Composing Reo Connector Animations

José Proença  David de Oliveira Costa

Centrum voor Wiskunde en Informatica

IPA Lentedagen, 2007
Outline

1. Motivation
2. What do we mean by animating Reo Connectors
3. Specifying Animations
4. Composing Animations
5. Implementing Animations
6. Conclusion
Outline

1. Motivation
2. What do we mean by animating Reo Connectors
3. Specifying Animations
4. Composing Animations
5. Implementing Animations
6. Conclusion
Simple Reo connectors can have complex behaviour

Formal Semantics exist:
- Coinductive Calculus for Component Connectors;
- Constraint Automata;
- Connector Colouring;
- Intuitionistic Linear Logic;
But

- Often connectors are not very *intuitive*
- Hard to design or debug

Enhance static representations of Reo Connectors
⇒ using Animations.
Motivation

But

- Often connectors are not very *intuitive*
- Hard to design or debug

Enhance static representations of Reo Connectors

⇒ using Animations.
Connector Colouring — Example
Ordering Connector
Connector Colouring — Example
Ordering Connector
Connector Colouring — Example
Ordering Connector

Connector with components

Colouring

J. Proença, D. Costa (CWI)
Connector Colouring — Example
Ordering Connector

See animation
Graphical Elements

Graphical elements presented in animations:

- data
- synchrony
- exclusion
- buffering
- replication
- discrimination
Graphical Elements

<table>
<thead>
<tr>
<th>Graphical elements presented in animations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
</tr>
<tr>
<td>synchrony</td>
</tr>
<tr>
<td>exclusion</td>
</tr>
<tr>
<td>buffering</td>
</tr>
<tr>
<td>replication</td>
</tr>
<tr>
<td>discrimination</td>
</tr>
</tbody>
</table>

Tokens move over channels and components to represent dataflow.
Graphical Elements

Graphical elements presented in animations:

- data
- synchrony
- exclusion
- buffering
- replication
- discrimination

The part of the connector where dataflow occurs in a synchronous slice is coloured in red.
Graphical Elements

Graphical elements presented in animations:

- data
- synchrony
- exclusion
- buffering
- replication
- discrimination

A blue triangle represents the propagation of impossibility of dataflow.
Graphical Elements

Graphical elements presented in animations:

- data
- synchrony
- exclusion
- buffering
- replication
- discrimination

FIFO1 channels can buffer data, which is represented by placing the token inside the channel.
Graphical Elements

Graphical elements presented in animations:

- data
- synchrony
- exclusion
- buffering
- replication
- discrimination

Data (token) is replicated at each node with more than one output.
The filter channel allows only the flow of some specific kind of data, which is represented by colouring the center with the same colour of tokens that are allowed.
Outline

1. Motivation

2. What do we mean by animating Reo Connectors

3. Specifying Animations

4. Composing Animations

5. Implementing Animations

6. Conclusion
The main idea

Using Domain Specific Languages (DSL)

- Simple DSL’s to **draw** and **animate** connectors;
- Conversion to a standard animation tool.

We can go further:

- using the Connector Colouring semantics
- compositionally build the animations
The main idea

Using Domain Specific Languages (DSL)

- Simple DSL’s to draw and animate connectors;
- Conversion to a standard animation tool.

We can go further:

- using the Connector Colouring semantics
- compositionally build the animations
Generating Animations

Connector Graph

Colouring

Animation
Generating Animations

**Connector Graph**

**Colouring**

**Animation Description**

```
move ((token1,blue), 10s, 5s, (15,20));
move ((token2,green), 10s, 5s, (15,1));
create ((token3,blue), 15s, (15,200));
create ((token4,green), 15s, (15,150));
move ((token1,blue), 15s, 6s, (30,12));
move ((token2,green), 15s, 6s, (30,1));
place ((token1,blue), 21s);
move ((token2,green), 15s, 6s, (30
```
Generating Animations

Motivation

Animating Reo?

