{(X_{n-1},m_{n-1})}) \\ X_n &= zip^{m_n}(X_0, \mathcal{T}_{(X_n,1)}, ..., \mathcal{T}_{(X_n,m_n)}) \end{aligned}$$

where $\mathcal{T}_{(X_i,n)} \in Terms(E)$ and $m_0, ..., m_n \in \mathbb{N}$.

Lastly, a system of recursive equations is zip-cycle-free if no zip-cycles exist.

Definition 4. *(Zip-cycle-freeness) A system of recursive equations is zip-cycle-free if and only if it contains no zip-cycles.*

2.1 Deciding zip-cycle-freeness

Below, an algorithm is given to decide `zip-cycle-freeness`. First, recognize that every recursive equation is of the shape $X = zip^n(\mathcal{T}_s, \mathcal{T}_{(X,1)}, ..., \mathcal{T}_{X,n})$ for some $n \in \mathbb{N}$ such that $\mathcal{T}_s$ is either a recursion variable or an action prefix operation. `zip-cycle-freeness` can be decided in $O(\sum_{Y \in Vars(E)} \#zip(Y))$, where $\#zip(X) \in \mathbb{N}$ is the number such that $X = zip^{\#zip(X)}(\mathcal{T}_s, \mathcal{T}_{(X,1)}, ..., \mathcal{T}_{X,\#zip(X)})$ holds.

```
1:  procedure DECIDE ZIP-CYCLE-FREENESS(E)
2:      W ← E
3:      mark all equations as unvisited
4:      while there are unvisited equations in W do
5:          choose arbitrary X = T_X ∈ W
6:          mark X = T_X as visited
7:          S ← {X}
8:          while S ≠ ∅ do
9:              while T_X ≡ zip(T_s, T_t) do
10:                 T_X ← T_s                    ▷ Strip away all zip function applications
11:             end while
12:             if T_X ≡ Y for some Y = T_Y ∈ W then
13:                 if Y ∈ S then
14:                     return false             ▷ there is a zip-cycle containing Y
15:                 else                                                    ▷ Y ∉ S
16:                     S ← S ∪ Y
17:                     T_X ← T_Y
18:                     mark Y = T_Y as visited
19:                 end if
20:             else if T_X = a : T_s then
21:                 S ← ∅
22:             end if
23:         end while
24:     end whilereturn true
25: end procedure
```

This procedure is at worst linear in the size of the equations. Since there can not be a cheaper method (equations must be parsed at least), this is as efficient as it can be.

3 Guardedness

`guardedness` is a notion that is seen in many forms in the literature, as recapped in [3] (*Related Work*). For our purposes, however, the notion of guardedness as considered in process algebraic approaches is sufficient.

Definition 5. *(Guardedness) Given a system of recursive equations E, we have a set of rewrite rules*

$$R_E = \{\ zip(a : xs, ys) \rightarrow a : zip(ys, xs)\ \} \cup \{\ X \rightarrow \mathcal{T}_X \mid X = \mathcal{T}_X \in E\ \}$$

- *A term $\mathcal{T}_s$ can be rewritten by applying the set of rewrite rules R_E to the term itself.*
- *An equation can be rewritten by rewriting the term in the right-hand side.*
- *A system of recursive equations E can be rewritten by rewriting its equations.*

Given the set of rewrite rules and rewriting operations, we have the notion of guardedness:

- *A term $\mathcal{T}_s$ is called completely guarded if the outermost operator is $(:)$, i.e. the term is of the form $a : \mathcal{T}_t$ for some other term $\mathcal{T}_t$.*
- *Term $\mathcal{T}_s$ is called guarded if and only if it can be rewritten into a completely guarded term, using the set of rewrite rules R_E.*
- *A system of recursive equations E is called completely guarded if and only if all right-hand sides of all recursive equations of E are completely guarded.*
- *A system of recursive equations is guarded if and only if it can be rewritten into a completely guarded system of recursive equations, using the rewrite rules that hold for that system.*

We introduce the following predicates over systems of recursive equations for a more concise notation:

$$\texttt{GUARDED}(E) \equiv (\forall X = \mathcal{T}_X \in E : (\exists a \in \mathbf{A}, \mathcal{T}_u \in Terms(E) :$$

$$\mathcal{T}_X \text{ can be rewritten using the set of rewrite rules } R_E \text{ to } a : \mathcal{T}_u))$$

3.1 Guardedness equivalent to zip-cycle-freeness

In this section, we will show that `guardedness` is equivalent to `zip-cycle-freeness`. First, we show a property of zip-cycles that we will use to prove the equivalence.

