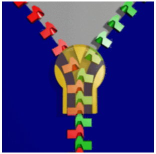
Automatic Sequences and Zip-Specifications



Clemens Grabmayer[1], Jörg Endrullis[2],
Dimitri Hendriks[2], Jan Willem Klop[2] and Lawrence S. Moss[3]

[1] Universiteit Utrecht
[2] Vrije Universiteit Amsterdam,
[3] Indiana University

LICS – Logic in Computer Science
Dubrovnik, Croatia, June 26, 2012

zipping

'Zipping' streams

$$\sigma = \sigma(0) : \sigma(1) : \sigma(2) : \ldots$$
$$\tau = \tau(0) : \tau(1) : \tau(2) : \ldots$$

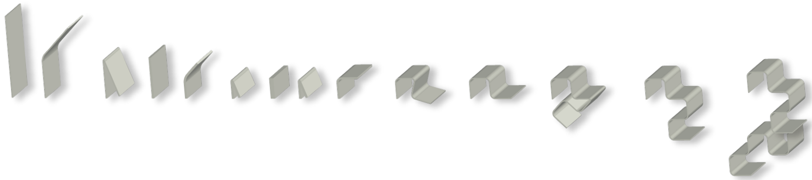
results in:

$$\mathbf{zip}(\sigma, \tau) = \sigma(0) : \tau(0) : \sigma(1) : \tau(1) : \sigma(2) : \tau(2) : \ldots$$

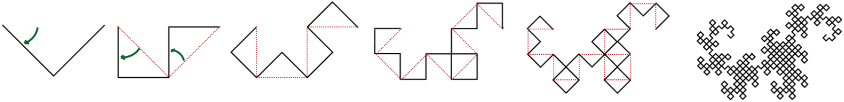
Defining equation:

$$\mathbf{zip}(x : \sigma, \tau) = x : \mathbf{zip}(\tau, \sigma)$$

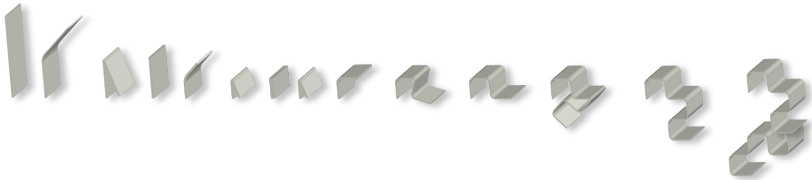
paperfolding: **zip**-specification



$$
\begin{aligned}
\text{Peaks} &= \wedge : \text{Peaks} \\
\text{Valleys} &= \vee : \text{Valleys} \\
\text{Tyrol} &= \textbf{zip}(\text{Peaks}, \text{Valleys}) \\
\text{Folds} &= \textbf{zip}(\text{Tyrol}, \text{Folds})
\end{aligned}
$$



paperfolding: **zip**-specification

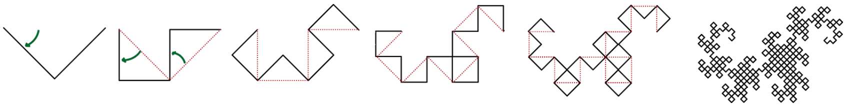


$$\text{Peaks} = \wedge : \text{Peaks} = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots$$

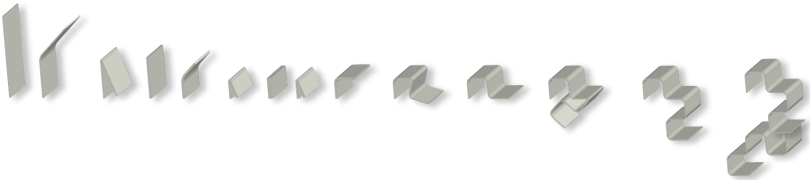
$$\text{Valleys} = \vee : \text{Valleys} = \vee : \vee : \vee : \vee : \vee : \ldots$$

$$\text{Tyrol} = \textbf{zip}(\text{Peaks}, \text{Valleys})$$

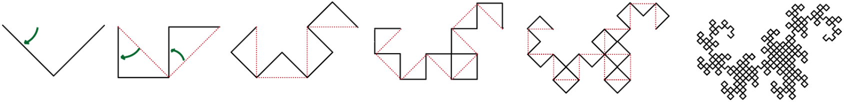
$$\text{Folds} = \textbf{zip}(\text{Tyrol}, \text{Folds})$$