Specifying Animations

Composing Animations

Implementing Animations

Conclusion

Generating Animations

J. Proença, D. Costa (CWI)

Composing Reo Connector Animations

IPA Lentedagen, 2007
Outline

1. Motivation
2. What do we mean by animating Reo Connectors
3. Specifying Animations
4. Composing Animations
5. Implementing Animations
6. Conclusion
Animation Specifications

Definition

Let C be a connector:

\[ \text{Colouring}_{CT}(C) \rightarrow \text{Set} \left( \text{ActionSpec}^* \right) \]

\[ \text{ActionSpec} = \text{Move Location Location} \]
\[ \quad \mid \text{Place Location} \]
\[ \quad \mid \text{Delete Location} \]
\[ \quad \mid \text{Create Location} \]
\[ \quad \mid \text{Copy Location} \]
\[ \quad \mid \text{NoFlow Location Location} \]
Animation Specifications
Example

Synchronous Drain Channel

\((n_1^i, n_2^i)_{SyncDrain}\)

\{c_1: \quad, c_2: \quad\text{NoFlow}\; n_1 \; n_2, c_3: \quad\text{NoFlow}\; n_2 \; n_1\}

Animation Specification

\[\text{anim}(c_1) = \{ [\text{Move}\; n_1 \; \text{SyncDrain}, \text{Delete}\; \text{SyncDrain}, \text{Move}\; n_2 \; \text{SyncDrain}, \text{Delete}\; \text{SyncDrain} ] \}\]

\[\text{anim}(c_2) = \{ [\text{NoFlow}\; n_1 \; n_2] \}\]

\[\text{anim}(c_3) = \{ [\text{NoFlow}\; n_2 \; n_1] \}\]
Animation Specifications

Example

**Synchronous Drain Channel**

\[(n_1^i, n_2^i)_{\text{SyncDrain}} \quad \{c_1: \quad , c_2: \quad - \quad \rightarrow \quad - \quad , c_3: \quad - \quad \leftarrow \quad - \quad \}\]

**Animation Specification**

\[\text{anim}(c_1) = \{ [\text{Move} \quad n_1 \quad \text{SyncDrain} \quad , \quad \text{Delete} \quad \text{SyncDrain} \quad , \quad \text{Move} \quad n_2 \quad \text{SyncDrain} \quad , \quad \text{Delete} \quad \text{SyncDrain} \} \}\]

\[\text{anim}(c_2) = \{ [\text{NoFlow} \quad n_1 \quad n_2] \}\]

\[\text{anim}(c_3) = \{ [\text{NoFlow} \quad n_2 \quad n_1] \}\]
Outline

1. Motivation
2. What do we mean by animating Reo Connectors
3. Specifying Animations
4. Composing Animations
5. Implementing Animations
6. Conclusion
Composing Animation Specifications

Let's look at an example

\[(n_1^i, n_2^i)_{\text{SyncDrain}} \quad \rightarrow \quad \{ c_1 : \_, c_2 : - -\rightarrow - -, c_3 : - -\leftarrow - - \}\quad \rightarrow \quad \{ c_4 : \_, c_5 : - -\rightarrow - -, c_6 : - -\leftarrow - - \}\quad \rightarrow \quad (m_1^i, m_2^o)_{\text{Sync}}\]

Example

\[\text{anim}(c_1) = \{ [\text{Move } n_1 \text{ SyncDrain}, \text{Delete } \text{SyncDrain}, \text{Move } n_2 \text{ SyncDrain}, \text{Delete } \text{SyncDrain} ] \}\]

\[\text{anim}(c_4) = \{ [\text{Move } m_1 \text{ m2} ] \}\]

\[\text{anim}(c_7) = ?\]
Composing Animation Specifications

Let's look at an example

\[(n_1^i, n_2^i)_{SyncDrain} \rightarrow \{ c_1 : \ldots , c_2 : \rightarrow \ldots , c_3 : \leftarrow \ldots \} \]

\[(m_1^i, m_2^o)_{Sync} \rightarrow \{ c_4 : \ldots , c_5 : \rightarrow \ldots , c_6 : \leftarrow \ldots \} \]

Example

\[ c_7 : \ldots \]

\[ \text{anim}(c_1) = \{ [ \text{Move} \ n_1 \ \text{SyncDrain} \ , \ \text{Delete} \ \text{SyncDrain} \ , \ \text{Move} \ n_2 \ \text{SyncDrain} \ , \ \text{Delete} \ \text{SyncDrain} ] \} \]

\[ \text{anim}(c_4) = \{ [ \text{Move} \ m_1 \ m_2 ] \} \]

\[ \text{anim}(c_7) = ? \]
Composing Animation Specifications

Let's look at an example

\[
\begin{align*}
(n^1_1, n^1_2)_{\text{SyncDrain}} & \rightarrow \{ c_1 : \_\_\_ , c_2 : \_\_\_\_\_\_\_ , c_3 : \_\_\_\_\_\_\_ \} \\
(m^1_1, m^2_2)_{\text{Sync}} & \rightarrow \{ c_4 : \_\_\_\_\_ , c_5 : \_\_\_\_\_\_\_ , c_6 : \_\_\_\_\_\_\_ \}
\end{align*}
\]