Lemma 6. *(Rewriting Zip-cycles) If a system of recursive equations E contains a zip-cycle over the set of recursion variables $\{X_0, ..., X_n\} \subseteq Vars(E)$, using the set of rewrite rules R_E we can rewrite this cycle to $n+1$ separate zip-cycles over $\{X_i\}$ for each $X_i \in Vars(E), 0 \leq i \leq n$.*

Proof. A zip-cycle is a set of recursion variables $\{X_0, ..., X_n\} \subseteq Vars(E)$ such that the defining equations of this set are of the following shape:

$$\begin{aligned} X_0 &= zip^{m_0}(X_1, \mathcal{T}_{(X_0,1)}, ..., \mathcal{T}_{(X_0,m_0)}) \\ X_1 &= zip^{m_1}(X_2, \mathcal{T}_{(X_1,1)}, ..., \mathcal{T}_{(X_1,m_1)}) \\ &... \\ X_{n-1} &= zip^{m_{n-1}}(X_n, \mathcal{T}_{(X_{n-1},1)}, ..., \mathcal{T}_{(X_{n-1},m_{n-1})}) \\ X_n &= zip^{m_n}(X_0, \mathcal{T}_{(X_n,1)}, ..., \mathcal{T}_{(X_n,m_n)}) \end{aligned}$$

where $\mathcal{T}_{(X_i,n)} \in Terms(E)$ and $m_0, ..., m_n \in \mathbb{N}$. We introduce a shorthand notation for the terms in the right-hand sides of each equation:

$$V_i = \mathcal{T}_{(X_i,1)}, ..., \mathcal{T}_{(X_i,m_i)}, 0 \leq i \leq n$$

so that we can write the zip-cycle as

$$\begin{aligned} X_0 &= zip^{m_0}(X_1, V_0) \\ X_1 &= zip^{m_1}(X_2, V_1) \\ &... \\ X_{n-1} &= zip^{m_{n-1}}(X_n, V_{n-1}) \\ X_n &= zip^{m_n}(X_0, V_n) \end{aligned} \tag{1}$$

so that equation 2 (below) doesn't become too large and unreadable. Any rewriting in the terms $\mathcal{T}_{(X_i,n)}, 0 \leq n \leq m_i$ does not change the structure of the zip-cycle, since such rewriting does not change the above structure, i.e. using the rewrite rules on those terms can only result in a set of terms $\mathcal{T}'_{(X_i,n)}, 0 \leq n \leq m_i$ such that $\mathcal{T}_{(X_i,n)} \rightarrow^*_{R_E} \mathcal{T}'_{(X_i,n)}$. Using our shorthand notation this means that any rewriting applied to a vector, will yield a new vector with exactly the same number of elements: $V_i \rightarrow^*_{R_E} V'_i$. Again, this does not change the structure. Besides rewriting in those terms, we only have one other type of rewriting left: substitution of recursion variables by their right-hand side. Using the set of rewrite rules R_E, by repeated substitution of recursion variables by the right-hand sides of their defining equations, we can rewrite the cycle (from equation 1) to the following:

$$\begin{aligned} X_0 &= zip^{\phi}(X_0, V'_n, V'_{n-1}, ..., V'_0) \\ X_1 &= zip^{\phi}(X_1, V'_0, V'_n, V'_{n-1}, ..., V'_1) \\ &... \\ X_{n-1} &= zip^{\phi}(X_{n-1}, V'_{n-2}, V'_{n-3}, ...V'_0, V'_n, V'_{n-1}) \\ X_n &= zip^{\phi}(X_n, V'_{n-1}, V'_{n-2}, ...V'_0, V'_n) \end{aligned} \tag{2}$$

where $\phi = m_0 + m_1 + ... + m_{n-1} + m_n$. Note that rewriting to this form takes $n * (n-1)$ substitutions. □

Next, we prove that if a system of recursive equations is guarded, it is zip-cycle-free.

Theorem 7. *(Guardedness implies Zip-cycle-freeness) If a system of recursive equations E is guarded, it is zip-cycle-free.*

Proof. Suppose E is a system of recursive equations and E is guarded. Also, suppose that E is not zip-cycle free. If E is not zip-cycle-free, there is a zip-cycle consisting of a number n of recursion variables, that can be rewritten to n separate zip-cycles over each recursion variable, as described in Lemma 6.

Let X_i be such a recursion variable in a zip-cycle and let us rewrite its recursive equation to:

$$X_i = zip^{m_i}(X_i, \mathcal{S}_{(X_i,1)}, ..., \mathcal{S}_{(X_i,m_i)})$$

As we can see, the only rewriting that is possible is rewriting in the terms $\mathcal{S}_{(X_i,n)}$, which does not change the shape of the recursive equation. In order to write the recursive equation to a completely guarded form, a prefix operator in the first argument is needed so that the rewrite rule $zip(a : xs, ys) \rightarrow a : zip(ys, xs)$ can be applied m_i times on the outermost zip function applications. Since this will not be achieved by rewriting in the terms $\mathcal{S}_{(X_i,n)}$, the recursive equation X_i can not be rewritten to a guarded form. This is a contradiction with the assumption that E is guarded, which means that if a system of recursive equations is guarded, it must be zip-cycle-free. □

Next, we prove that if a system of recursive equations is not guarded, it is not zip-cycle-free.

Theorem 8. *(Zip-cycle free implies guarded) If a system of recursive equations E is not guarded, it is not zip-cycle-free. More specifically: If a recursion variable in E is guarded, it is not part of a zip-cycle.*

Proof. Suppose E is a system of recursive equations and E is not guarded. Also, suppose that E is zip-cycle-free.

Firstly, note that the structure of any recursive equation adheres to the following structure:

$$Y = zip^n(\mathcal{T}_{Rec}, \mathcal{T}_{(Y,1)}, ..., \mathcal{T}_{(Y,n)})$$

such that $n \in \mathbb{N}$ and $\mathcal{T}_{Rec} \not\equiv zip(\mathcal{T}_y, \mathcal{T}_z)$ for any $\mathcal{T}_y, \mathcal{T}_z \in Terms(A, E)$. Also, note that Y_0 is guarded if and only if $\mathcal{T}_{Rec}$ is guarded.