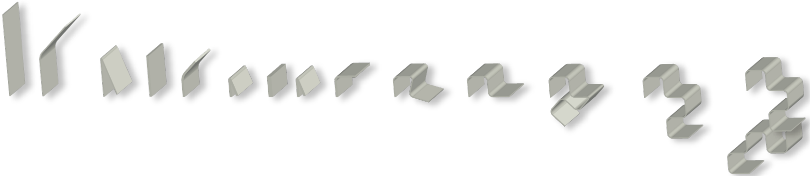
paperfolding: zip-specification



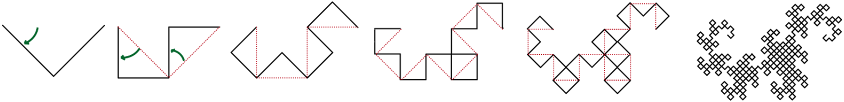
$$
\begin{aligned}
\text{Peaks} &= \wedge : \text{Peaks} && = \wedge : \wedge : \wedge : \wedge : \wedge : \dots \\
\text{Valleys} &= \vee : \text{Valleys} && = \vee : \vee : \vee : \vee : \vee : \dots \\
\text{Tyrol} &= \mathbf{zip}(\text{Peaks}, \text{Valleys}) && = \wedge : \vee : \wedge : \vee : \wedge : \vee : \dots \\
\text{Folds} &= \mathbf{zip}(\text{Tyrol}, \text{Folds})
\end{aligned}
$$



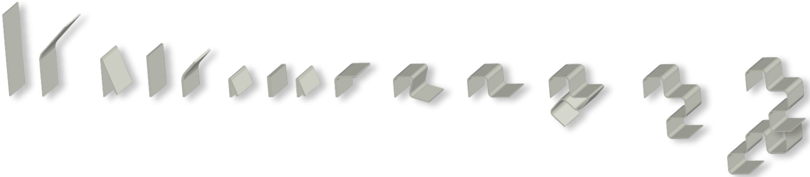
paperfolding: **zip**-specification



$$
\begin{aligned}
\text{Peaks} &= \wedge : \text{Peaks} && = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots \\
\text{Valleys} &= \vee : \text{Valleys} && = \vee : \vee : \vee : \vee : \vee : \ldots \\
\text{Tyrol} &= \mathbf{zip}(\text{Peaks}, \text{Valleys}) && = \wedge : \vee : \wedge : \vee : \wedge : \vee : \ldots \\
\text{Folds} &= \mathbf{zip}(\text{Tyrol}, \text{Folds}) && = \wedge : \quad : \vee : \quad : \wedge : \quad : \vee : \quad : \wedge : \quad : \vee : \ldots
\end{aligned}
$$



paperfolding: **zip**-specification

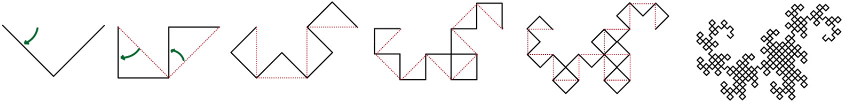


$$\text{Peaks} = \wedge : \text{Peaks} \qquad = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots$$

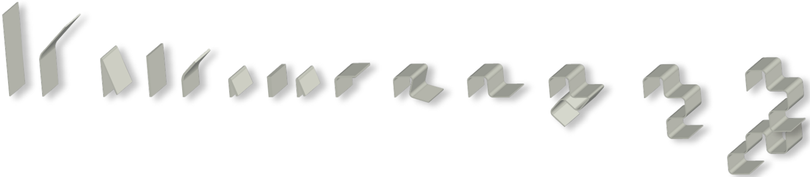
$$\text{Valleys} = \vee : \text{Valleys} \qquad = \vee : \vee : \vee : \vee : \vee : \ldots$$

$$\text{Tyrol} = \textbf{zip}(\text{Peaks}, \text{Valleys}) \qquad = \wedge : \vee : \wedge : \vee : \wedge : \vee : \ldots$$