Example

\[
\begin{align*}
\text{anim}(c_1) &= \{ [ \text{Move} \ n_1 \ \text{SyncDrain} \ 
\text{, Delete} \ \text{SyncDrain} \ 
\text{, Move} \ n_2 \ \text{SyncDrain} \ 
\text{, Delete} \ \text{SyncDrain} ] \} \\
\text{anim}(c_4) &= \{ [ \text{Move} \ m_1 \ m_2 ] \}
\end{align*}
\]
Composing Animation Specifications

Let's look at an example:

\[
\begin{align*}
(n_1^n, n_2^n)_{\text{SyncDrain}} & \quad \Rightarrow \quad \{ c_1 : \text{---}, c_2 : \text{---->----}, c_3 : \text{----<----} \} \\
(m_1^i, m_2^o)_{\text{Sync}} & \quad \Rightarrow \quad \{ c_4 : \text{---}, c_5 : \text{---->----}, c_6 : \text{----<----} \}
\end{align*}
\]

**Example**

\[
\begin{align*}
\text{anim}(c_1) &= \{ \text{Move } n_1 \text{ SyncDrain}, \text{Delete } \text{SyncDrain}, \text{Move } n_2 \text{ SyncDrain}, \text{Delete } \text{SyncDrain} \} \\
\text{anim}(c_4) &= \{ \text{Move } m_1 \text{ m}_2 \}
\end{align*}
\]

\[
\text{anim}(c_7) = ?
\]
Composing Animation Specifications

\[
\text{anim}(c_7) = \text{anim}(c_1 \cup c_4) \\
= \text{anim}(c_1) \cup \text{anim}(c_4) \\
= \{ [\text{Move } n_1 \text{ SyncDrain} \\
, \text{Delete} \text{ SyncDrain} \\
, \text{Move} n_2 \text{ SyncDrain} \\
, \text{Delete} \text{ SyncDrain} ] \\
[\text{Move } m_1 \text{ m}_2 ] \} \\
\]

General rule

\[
\text{anim}(c_1 \cup \ldots \cup c_n) = \text{anim}(c_1) \cup \ldots \cup \text{anim}(c_n)
\]
Composing Animation Specifications

\[
\text{anim}(c_7) = \text{anim}(c_1 \cup c_4) \\
= \text{anim}(c_1) \cup \text{anim}(c_4) \\
= \{ [\text{Move } n_1 \text{ SyncDrain} \\
, \text{Delete} \text{ SyncDrain} \\
, \text{Move} n_2 \text{ SyncDrain} \\
, \text{Delete} \text{ SyncDrain} ] \\
[\text{Move } m_1 \text{ } m_2 ] \} 
\]

General rule

\[
\text{anim}(c_1 \cup \ldots \cup c_n) = \text{anim}(c_1) \cup \ldots \cup \text{anim}(c_n) 
\]
Composing Animation Specifications

\[ \text{anim}(c_7) = \text{anim}(c_1 \cup c_4) = \text{anim}(c_1) \cup \text{anim}(c_4) = \{ \text{[ Move } n_1 \text{ SyncDrain}, \text{ Delete } \text{SyncDrain}, \text{ Move } n_2 \text{ SyncDrain}, \text{ Delete } \text{SyncDrain} ] \} \text{[ Move } m_1 \text{ } m_2 \text{ ]} \}

General rule

\[ \text{anim}(c_1 \cup \ldots \cup c_n) = \text{anim}(c_1) \cup \ldots \cup \text{anim}(c_n) \]
Outline

1. Motivation
2. What do we mean by animating Reo Connectors
3. Specifying Animations
4. Composing Animations
5. Implementing Animations
6. Conclusion
Animation Description

- Lower lever:
  - What tokens
  - When
  - How long
  - Where

- Multiple Steps, each derived from a colouring;
- Steps can be combined, by incrementing the When value;
- Each Animation Step:
  - actions, absence of flow, changes to the environment

\[
Action = \begin{cases} 
  move \ (Token, \ When, \ Duration, \ From, \ To) \\
  place \ (Token, \ When, \ Where) \\
  delete \ (Token, \ When) \\
  create \ (Token, \ When, \ Where)
\end{cases}
\]
Animation Description

- Lower lever:
  - What tokens
  - When
  - How long
  - Where

- Multiple Steps, each derived from a colouring;