Let Y_0 be an arbitrary unguarded recursion variable in E. We analyze the structure of Y_0's defining recursive equation:

- $\mathcal{T}_{Rec} \equiv Y_0$. The recursive equation for Y_0 is is a zip-cycle, which is a contradiction to the fact that E is zip-cycle-free, so this can not be the case.
- $\mathcal{T}_{Rec} \equiv a : \mathcal{T}_s$. The recursive equation for Y_0 is guarded (no matter the value of n), which is a contradiction to Y_0's unguardedness, so this can not be the case.

- $\mathcal{T}_{Rec} \equiv Y_1$ and $Y_1 \not\equiv Y$ and Y_1 unguarded. Note that, if Y_1 is unguarded, it in turn must depend on a recursion variable. However, this recursion variable may not be Y_0 or Y_1, because this would constitute a zip-cycle. So, Y_1 must depend on yet another recursion variable, say Y_2. Since Y_2 must again depend on a different recursion variable, so as not to constitute a zip-cycle, this would imply that an infinite number of recursion variables exist such that Y_i depends on Y_{i+1}. However, this is a contradiction to the fact that a system of recursive equations consists of a finite number of recursive equations (as specified in Section 1).

The cases of the above conclude that Y_0 can not exist if no zip-cycle exists, i.e. Y_0 is part of a zip-cycle. Since Y_0 is an arbitrary unguarded recursion variable, this means that no unguarded recursion variables can exist, i.e. that the system is guarded. This is a contradiction to the fact that the system is unguarded, which means that if a system of recursive equations is unguarded, it must contain a zip-cycle. □

4 Observability

`observability` acts as an intermediary between `guardedness` and `well-definedness`; in Sections 4.1 and 4.2 these three properties will be shown to be equivalent.

Let $obs : Term \times \mathbb{N} \to Term \times \mathbb{N} \cup \mathbf{A}$ be a total function defined as follows:

$$\begin{aligned} obs(a : \mathcal{T}_x, 0) &= a \\ obs(a : \mathcal{T}_x, i+1) &= (\mathcal{T}_x, i) \\ obs(zip(\mathcal{T}_x, \mathcal{T}_y), 2i) &= (\mathcal{T}_x, i) \\ obs(zip(\mathcal{T}_x, \mathcal{T}_y), 2i+1) &= (\mathcal{T}_y, i) \\ obs(X, i) &= (\mathcal{T}_X, i) \text{ for all } X = \mathcal{T}_X \in E \end{aligned}$$

Note that this function is different for every system of recursive equations, because it uses the recursive equations of the system of recursive equations. If the n-fold application of obs to $(\mathcal{T}_s, j)$ equals a, then a is the j-th element of the stream described by $\mathcal{T}_s$.

Definition 9. *(Observability) Let E be a system of recursive equations over a data set $\mathbf{A}$. System of recursive equations E is observable if and only if:*

$$(\forall \mathcal{T}_s \in Terms(E), i \in \mathbb{N} : (\exists n \in \mathbb{N}, a \in \mathbf{A} : n > 0 : obs^n(\mathcal{T}_s, i) = a))$$

We introduce the following predicate over terms for a more concise notation:

$$\texttt{OBS}(\mathcal{T}_s) \equiv (\forall i \in \mathbb{N} : (\exists n \in \mathbb{N}, a \in \mathbf{A} : n > 0 : obs^n(\mathcal{T}_s, i) = a))$$

Thus, a system of recursive equations E is observable if and only if:

$$(\forall \mathcal{T}_s \in Terms(E) :: \texttt{OBS}(\mathcal{T}_s))$$

4.1 Observability equivalent to Guardedness

In this section, we first show that `guardedness` implies `observability` and later we show that `observability` implies `guardedness`.

4.1.1 Guardedness implies Observability

In order to prove that `guardedness` implies `observability`, we first show that if and only if a system of recursive equations is guarded, every term is guarded. Also, we show the relationship between the rewrite rules and the observation function.

Theorem 10. *(Guardedness of Terms) If and only if a system of recursive equations E is guarded, every term in $Terms(E)$ is guarded.*

$$\texttt{GUARDED}(E) \equiv (\forall \mathcal{T}_s \in Terms(E) : (\exists a \in \mathbf{A}, \mathcal{T}_u \in Terms(E) :$$

$$\mathcal{T}_s \text{ can be rewritten using the set of rewrite rules } R_E \text{ to } a : \mathcal{T}_u))$$

Proof.

- The first implication:

$$\texttt{GUARDED}(E) \Rightarrow (\forall \mathcal{T}_s \in E : (\exists a \in \mathbf{A}, \mathcal{T}_u \in Terms(E) :$$

$$\mathcal{T}_s \text{ can be rewritten using the set of rewrite rules } R_E \text{ to } a : \mathcal{T}_u))$$

  Let $\mathcal{T}_s$ be an arbitrary term. Via structural induction on $\mathcal{T}_s$ we derive the completely guarded form of $\mathcal{T}_s$.