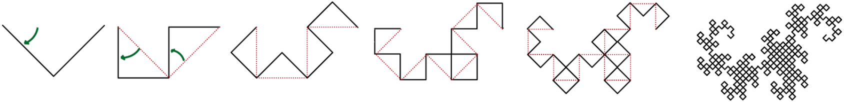
$$\text{Folds} = \textbf{zip}(\text{Tyrol}, \text{Folds}) \qquad = \wedge : \wedge : \vee : \quad : \wedge : \quad : \vee : \quad : \wedge : \quad : \vee : \ldots$$



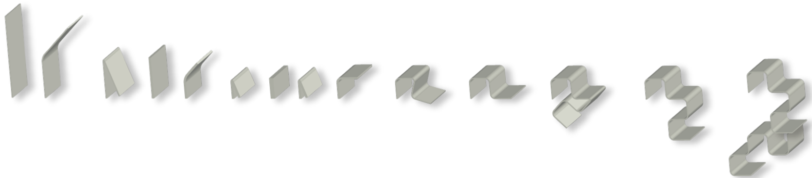
paperfolding: **zip**-specification



$$
\begin{aligned}
\text{Peaks} &= \wedge : \text{Peaks} && = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots \\
\text{Valleys} &= \vee : \text{Valleys} && = \vee : \vee : \vee : \vee : \vee : \ldots \\
\text{Tyrol} &= \textbf{zip}(\text{Peaks}, \text{Valleys}) && = \wedge : \vee : \wedge : \vee : \wedge : \vee : \ldots \\
\text{Folds} &= \textbf{zip}(\text{Tyrol}, \text{Folds}) && = \wedge : \wedge : \vee : \wedge : \wedge : \quad : \vee : \quad : \wedge : \quad : \vee : \ldots
\end{aligned}
$$



paperfolding: **zip**-specification

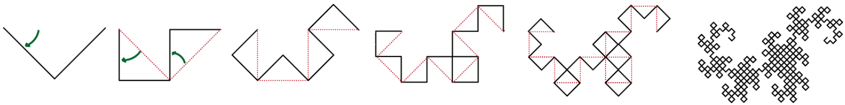


$$\text{Peaks} = \wedge : \text{Peaks} \quad = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots$$

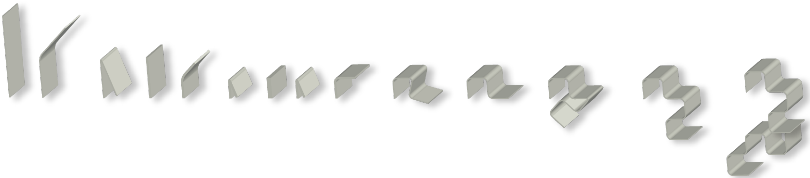
$$\text{Valleys} = \vee : \text{Valleys} \quad = \vee : \vee : \vee : \vee : \vee : \ldots$$

$$\text{Tyrol} = \textbf{zip}(\text{Peaks}, \text{Valleys}) \quad = \wedge : \vee : \wedge : \vee : \wedge : \vee : \ldots$$

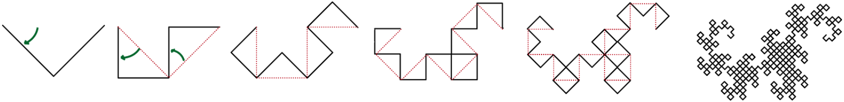
$$\text{Folds} = \textbf{zip}(\text{Tyrol}, \text{Folds}) \quad = \wedge : \wedge : \vee : \wedge : \wedge : \vee : \vee : \quad : \wedge : \quad : \vee : \ldots$$



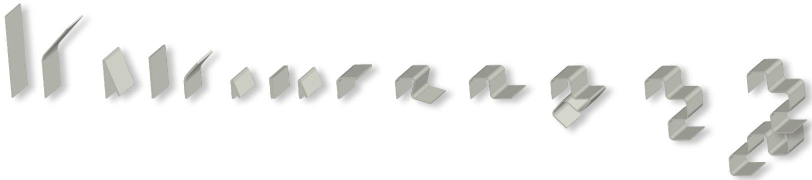
paperfolding: **zip**-specification



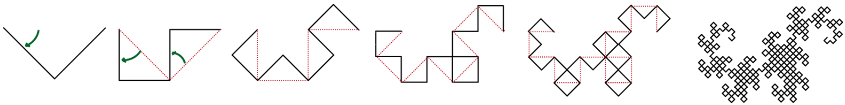
$$
\begin{aligned}
\text{Peaks} &= \wedge : \text{Peaks} && = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots \\
\text{Valleys} &= \vee : \text{Valleys} && = \vee : \vee : \vee : \vee : \vee : \ldots \\
\text{Tyrol} &= \textbf{zip}(\text{Peaks}, \text{Valleys}) && = \wedge : \vee : \wedge : \vee : \wedge : \vee : \ldots \\
\text{Folds} &= \textbf{zip}(\text{Tyrol}, \text{Folds}) && = \wedge : \wedge : \vee : \wedge : \wedge : \vee : \vee : \wedge : \wedge : \quad : \vee : \ldots
\end{aligned}
$$