- Steps can be combined, by incrementing the **When** value;

- Each Animation Step:
  - **actions**, absence of flow, changes to the environment

\[
Action \quad = \quad \begin{cases} 
\text{move} & (\text{Token, When, Duration, From, To}) \\
\text{place} & (\text{Token, When, Where}) \\
\text{delete} & (\text{Token, When}) \\
\text{create} & (\text{Token, When, Where}) 
\end{cases}
\]
Animation Description

- **Lower lever:**
  - What tokens
  - When
  - How long
  - Where

- **Multiple Steps**, each derived from a colouring;

- **Steps can be combined**, by incrementing the *When* value;

- **Each Animation Step:**
  - actions, absence of flow, changes to the environment

\[
\text{Action} = \begin{align*}
\text{move} & (\text{Token}, \text{When}, \text{Duration}, \text{From}, \text{To}) \\
\text{place} & (\text{Token}, \text{When}, \text{Where}) \\
\text{delete} & (\text{Token}, \text{When}) \\
\text{create} & (\text{Token}, \text{When}, \text{Where})
\end{align*}
\]
Animation Description

- Lower lever:
  - What tokens
  - When
  - How long
  - Where
- Multiple Steps, each derived from a colouring;
- Steps can be combined, by incrementing the When value;
- Each Animation Step:
  - actions, absence of flow, changes to the environment

\[
Action = \begin{cases} 
\text{move} & (\text{Token}, \text{When}, \text{Duration}, \text{From}, \text{To}) \\
\text{place} & (\text{Token}, \text{When}, \text{Where}) \\
\text{delete} & (\text{Token}, \text{When}) \\
\text{create} & (\text{Token}, \text{When}, \text{Where}) 
\end{cases}
\]
From Previous Example

Animating colouring $c7$ – only flow

(0,0) (5,0) (10,0)

$\text{anim}(\epsilon)
= \{ \text{Move } n1 \text{ SyncDrain, Delete SyncDrain, Move } n2 \text{ SyncDrain, Delete SyncDrain, Move } m1 \text{ m2 } \}

First block not enabled; Second block evaluated.

First block evaluated.

move ( , 30, 20, (0,0), (5,0))
move ( , 50, 10, (5,0), (7.5,0))
delete ( , 50)
move ( , 50, 5, (10,0), (7.5,0))
delete ( , 50)
From Previous Example

Animating colouring \(c7\) – only flow

\[(0,0) \rightarrow (5,0) \rightarrow (10,0)\]

\[m1 \quad m2 \equiv n1 \downarrow n2\]

\[\text{SyncDrain}\]

\[\text{anim(}\quad\text{)}\]
\[= \{ \text{Move } n1 \text{ SyncDrain}, \text{Delete } \text{SyncDrain}, \text{Move } n2 \text{ SyncDrain}, \text{Delete } \text{SyncDrain} \}, \{ \text{Move } m1 \quad m2 \} \}

First block not enabled;
Second block evaluated.

First block evaluated.

move (\ , 30, 20, (0,0), (5,0))
move (\ , 50, 10, (5,0), (7.5,0))
delete (\ , 50)
move (\ , 50, 5, (10,0), (7.5,0))
delete (\ , 50)
From Previous Example

Animating colouring $c7$ – only flow

$\begin{align*}
(0,0) & \quad (5,0) & \quad (10,0) \\
\begin{array}{c}
m1 \quad m2 \equiv n1 \downarrow n2 \\
\text{SyncDrain}
\end{array}
\end{align*}$

$\text{anim}()$

$= \{ [ \text{Move} \ n1 \ \text{SyncDrain} , \ \text{Delete} \ \text{SyncDrain} , \ \text{Move} \ n2 \ \text{SyncDrain} , \ \text{Delete} \ \text{SyncDrain}] , [ \text{Move} \ m1 \ m2 ] \}$

First block not enabled;
Second block evaluated.

First block evaluated.

move ( , 30, 20, (0,0), (5,0))
move ( , 50, 10, (5,0), (7.5,0))
delete ( , 50)
move ( , 50, 5, (10,0), (7.5,0))
delete ( , 50)
From Previous Example
Animating colouring \(c7\) – only flow

\[
\begin{align*}
(0,0) & \quad (5,0) & \quad (10,0) \\
\uparrow & \quad \uparrow & \quad \uparrow \\
m1 & \quad m2 & \equiv n1 \downarrow n2 \\
\text{SyncDrain}
\end{align*}
\]

\[
\text{anim}() = \{ [\text{Move} \ n1 \ \text{SyncDrain} \\
, \text{Delete} \ \text{SyncDrain} \\
, \text{Move} \ n2 \ \text{SyncDrain} \\
, \text{Delete} \ \text{SyncDrain} ] \\
, [\text{Move} \ m1 \ m2 ] \}
\]

First block not enabled; Second block evaluated.