  - Suppose $\mathcal{T}_s \equiv X$. Since X is guarded, we substitute $\mathcal{T}_s$ by the completely guarded form of X to obtain the completely guarded form of $\mathcal{T}_s$.
  - Suppose $\mathcal{T}_s \equiv a : \mathcal{T}_t$ for some $a \in \mathbf{A}, \mathcal{T}_t \in Terms(E)$. Trivially, $\mathcal{T}_s$ is in its completely guarded form.
  - Suppose $\mathcal{T}_s \equiv zip(\mathcal{T}_u, \mathcal{T}_v)$. Via structural induction, $\mathcal{T}_u$ is guarded. We derive the completely guarded form of $\mathcal{T}_s$ by substituting $\mathcal{T}_u$ by its completely guarded form and applying the rewrite rule for *zip*.

- The second implication:

$$\texttt{GUARDED}(E) \Leftarrow (\forall \mathcal{T}_s \in Terms(E) : (\exists a \in \mathbf{A}, \mathcal{T}_u \in Terms(E) :$$

$$\mathcal{T}_s \text{ can be rewritten using the set of rewrite rules } R_E \text{ to } a : \mathcal{T}_u))$$

  This is trivially true, since if all terms can be rewritten to a completely guarded form, then surely all terms in the right-hand sides of the defining equations can be rewritten to a completely guarded form.

□

Theorem 11. *(Rewriting and the Observation Function) If every term can be rewritten to a completely guarded form, the observation function applied to an arbitrary term $\mathcal{T}_s$ and index i will at some point (after an n-fold application for some $n \in \mathbb{N}$) encounter a tuple consisting of completely guarded term in the first argument, and the same index i in the second argument:*

$(\forall \mathcal{T}_s \in Terms(E) : \mathcal{T}_s \textit{ can be rewritten to completely guarded form})$
$\Rightarrow (\forall i \in \mathbb{N}, \mathcal{T}_s \in Terms(E) :: (\exists n \in \mathbb{N}, a \in \mathbf{A}, \mathcal{T}_t \in Terms(E) :: obs^n(\mathcal{T}_s, i) = (a : \mathcal{T}_t, i)))$

Proof. We choose to prove that if the observation function applied to an arbitrary term does not encounter a completely guarded term at the same index, then not every term can be rewritten to a completely guarded form:

$\neg(\forall i \in \mathbb{N}, \mathcal{T}_s \in Terms(E) :: (\exists n \in \mathbb{N}, a \in \mathbf{A}, \mathcal{T}_t \in Terms(E) :: obs^n(\mathcal{T}_s, i) = (a : \mathcal{T}_t, i)))$
$\Rightarrow \neg(\forall \mathcal{T}_s \in Terms(E) : \mathcal{T}_s \text{ can be rewritten to completely guarded form})$

Suppose that for some term $\mathcal{T}_s$ and some index i, the observation function does not encounter a completely guarded term. By inspection of *obs* we see that the second argument of the returned tuple is always a natural number; if the second argument is greater than zero,

it either does not decrease or decreases by at most one and if the second argument is equal to zero, it remains equal to zero. Also, the second argument is never increased. This implies that, after a certain number of *obs* applications, the second argument of the returned tuple stabilizes to some j: it remains the same, no matter how many *obs* applications follow. By inspection of *obs*, we see that either:

- $j = 0$ and for every application of *obs* the observation function returns $(zip(\mathcal{T}_x, \mathcal{T}_y), 0)$ for some $\mathcal{T}_x, \mathcal{T}_y \in Terms(E)$ or $(X, 0)$ for some $X \in Vars(E)$, or
- $j > 0$ and for every application of *obs* the observation function returns some (X, j) for some $X \in Vars(E)$.

In both cases, we observe that this must be a zip-cycle, the shape of which has been described in Definition 3. From Section 3.1 we know that if the system of recursive equations is not zip-cycle-free, then is not guarded. Thus, there are recursion variables that can not be rewritten to a completely guarded form. □

Theorem 12. *(Guardedness implies Observability) Let E be a system of recursive equations over data set $\mathbf{A}$. If E is guarded, it is observable:*

$$\texttt{GUARDED}(E) \Rightarrow (\forall \mathcal{T}_s \in Terms(E) : \texttt{OBS}(\mathcal{T}_s))$$

Proof. Let $\mathcal{T}_s$ be an arbitrary term in $Terms(E)$. By Theorem 10 we know that $\mathcal{T}_s$ is guarded. Via induction on i:

($i = 0$) Since $\mathcal{T}_s$ is guarded, then by Theorem 11 the observation function will encounter a completely guarded term at the same index:

$$obs^n(\mathcal{T}_s, 0) = ... = (b : \mathcal{T}_t, 0)$$

Applying *obs* yields

$$obs(b : \mathcal{T}_t, 0) = b$$

which concludes this case of the proof.

($i > 0$) Induction hypothesis:

$$(\forall j \in \mathbb{N} : j < i : (\exists n \in \mathbb{N}, a \in \mathbf{A} : n > 0 : obs^n(\mathcal{T}_s, j) = a))$$

Suppose $i = i' + 1$. Since $\mathcal{T}_s$ is guarded, then by Theorem 11 the observation function will encounter a completely guarded term at the same index:

$$obs^n(\mathcal{T}_s, i' + 1) = ... = (b : \mathcal{T}_t, i' + 1)$$

Applying *obs* yields

$$obs(b : \mathcal{T}_t, i' + 1) = (\mathcal{T}_t, i')$$

Since $i' < i$, by induction hypothesis there, exists a $l \in \mathbb{N}, a \in \mathbf{A}$ such that $obs^l(\mathcal{T}_t, j) = a$, which concludes this case of the proof.