paperfolding: **zip**-specification



$$
\begin{aligned}
\text{Peaks} &= \wedge : \text{Peaks} && = \wedge : \wedge : \wedge : \wedge : \wedge : \ldots \\
\text{Valleys} &= \vee : \text{Valleys} && = \vee : \vee : \vee : \vee : \vee : \ldots \\
\text{Tyrol} &= \textbf{zip}(\text{Peaks}, \text{Valleys}) && = \wedge : \vee : \wedge : \vee : \wedge : \vee : \ldots \\
\text{Folds} &= \textbf{zip}(\text{Tyrol}, \text{Folds}) && = \wedge : \wedge : \vee : \wedge : \wedge : \vee : \vee : \wedge : \wedge : \wedge : \vee : \ldots
\end{aligned}
$$



zip-specifications

We consider finite **zip**-specifications for streams using recursion equations built from:

- alphabet letters c_1, c_2, ...
- stream constructor ':'
- **zip**

Motivating Questions

- Is equivalence of zip-specifications decidable?
- Which class of streams is specifiable?

unzipping: basis for coalgebraic analysis

Using ‘zip-destructors’

$$\mathsf{even}(v) = v(0) : v(2) : v(4) : \dots$$
$$\mathsf{odd}(v) = v(1) : v(3) : v(5) : \dots$$

unzipping can be done:

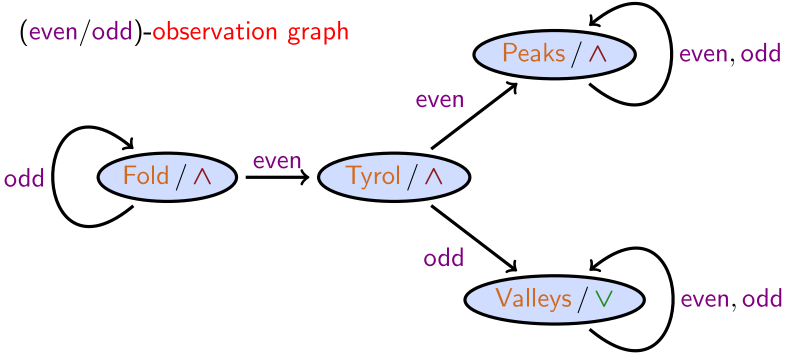
$$\mathsf{even}(\mathbf{zip}(\sigma, \tau)) = \sigma$$
$$\mathsf{odd}(\mathbf{zip}(\sigma, \tau)) = \tau$$

Defining equations:

$$\mathsf{even}(x : \sigma) = x : \mathsf{odd}(\sigma)$$
$$\mathsf{odd}(x : \sigma) = \mathsf{even}(\sigma)$$

paperfolding: (even/odd)-observation graph of **zip**-spec

(even/odd)-observation graph

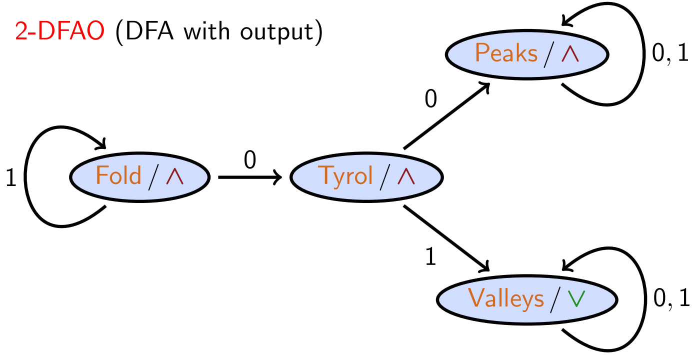


$$\mathsf{Folds} = \mathbf{zip}(\mathsf{Tyrol}, \mathsf{Folds})$$
$$\mathsf{Tyrol} = \mathbf{zip}(\mathsf{Peaks}, \mathsf{Valleys})$$
$$\mathsf{Peaks} = \wedge : \mathsf{Peaks}$$
$$\mathsf{Valleys} = \vee : \mathsf{Valleys}$$

$$\mathsf{Folds} \to^{\omega} \wedge : \wedge : \vee : \wedge : \wedge : \vee : \vee : \wedge : \wedge : \wedge : \vee : \vee : \wedge : \vee : \vee : \wedge \ldots$$

paperfolding: specification as automatic sequence

2-DFAO (DFA with output)