First block evaluated.

- move \((\quad , 30, 20, (0,0), (5,0))\)
- move \((\quad , 50, 10, (5,0), (7.5,0))\)
- delete \((\quad , 50)\)
- move \((\quad , 50, 5, (10,0), (7.5,0))\)
- delete \((\quad , 50)\)
From Previous Example

Animating colouring $c7$ – only flow

\[
(0,0) \quad (5,0) \quad (10,0)
\]

\[
\bullet \quad \bullet \quad \bullet
\]

\[
m1 \quad m2 \equiv n1 \downarrow n2
\]

\[
\text{SyncDrain}
\]

\[
\text{anim}()
\]

\[
= \{ \text{Move } n1 \text{ SyncDrain}, \text{Delete } \text{SyncDrain}, \text{Move } n2 \text{ SyncDrain}, \text{Delete } \text{SyncDrain}, \text{Move } m1 \text{ m2 } \}
\]

First block not enabled; Second block evaluated.

move \((\bullet, 30, 20, (0,0), (5,0))\)
move \((\ , 50, 10, (5,0), (7.5,0))\)
delete \((\ , 50)\)
move \((\ , 50, 5, (10,0), (7.5,0))\)
delete \((\ , 50)\)
From Previous Example

Animating colouring $c7$ – only flow

$$\begin{align*}
(0,0) & \rightarrow (5,0) & \rightarrow (10,0) \\
\text{m1} & \quad \text{m2} \equiv \text{n1} \downarrow \quad \text{n2} \\
\text{move} & \quad \text{n1 SyncDrain} & \text{Deleting SyncDrain} & \text{move} \\
\text{move} & \quad \text{n2 SyncDrain} & \text{Deleting SyncDrain} & \text{move} \\
\text{move} & \quad \text{m1 m2} & \text{Deleting SyncDrain} & \text{move}
\end{align*}$$

$\text{anim(_______)}$

$= \{ \text{[ Move n1 SyncDrain, Delete SyncDrain, Move n2 SyncDrain, Delete SyncDrain]}, \text{[ Move m1 m2]} \}$

First block not enabled; Second block evaluated.

First block evaluated.

move ($\bullet$, 30, 20, (0,0), (5,0))
move ( , 50, 10, (5,0), (7.5,0))
delete ( , 50)
move ( , 50, 5, (10,0), (7.5,0))
delete ( , 50)
From Previous Example

Animating colouring $c7$ – only flow

\[(0,0) \rightarrow (5,0) \rightarrow (10,0)\]

\[m1 \quad m2 \equiv n1 \downarrow \quad n2\]

\[\text{SyncDrain}\]

\[\text{anim}()\]
\[= \{ \text{Move } n1 \text{ SyncDrain} , \text{Delete } \text{SyncDrain} , \text{Move } n2 \text{ SyncDrain} , \text{Delete } \text{SyncDrain} \} , \{ \text{Move } m1 \quad m2 \} \]

\[\text{at } (5,0), \text{ time } 50\]

\[\text{at } (10,0), \text{ time } 30\]

First block not enabled; Second block evaluated.

First block evaluated.

\[\text{move } (\bullet, 30, 20, (0,0), (5,0))\]
\[\text{move } (\bullet, 50, 10, (5,0), (7.5,0))\]
\[\text{delete } (\bullet, 50)\]
\[\text{move } (\bullet, 50, 5, (10,0), (7.5,0))\]
\[\text{delete } (\bullet, 50)\]
**From Previous Example**

**Animating colouring c7 – only flow**

\[(0,0) \rightarrow (5,0) \rightarrow (10,0)\]

\[\text{n1 \Downarrow n2} \quad \text{SyncDrain}\]

\[\text{anim}() = \{ [\text{Move n1 SyncDrain}, \text{Delete SyncDrain}, \text{Move n2 SyncDrain}, \text{Delete SyncDrain}] , [\text{Move m1 m2}] \} \]

- Blue at (5,0), time 50
- Green at (10,0), time 50

First block not enabled; Second block evaluated.

First block evaluated.