□

4.1.2 Observability implies Guardedness

Next, we wish to prove that observability implies guardedness: if all terms in a system of recursive equations are observable, then the system is guarded.

Theorem 13. *(Observability implies Guardedness) If a system of recursive equations E is observable, it is guarded:*

$$(\forall \mathcal{T}_t \in Terms(E) : \mathit{OBS}(\mathcal{T}_t)) \Rightarrow \mathit{GUARDED}(E)$$

Proof. We show that if a system of recursive equations is not guarded, then one or more of the terms that can be created using that system, can not be observed. We choose this approach because the proof is more elegant and less involved due to the fact that unguarded specifications have an easy to describe form.

To prove: If a system of recursive equations E is not guarded, it is not observable:

$$\neg\mathtt{GUARDED}(E) \Rightarrow (\exists \mathcal{T}_t \in Terms(E) : \neg\mathtt{OBS}(\mathcal{T}_t))$$

Suppose E is an unguarded system of recursive equations. As E is unguarded, by Section 3.1, we know that it is not zip-cycle-free. This implies that there is a set of recursive equations over a set of recursion variables $\{X_0, ..., X_n\} \subseteq Vars(E)$ such that the defining equations of this set constitute a zip-cycle. As shown in Lemma 6, using the set of rewrite rules R_E, by repeated substitution of recursion variables by the right-hand sides of their defining equations we can rewrite this cycle to the following:

$$\begin{aligned}
X_0 &= zip^{\phi}(X_0, \mathcal{S}_{(X_0,1)}, \mathcal{S}_{(X_0,2)}, ..., \mathcal{S}_{(X_0,\phi)}) \\
X_1 &= zip^{\phi}(X_1, \mathcal{S}_{(X_1,1)}, \mathcal{S}_{(X_1,2)}, ..., \mathcal{S}_{(X_1,\phi)}) \\
&... \\
X_{n-1} &= zip^{\phi}(X_{n-1}, \mathcal{S}_{(X_{n-1},1)}, \mathcal{S}_{(X_{n-1},2)}, ..., \mathcal{S}_{(X_{n-1},\phi)}) \\
X_n &= zip^{\phi}(X_n, \mathcal{S}_{(X_n,1)}, \mathcal{S}_{(X_n,2)}, ..., \mathcal{S}_{(X_n,\phi)})
\end{aligned}$$

where $\mathcal{S}_{(X_i,n)}$ are terms in $Terms(E)$. We show that we can not observe X_i, by induction on $\mathbb{N}$:

- $(n = 0)$: $X_i = X_i$. By inspection of *obs*, we see that $obs(X_i, i) = (X_i, i)$, and thus that $(\forall m \in \mathbb{N}, a \in \mathbf{A} : obs^m(X_i, i) \neq a)$, i.e. that X_i is not observable.
- $(n \geq 1)$: $X_i = zip^n(X_i, \mathcal{S}_{(X_i,1)}, \mathcal{S}_{(X_i,2)}, ..., \mathcal{S}_{(X_i,n)})$. By inspection of *obs*, we see that

$$\begin{aligned}
obs^{n+1}(X_i, i) &= obs^n(zip^n(X_i, \mathcal{S}_{(X_i,1)}, \mathcal{S}_{(X_i,2)}, ..., \mathcal{S}_{(X_i,n)}), i) \\
&= obs^{n-1}(zip^{n-1}(X_i, \mathcal{S}_{(X_i,1)}, \mathcal{S}_{(X_i,2)}, ..., \mathcal{S}_{(X_i,n-1)}), i) \\
&= ... \\
&= (X_i, i)
\end{aligned}$$

  and thus that $(\forall m \in \mathbb{N}, a \in \mathbf{A} : obs^m(X_i, 0) \neq a)$, i.e. that X_i is not observable.

□

4.2 Observability equivalent to Well-definedness

Now, we show that observability is equivalent to well-definedness. First, we show that observability implies well-definedness.

Theorem 14. *(Observability implies Well-definedness) If a system of recursive equations is observable, then it is well-defined.*

Proof. By construction: If for every j, the n-fold application of *obs* to any recursion variable equals some element $a \in \mathbf{A}$, then we know that every element is observable and thus that the system of recursive equations is well-defined. □

In order to show that if a system of recursive equations is unobservable it has multiple solutions, we need some additional properties. Theorem 15 shows that all terms in $Terms(E)$ are observable if and only if all recursion variables in $Vars(E)$ are observable. This allows us to reason about both observable terms and observable recursion variables, since an unobservable term can not occur if all recursion variables are observable and vice versa. Theorem 16 shows that if a system of recursive equations is unobservable, there is a recursion variable of which the first element is unobservable. In Theorem 17, we use this property to derive that this recursion variable has multiple solutions. In Theorem 18 we show that every recursion variable has at least one solution. Lastly, in Theorem 19, we combine these results to show that if a system is well-defined it is observable.