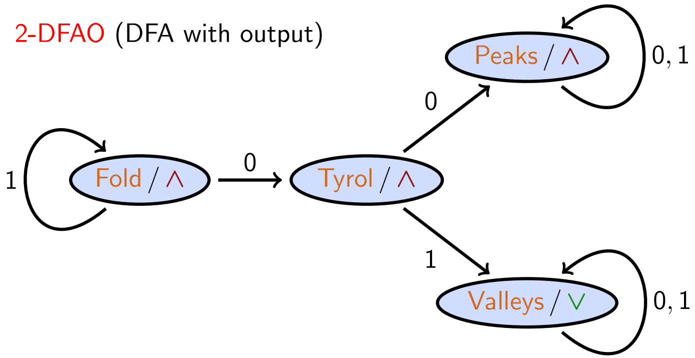


$((2)_2)_{\text{Folds}} = (10)_{\text{Folds}} \xrightarrow{0} (1)_{\text{Tyrol}} \xrightarrow{1} ()_{\text{Valleys}}$ . . . output: $\boxed{\vee}$

$\text{Folds} \to^{\omega} \wedge : \wedge : \boxed{\vee} : \wedge : \wedge : \vee : \vee : \wedge : \wedge : \wedge : \vee : \vee : \wedge : \vee : \vee : \wedge \ldots$

paperfolding: specification as automatic sequence

2-DFAO (DFA with output)



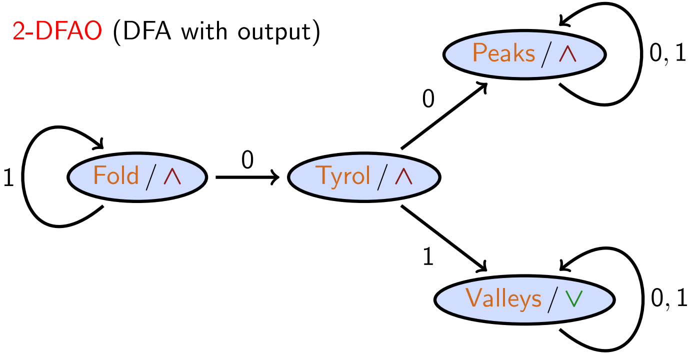
$((2)_2)_{\text{Folds}} = (10)_{\text{Folds}} \xrightarrow{0} (1)_{\text{Tyrol}} \xrightarrow{1} ()_{\text{Valleys}}$ ... output: $\vee$

$((4)_2)_{\text{Folds}} = (100)_{\text{Folds}} \xrightarrow{0} (10)_{\text{Tyrol}} \xrightarrow{1} (1)_{\text{Peaks}} \xrightarrow{1} ()_{\text{Peaks}}$ ... output: $\boxed{\wedge}$

$\text{Folds} \rightarrow^{\omega} \wedge : \wedge : \vee : \wedge : \boxed{\wedge} : \vee : \vee : \wedge : \wedge : \wedge : \vee : \vee : \wedge : \vee : \vee : \wedge \ldots$

paperfolding: specification as automatic sequence

2-DFAO (DFA with output)



$((2)_2)_{\text{Folds}} = (10)_{\text{Folds}} \xrightarrow{0} (1)_{\text{Tyrol}} \xrightarrow{1} ()_{\text{Valleys}}$ ... output: $\vee$

$((4)_2)_{\text{Folds}} = (100)_{\text{Folds}} \xrightarrow{0} (10)_{\text{Tyrol}} \xrightarrow{1} (1)_{\text{Peaks}} \xrightarrow{1} ()_{\text{Peaks}}$ ... output: $\wedge$

$\text{Folds} \to^{\omega} \wedge : \wedge : \vee : \wedge : \wedge : \vee : \vee : \wedge : \wedge : \wedge : \vee : \vee : \wedge : \vee : \vee : \wedge \ldots$

automatic = **zip**-specifiable = finite observation graph

Main Theorem

For streams $\sigma \in \Delta^\omega$ the following properties are equivalent:

1. *σ is 2-automatic.*
2. *σ can be defined by a $\mathbf{zip}_2$-specification.*
3. *The (even/odd)-observation graph of σ is finite.*

automatic = **zip**-specifiable = finite observation graph

Generalises to all $k \geq 2$!