- Move (\(\bullet\), 30, 20, (0,0), (5,0))
- Move (\(\bullet\), 50, 10, (5,0), (7.5,0))
- Delete (\(\bullet\), 50)
- Move (\(\bullet\), 50, 5, (10,0), (7.5,0))
- Delete (\(\bullet\), 50)
From Previous Example
Animating colouring $c7$ – only flow

$$\begin{align*}
(0,0) & \quad (5,0) & \quad (10,0) \\
m1 & \quad m2 & \equiv n1 \downarrow n2 & \quad \text{SyncDrain}
\end{align*}$$

\[ \text{anim}() = \{ \begin{array}{l}
\text{Move } n1 \text{ SyncDrain }, \\
\text{Delete } \text{SyncDrain}, \\
\text{Move } n2 \text{ SyncDrain}, \\
\text{Delete } \text{SyncDrain} \\
\text{Move } m1 \text{ m2 } \end{array} \} \]

- \text{at } (7.5,0), \text{ time } 60
- \text{at } (7.5,0), \text{ time } 60

First block not enabled; Second block evaluated.

First block evaluated.

- \text{move } (\text{ }, 30, 20, (0,0), (5,0))
- \text{move } (\text{ }, 50, 10, (5,0), (7.5,0))
- \text{delete } (\text{ }, 50)
- \text{move } (\text{ }, 50, 5, (10,0), (7.5,0))
- \text{delete } (\text{ }, 50)
Obtaining Animation Descriptions

Parameterised transformation on:
- speed,
- fading time,
- time between steps
Obtaining Animation Descriptions

Parameterised transformation on:
- speed,
- fading time,
- time between steps

Missing time information

Animation Description:
- move ((token1,blue), 10s, 5s, (15,20))
- move ((token2,green), 10s, 5s, (15,1))
- create ((token3,blue), 15s, (15,200))
- create ((token4,green), 15s, (15,150))
- move ((token1,blue), 15s, 6s, (30,12))
- move ((token2,green), 15s, 6s, (30,1))
- place ((token1,blue), 21s)
- move ((token2,green), 15s, 6s, (30)
The Algorithm
A breath-first traversal

A breath-first traversal

t=0

\[ \text{Diagram} \]

\[ \text{Diagram} \]

J. Proença, D. Costa (CWI)  Composing Reo Connector Animations  IPA Lentedagen, 2007 21 / 26
The Algorithm
A breath-first traversal

The Algorithm
A breath-first traversal

\[ t=0 \]

\[ A \]

\[ B \]

\[ C \]
The Algorithm
A breath-first traversal
The Algorithm
A breath-first traversal
The Algorithm
A breadth-first traversal
The Algorithm
A breadth-first traversal
The Algorithm
A breath-first traversal
The Algorithm
A breath-first traversal

The Algorithm
A breath-first traversal
The Algorithm
A breath-first traversal

J. Proença, D. Costa (CWI)
Composing Reo Connector Animations
IPA Lentedagen, 2007
The Algorithm
A breath-first traversal
The Algorithm
A breath-first traversal
Outline

1. Motivation
2. What do we mean by animating Reo Connectors
3. Specifying Animations
4. Composing Animations
5. Implementing Animations
6. Conclusion
Conclusion and Future Work

- Precise and *intuitive* representation of the behaviour of circuits;
- Animations produced by composing Animation Steps;
- Steps can be composed.

- Framework for developing Reo Connectors;
- Interaction: choosing existing components and non-deterministic choices during animation.

See animations online: [www.cwi.nl/~proenca/webreo](http://www.cwi.nl/~proenca/webreo)
Conclusion and Future Work

- Precise and *intuitive* representation of the behaviour of circuits;
- Animations produced by composing Animation Steps;
- Steps can be composed.

- Framework for developing Reo Connectors;
- Interaction: choosing existing components and non-deterministic choices during animation.

See animations online: [www.cwi.nl/~proenca/webreo](http://www.cwi.nl/~proenca/webreo)
Related work

- David Harel’s statecharts (Statemate and Rhapsody tools)
  *Reactive Animations* — Flash animations interacting via TCP/IP with Rhapsody executing statecharts.

- *Animation of Behaviour Models*, by Magee et al.; extension of LTS with mappings from labels to actions and conditions; based on Timed Automata to allow composition; resulting XML used to generate JavaBeans.

- *Goal-Oriented Requirements Animation*, by Tran Van, van Lamsweerde, Massonet, Ponsard; good way to communicate with stakeholders; integrated in the FAUST formal analysis suite; multiple stakeholders can interact with the animation over the Internet.
Back to Dave