Theorem 15. *(Observability of terms and recursion variables) Observability of all terms in $Terms(E)$ is equivalent to observability of all recursion variables in $Vars(E)$. Note that recursion variables in themselves are terms as well.*

$$(\forall \mathcal{T}_t \in Terms(E) :: \mathit{OBS}(\mathcal{T}_t)) \\ \Leftrightarrow (\forall \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : \mathit{OBS}(\mathcal{T}_t))$$

Proof. We prove each implication separately.

- The first implication

  $$(\forall \mathcal{T}_t \in Terms(E) :: \mathtt{OBS}(\mathcal{T}_t)) \\ \Rightarrow (\forall \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : \mathtt{OBS}(\mathcal{T}_t))$$

  follows from

  $$\{\mathcal{T}_t \mid \mathcal{T}_t \in Terms(E) \wedge \mathcal{T}_t \in Vars(E) \wedge \mathtt{OBS}(\mathcal{T}_t)\} \\ \subset \\ \{\mathcal{T}_t \mid \mathcal{T}_t \in Terms(E) \wedge \mathtt{OBS}(\mathcal{T}_t)\}$$

- The second implication
  $(\forall \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : \mathtt{OBS}(\mathcal{T}_t)) \\ \Rightarrow (\forall \mathcal{T}_t \in Terms(E) :: \mathtt{OBS}(\mathcal{T}_t))$

  Given that $(\forall \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : \mathtt{OBS}(\mathcal{T}_t))$, suppose $\mathcal{T}_s$ is a term such that $\neg\mathtt{OBS}(\mathcal{T}_s)$. Via the structure of $\mathcal{T}_s$ we observe:

- $\mathcal{T}_s \equiv Y$ for some $Y \in Vars(E)$. Then, a contradiction exists, because $\mathcal{T}_s$ would be observable since Y is observable.
- $\mathcal{T}_s \equiv a : Y$ for some $a \in \mathbf{A}, Y \in Vars(E)$. Then, for $i \in \mathbb{N}$, if $i = 0$ then $obs(\mathcal{T}_s, i) = a$ and if $i > 0$ then $obs^2(\mathcal{T}_s, i) = obs(a : Y, i) = (Y, i-1)$. Then, $\mathcal{T}_s$ is be observable since Y is observable. In both cases, $\mathcal{T}_s$ is observable, which is a contradiction.
- $\mathcal{T}_s \equiv zip(Y, Z)$ for some $Y, Z \in Vars(E)$. Then, for $i, j \in \mathbb{N}$, if $i = 2j$ then $obs^2(\mathcal{T}_s, i) = obs(zip(Y, Z), i) = (Y, j)$ and if $i = 2j+1$ then $obs^2(\mathcal{T}_s, i) = obs(zip(Y, Z), i) = obs(Z, j)$. Then, $\mathcal{T}_s$ would be observable since Y, Z are observable, which is a contradiction.
- $\mathcal{T}_s \equiv a : \mathcal{T}_t$ for some $a \in \mathbf{A}, \mathcal{T}_t \in Terms(E)$. Via structural induction on the above, we know that $\mathcal{T}_t$ is observable. Then, for $i \in \mathbb{N}$, if $i = 0$ then $obs(\mathcal{T}_s, i) = a$ and if $i > 0$ then $obs^2(\mathcal{T}_s, i) = obs(\mathcal{T}_t, i-1)$. Then, $\mathcal{T}_s$ is be observable since $\mathcal{T}_t$ is observable. In both cases, $\mathcal{T}_s$ is observable, which is a contradiction.
- $\mathcal{T}_s \equiv zip(\mathcal{T}_Y, \mathcal{T}_Z)$ for some $\mathcal{T}_Y, \mathcal{T}_Z \in Terms(E)$. Via structural induction on the above cases, we know that $\mathcal{T}_Y, \mathcal{T}_Z$ are observable. Then, for $i, j \in \mathbb{N}$, if $i = 2j$ then $obs^2(\mathcal{T}_s, i) = obs(zip(\mathcal{T}_Y, \mathcal{T}_Z), i) = (\mathcal{T}_Y, j)$ and if $i = 2j + 1$ then $obs^2(\mathcal{T}_s, i) = obs(zip(\mathcal{T}_Y, \mathcal{T}_Z), i) = obs(\mathcal{T}_Z, j)$. Then, $\mathcal{T}_s$ would be observable since $\mathcal{T}_Y, \mathcal{T}_Z$ are observable, which is a contradiction.

We conclude that if $(\forall \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : \texttt{OBS}(\mathcal{T}_t))$, then any term in $Terms(E)$ is observable.

□

Theorem 16. *(First element unobservable) If a system of recursive equations E is unobservable, there is a recursion variable X in E of which the first element is unobservable:*

$$(\exists \mathcal{T}_s \in Terms(E) :: \neg \texttt{OBS}(\mathcal{T}_s)) \Rightarrow \\ (\exists \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : (\forall n \in \mathbb{N}, a \in \mathbf{A} : n > 0 : obs^n(\mathcal{T}_t, 0) \neq a))$$

Proof. Using proofs from Section 4.1, we know that if some recursion variable is not observable, then system E is not guarded. Note that the proofs from this Section may be used because they do not depend on the equivalence between observability and well-definedness, so no circle proof arises. Given that E is not guarded, we wish to show that a recursion variable exists of which the first element is unobservable. Suppose the opposite: for every recursion variable, the first element is observable.

$$(\forall \mathcal{T}_t \in Terms(E) : \mathcal{T}_t \in Vars(E) : (\exists n \in \mathbb{N}, a \in \mathbf{A} : n > 0 : obs^n(\mathcal{T}_t, 0) = a))$$

The goal is to show that every term is observable or, using the equivalence between guardedness and observability, that every term is guarded, to complete the proof.