Main Theorem

For streams $\sigma \in \Delta^\omega$ the following properties are equivalent:

1. *σ is k-automatic.*
2. *σ can be defined by a $\mathbf{zip}_k$-specification.*
3. *The $(\pi_{0,k}, \pi_{1,k}, \ldots, \pi_{k-1,k})$-observation graph of σ is finite.*

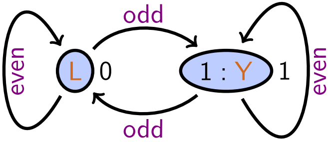
Proof: by a careful coalgebraic analysis.

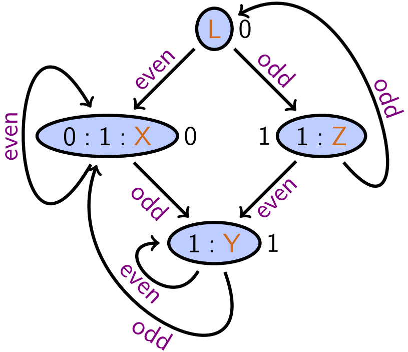
Equivalence of **zip**-specs is decidable

$$\begin{aligned} L &= 0 : X \\ X &= 1 : \mathbf{zip}(X, Y) \\ Y &= 0 : \mathbf{zip}(Y, X) \end{aligned}$$

$$\begin{aligned} L &= 0 : \mathbf{zip}(1 : Z, 1 : X) \\ X &= 1 : \mathbf{zip}(Y, X) \\ Y &= 0 : \mathbf{zip}(Y, 1 : X) \\ Z &= \mathbf{zip}(L, Y) \end{aligned}$$

Zip-specifications are equivalent iff their observation graphs are bisimilar



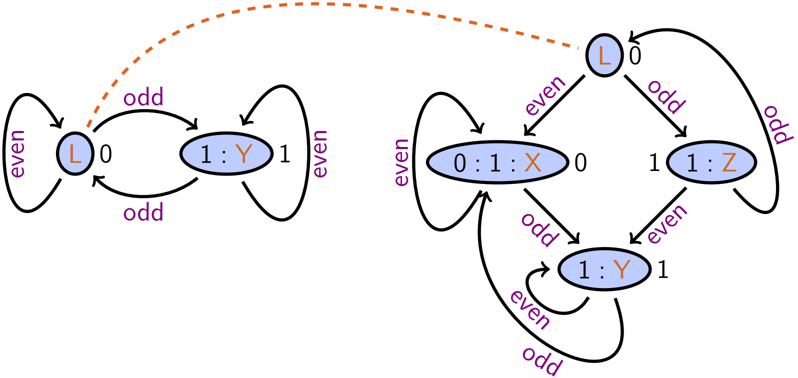


Equivalence of **zip**-specs is decidable

$$\mathsf{L} = 0 : \mathsf{X}$$
$$\mathsf{X} = 1 : \mathbf{zip}(\mathsf{X}, \mathsf{Y})$$
$$\mathsf{Y} = 0 : \mathbf{zip}(\mathsf{Y}, \mathsf{X})$$

$$\mathsf{L} = 0 : \mathbf{zip}(1 : \mathsf{Z}, 1 : \mathsf{X})$$
$$\mathsf{X} = 1 : \mathbf{zip}(\mathsf{Y}, \mathsf{X})$$
$$\mathsf{Y} = 0 : \mathbf{zip}(\mathsf{Y}, 1 : \mathsf{X})$$
$$\mathsf{Z} = \mathbf{zip}(\mathsf{L}, \mathsf{Y})$$