Suppose $\mathcal{T}_s$ is an arbitrary term in $Terms(E)$. We wish to derive that this term is guarded.

Via induction on n, we derive for each term its completely guarded form to show that it is guarded:

- $n = 1$. Via inspection of *obs*, we see that $\mathcal{T}_s$ is completely guarded.
- $n > 1$. Induction Hypothesis: For all terms whose observation function requires fewer applications, the desired property holds.

  $$(\forall \mathcal{T}_t \in Terms(E) : (\exists m \in \mathbb{N}, a \in \mathbf{A} : n > m > 0 : obs^m(\mathcal{T}_t, 0) = a) : \mathcal{T}_t \text{ is guarded })$$

  Via the structure of $\mathcal{T}_s$:

  - $\mathcal{T}_s \equiv X$ for some $X \in Vars(E)$. By inspection of *obs*, we have $obs^n(\mathcal{T}_s, 0) = obs^{n-1}(X, 0) = a$. Via the induction hypothesis, we know that X is guarded. We derive the completely guarded form of X and substitute this for $\mathcal{T}_s$ to acquire the completely guarded form of $\mathcal{T}_s$.
  - $\mathcal{T}_s \equiv a : \mathcal{T}_t$ for some $a \in \mathbf{A}, \mathcal{T}_t \in Terms(E)$. By inspection of *obs*, we know that $n = 2$. Trivially, $\mathcal{T}_s$ is completely guarded.
  - $\mathcal{T}_s \equiv zip(\mathcal{T}_t, \mathcal{T}_u)$ for some $\mathcal{T}_t, \mathcal{T}_u \in Terms(E)$. By inspection of *obs*, we have $obs^n(\mathcal{T}_s, 0) = obs^{n-1}(zip(\mathcal{T}_t, \mathcal{T}_u), 0) = obs^{n-2}(\mathcal{T}_t, 0) = a$. Via the induction hypothesis, we know that $\mathcal{T}_t$ is guarded. We derive the completely guarded form for $\mathcal{T}_t$ and substitute this for $\mathcal{T}_t$. Next, apply the rewrite rule for *zip* to obtain a completely guarded term for $\mathcal{T}_s$.

Since $\mathcal{T}_s$ is guarded, it is observable. Since $\mathcal{T}_s$ is an arbitrary term, every term in E is guarded, which means E is guarded, which is a contradiction. Therefore, not every recursion variable's first element is observable. □

Theorem 17. *(Multiple Solutions to defining equation) If the first element of a recursion variable X_i in the system of recursive equations E is undefined, X_i has multiple solutions.*

Proof. Using proofs from Section 4.1, we know that if X_i is not observable, then system E is not guarded. Note that the proofs from this section may be used because they do not depend on the equivalence between observability and well-definedness, so no circle proof arises. We also know that X_i is unguarded, for if X_i were guarded then the first element of X_i would be observable. By Theorem 8 we know that X_i is part of a zip-cycle. This implies that there is a set of recursive equations over a set of recursion variables $\{X_0, ..., X_n\} \subseteq Vars(E)$ such that the defining equations of this set constitute a zip-cycle and X_i is part of this set $(0 \leq i \leq n)$. As shown in Lemma 6, using the set of rewrite rules R_E, by repeated substitution of recursion variables by the right-hand sides of their defining equations we can rewrite this cycle to the following:

$$\begin{aligned}
X_0 &= zip^{\phi}(X_0, \mathcal{S}_{(X_0,1)}, \mathcal{S}_{(X_0,2)}, ..., \mathcal{S}_{(X_0,\phi)}) \\
X_1 &= zip^{\phi}(X_1, \mathcal{S}_{(X_1,1)}, \mathcal{S}_{(X_1,2)}, ..., \mathcal{S}_{(X_1,\phi)}) \\
&... \\
X_{n-1} &= zip^{\phi}(X_{n-1}, \mathcal{S}_{(X_{n-1},1)}, \mathcal{S}_{(X_{n-1},2)}, ..., \mathcal{S}_{(X_{n-1},\phi)}) \\
X_n &= zip^{\phi}(X_n, \mathcal{S}_{(X_n,1)}, \mathcal{S}_{(X_n,2)}, ..., \mathcal{S}_{(X_n,\phi)})
\end{aligned}$$

where $\mathcal{S}_{(X_i,n)}$ are terms in $Terms(E)$. Below, we show that for any natural number $n \in \mathbb{N}$, any recursive equation $X = zip^n(X, \mathcal{S}_1, ..., \mathcal{S}_n)$, X has at least two solutions.

By induction on n:

- If $n = 0$, then $X = X$. Since $\mathbf{A}$ contains at least two elements, there are at least two different streams, for example the constant streams. Each of these streams satisfies the equation.
- If $n \geq 1$: $X = zip^n(X, \mathcal{S}_{X,1}, ..., \mathcal{S}_{X,n})$.
  - Let $a, b \in \mathbf{A}, a \neq b$.
  - Assume the first element of X is a, i.e. the solution to the system of recursive equations ϕ_1 is such that $\phi_1(X)(0) = a$. The rest of X is a solution for the equation $Z = zip^n(\mathcal{S}_{(X,n)}, ..., \mathcal{S}_{(X,1)}, Z)$, where Z is a recursion variable that is not used in E. Thus, $a : Z$ is a solution to X if we can show that $a : Z$ satisfies the defining equation of X:

$$\begin{aligned} a : Z &= a : zip^n(\mathcal{S}_{n-1}, ..., \mathcal{S}_1, Z) \\ &= zip^n(a : Z, \mathcal{S}_1, ..., \mathcal{S}_{n-1}) \end{aligned}$$

  - Assume the first element of X is b, i.e. the solution to the system of recursive equations ϕ_2 is such that $\phi_2(X)(0) = b$. The rest of X, is again Z, like above. By the same reasoning, the solution $b : Z$ is a solution to X.
  - We know, by the above, that either the first element of Z is defined, or Z is unguarded and any element in $\mathbf{A}$ can be in place of the first element. We may repeat this (inspect the first element or choose an arbitrary value for the first element, then continue to inspect the rest of the stream) indefinitely, and therefore we know that Z has one or more solutions.
  - Both $a : Z$ and $b : Z$ satisfy $X = zip^n(X, \mathcal{S}_{(X,1)}, ..., \mathcal{S}_{(X,n)})$, so both are a solution to X. Given that $a \neq b$ it has been shown that $a : Z \neq b : Z$ and thus that $\phi_1(X) \neq \phi_2(X)$, i.e. X has multiple solutions.

□

Theorem 18. *(Recursion Variables At Least One Solution) Every recursion variable Z in a system of recursive equations E, has at least one solution.*

Proof. As shown in Theorem 17, either the recursion variable's defining equation is guarded and the first element of a recursion variable Z is defined, or it is unguarded and any element in $\mathbf{A}$ can be in place of the first element. We may repeat this (inspect the first element or choose an arbitrary value for the first element, then continue to inspect the rest of the stream) indefinitely, and therefore we know that Z has one or more solutions. □

Theorem 19. *(Well-definedness implies Observability) If a system of recursive equations E is well-defined, it is observable.*

Proof. We choose to prove the contrapositive: If a system of recursive equations E is unobservable, it has multiple solutions (i.e. it is not well-defined).

Using proofs from Section 4.1, we know that if E is not observable, then E is not guarded. Note that the proofs from this section may be used because they do not depend on the

equivalence between observability and well-definedness, so no circle proof arises. Also, using Theorem 16 we know that if E is not guarded, there is a recursion variable X of which the first element is unobservable. Using Theorem 17, we know that this recursion variable has multiple solutions. Furthermore, we know by Theorem 18 that every recursion variable has at least one solution. And so, E has multiple solutions:

$$(\exists \phi : Vars(E) \to \mathbf{A}^\omega :: (\forall X = \mathcal{T}_X \in E :: \phi(X) \equiv \phi(\mathcal{T}_X)))$$

□

5 Conclusion

We have studied systems of recursive equations over the action prefix operator (:) and function *zip*, that strictly interleaves streams. In order to decide `well-definedness` of a system of recursive equations in this setting, we have introduced three properties of systems of recursive equations: `zip-cycle-freeness`, `guardedness` and `observability`. We have shown that all three are equivalent to well-definedness by showing that `zip-cycle-freeness` is equivalent to `guardedness`, which is equivalent to `observability`, which in turn is equivalent to `well-definedness`:

$$E \text{ is } \texttt{zip-cycle-free} \equiv \texttt{GUARDED}(E) \equiv (\forall \mathcal{T}_s \in Terms(E) :: \texttt{OBS}(\mathcal{T}_s)) \equiv E \text{ is } \texttt{well-defined}$$

thus proving that `zip-cycle-freeness` is equivalent to `well-definedness`. For `zip-cycle-freeness` we have a decision procedure that is at worst linear in the size of the system of recursive equations and so, due to the equality between zip-cycle-freeness and well-definedness, we have a very efficient decision procedure for well-definedness.

References

[1] [Baeten, Basten, Reniers, 2007] J.C.M. Baeten, T. Basten and M.A. Reniers, Process alge- bra (equational theories of communicating processes), Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, to appear end 2006 or beginning 2007.

[2] [Rutten, 2000] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series, Report SEN-R0023, CWI, 2000.

[3] [Endrullis, Grabmayer, Hendriks, 2008] J. Endrullis, C. Grabmayer, and D. Hendriks. Data-oblivious stream productivity. In Proceedings of the 11th International Conference on Logic for Programming, Artifcial Intelligence, and Reasoning (LPAR'08), volume 5330 of Lecture Notes in Computer Science, pages 79-96. Springer, 2008.

[4] [Groote, Reniers, 2008] Jan Friso Groote & Michel Reniers: Modelling and Analysis of Communicating Systems, Department of Computer Science, Eindhoven University of Technology, Eindhoven.