Zip-specifications are equivalent iff their observation graphs are bisimilar

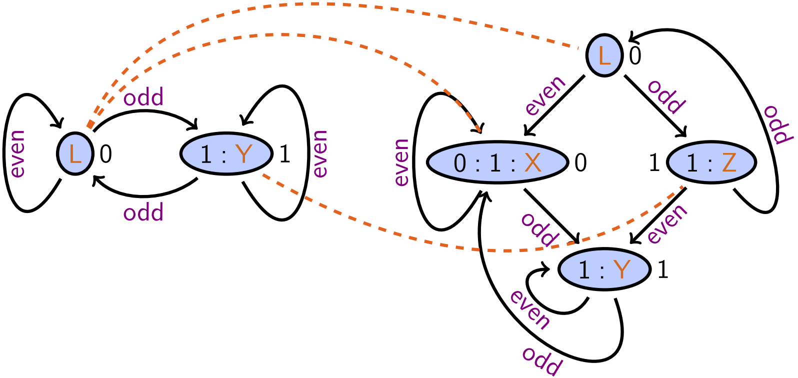


Equivalence of **zip**-specs is decidable

$$L = 0 : X$$
$$X = 1 : \mathbf{zip}(X, Y)$$
$$Y = 0 : \mathbf{zip}(Y, X)$$

$$L = 0 : \mathbf{zip}(1 : Z, 1 : X)$$
$$X = 1 : \mathbf{zip}(Y, X)$$
$$Y = 0 : \mathbf{zip}(Y, 1 : X)$$
$$Z = \mathbf{zip}(L, Y)$$

Zip-specifications are equivalent iff their observation graphs are bisimilar

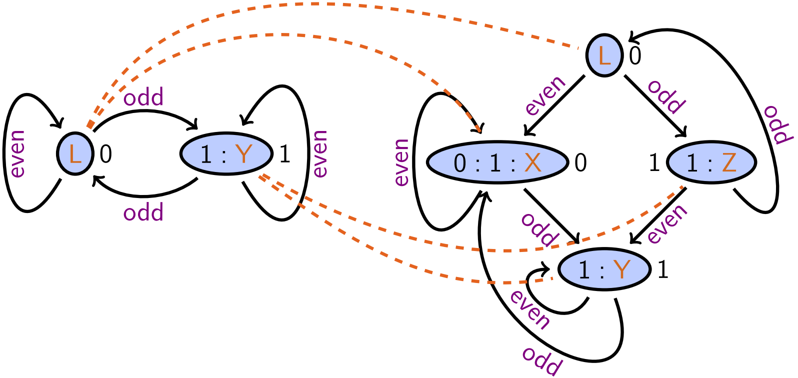


Equivalence of **zip**-specs is decidable

$$\mathsf{L} = 0 : \mathsf{X}$$
$$\mathsf{X} = 1 : \mathbf{zip}(\mathsf{X}, \mathsf{Y})$$
$$\mathsf{Y} = 0 : \mathbf{zip}(\mathsf{Y}, \mathsf{X})$$

$$\mathsf{L} = 0 : \mathbf{zip}(1 : \mathsf{Z}, 1 : \mathsf{X})$$
$$\mathsf{X} = 1 : \mathbf{zip}(\mathsf{Y}, \mathsf{X})$$
$$\mathsf{Y} = 0 : \mathbf{zip}(\mathsf{Y}, 1 : \mathsf{X})$$
$$\mathsf{Z} = \mathbf{zip}(\mathsf{L}, \mathsf{Y})$$

Zip-specifications are equivalent iff their observation graphs are bisimilar

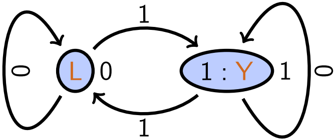


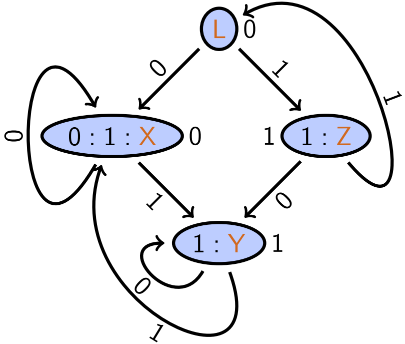
Equivalence of **zip**-specs is decidable

$$\begin{aligned} L &= 0 : X \\ X &= 1 : \mathbf{zip}(X, Y) \\ Y &= 0 : \mathbf{zip}(Y, X) \end{aligned}$$

$$\begin{aligned} L &= 0 : \mathbf{zip}(1 : Z, 1 : X) \\ X &= 1 : \mathbf{zip}(Y, X) \\ Y &= 0 : \mathbf{zip}(Y, 1 : X) \\ Z &= \mathbf{zip}(L, Y) \end{aligned}$$

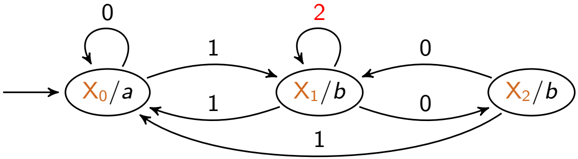
Zip-specifications are equivalent iff their associated DFAO's are language equivalent





zip-mix and mix-automatic

mix-DFAO

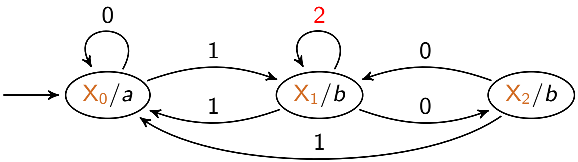


corresponding zip-mix specification:

$$X_0(0) = a \qquad X_0 = \mathbf{zip}_2(X_0, X_1)$$
$$X_1(0) = b \qquad X_1 = \mathbf{zip}_3(X_2, X_0, X_1)$$
$$X_2(0) = b \qquad X_2 = \mathbf{zip}_2(X_1, X_0)$$

zip-mix and mix-automatic

mix-DFAO

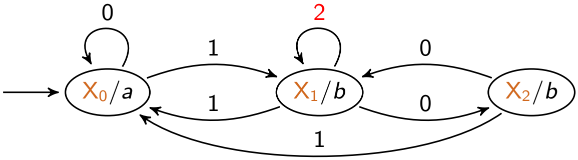


corresponding **zip**-mix specification:

$$\begin{aligned} X_0 &= a : X_0' & \qquad X_0' &= \mathbf{zip}_2(X_1, X_0') \\ X_1 &= b : X_1' & \qquad X_1' &= \mathbf{zip}_3(X_0, X_1, X_2') \\ X_2 &= b : X_2' & \qquad X_2' &= \mathbf{zip}_2(X_0, X_1') \end{aligned}$$

zip-mix and mix-automatic

mix-DFAO



corresponding **zip**-mix specification:

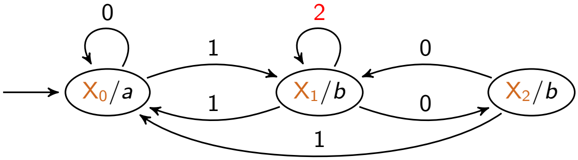
$$
\begin{aligned}
X_0 &= a : X_0' & \qquad X_0' &= \mathbf{zip}_2(X_1, X_0') \\
X_1 &= b : X_1' & \qquad X_1' &= \mathbf{zip}_3(X_0, X_1, X_2') \\
X_2 &= b : X_2' & \qquad X_2' &= \mathbf{zip}_2(X_0, X_1')
\end{aligned}
$$

mix-automatic sequences

- specifiable by/correspond to **zip**-mix-specifications
- properly extend automatic sequences
- decidable: comparison with automatic sequences
- undecidable: equivalence of (**zip** + unzip)-mix specifications

zip-mix and mix-automatic

mix-DFAO



corresponding **zip**-mix specification:

$$X_0 = a : X_0' \qquad X_0' = \mathbf{zip}_2(X_1, X_0')$$
$$X_1 = b : X_1' \qquad X_1' = \mathbf{zip}_3(X_0, X_1, X_2')$$
$$X_2 = b : X_2' \qquad X_2' = \mathbf{zip}_2(X_0, X_1')$$

open questions for mix-automatic sequences

- decidable equivalence problem ?
- mix-automatic $\Longrightarrow$ morphic ?

Our results

- **zip**$_k$-stream-specifications
  - coalgebraic treatment in terms of observation graphs
  - equivalence problem is decidable
    (reduction to bisimilarity/language equiv. of observation graphs)
- Correspondence with automatic sequences:
  - observation graphs correspond to DFAO's
  - k-automatic $=$ **zip**$_k$-definable
- mix-automatic sequences
  - produced by mix-DFAO's, correspondence with **zip**-mix specifications
  - properly extend automatic sequences
  - equivalence problem still decidable?
  - undecidable if unzip-mix operations are added
- dynamic logic representation of automatic sequences