Labelled Transition Systems (I)

Luís S. Barbosa

DI-CCTC
Universidade do Minho
Braga, Portugal

22 February, 2010

Reactive systems

Reactive system

system that computes by reacting to stimuli from its environment along its overall computation

- in contrast to sequential systems whose meaning is defined by the results of finite computations, the behaviour of reactive systems is mainly determined by interaction and mobility of non-terminating processes, evolving concurrently.
- observation $\Leftrightarrow$ interaction
- behaviour $\Leftrightarrow$ a structured record of interactions

Reactive systems

Concurrency vs interaction

$$x := 0;$$
$$x := x + 1 \mid x := x + 2$$

- both statements in parallel could read x before it is written
- which values can x take?
- which is the program outcome if exclusive access to memory and atomic execution of assignments is guaranteed?

Models of computation for continuous interaction

two reactive systems you are already familiar with

Functions $f : O \longleftarrow I$

- one-step, input-output behaviour
- but what about functions manipulating infinite data structures?

$$\text{merge} : A^\omega \longleftarrow A^\omega \times A^\omega$$

Automata

- multi-step behaviour: accepted language

Models of computation for continuous interaction

two reactive systems you are already familiar with

Functions $f : O \longleftarrow I$

- one-step, input-output behaviour
- but what about functions manipulating infinite data structures?

$$\text{merge} : A^\omega \longleftarrow A^\omega \times A^\omega$$

Automata

- multi-step behaviour: accepted language

Ex: Functions over streams

Streams are coalgebraic structures: specified by observers

$$\langle \mathrm{hd}, \mathrm{tl} \rangle : A \times A^{\omega} \longleftarrow A^{\omega}$$

- Function $\langle \mathrm{hd}, \mathrm{tl} \rangle$ is the observation structure of A^{ω}.
- The shape of such an observation is given by functor $\mathsf{T} : A \times X \longleftarrow X$ for which $\langle \mathrm{hd}, \mathrm{tl} \rangle$ is a coalgebra.

Coalgebra

a lens:	○⌢○
a tool box:	⋒ ◫
an observation structure:	○⌢○ universe $\xleftarrow{\alpha}$ universe
an assembly process:	artifact $\xleftarrow{d}$ ⋒ ◫ artifact

$$\alpha : \mathsf{F}\, U \longleftarrow U$$

- coalgebras describe transition systems
- and abstract behaviour types as (final) coalgebras
- compare with (initial) algebras and (finite) data structures

Coalgebra

a lens:	
a tool box:	
an observation structure:	 universe $\xleftarrow{\alpha}$ universe
an assembly process:	artifact $\xleftarrow{d}$  artifact

$$\alpha : \mathsf{F}\, U \longleftarrow U$$

- coalgebras describe transition systems
- and abstract behaviour types as (final) coalgebras
- compare with (initial) algebras and (finite) data structures

Ex: Functions over streams

- Coalgebras

$$p \;=\; \langle \mathsf{at}, \mathsf{m} \rangle : A \times U \longleftarrow U$$

for the same functor, relate through morphisms:
structure-preserving functions,

$$\begin{array}{ccc} U & \xrightarrow{\langle \mathsf{at},\mathsf{m} \rangle} & A \times U \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle \mathsf{id} \times h} \\ V & \xrightarrow{\langle \mathsf{at}',\mathsf{m}' \rangle} & A \times V \end{array}$$

$$\mathsf{at} = \mathsf{at}' \cdot h \quad \text{and} \quad h \cdot \mathsf{m} = \mathsf{m}' \cdot h$$

- The behaviour of $\langle \mathsf{at}, \mathsf{m} \rangle$, from an initial value u, is given by successive observations:

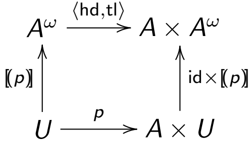
$$[\![p]\!]\; u \;=\; [\mathsf{at}\; u, \mathsf{at}\; (\mathsf{m}\; u), \mathsf{at}\; (\mathsf{m}\; (\mathsf{m}\; u)), \ldots]$$

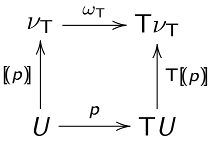
originating a stream of A values.

Ex: Functions over streams

$$\langle \mathsf{hd}, \mathsf{tl} \rangle : A \times A^\omega \longleftarrow A^\omega$$

is final, i.e. characterised by the following universal property: from any other coalgebra p there is a unique morphism $[\![p]\!]$ st





$$k = [\![p]\!] \quad \Leftrightarrow \quad \omega_\mathsf{T} \cdot k = \mathsf{T}\ k \cdot p$$

from where one derives the usual toolkit:

cancelation $\quad \omega_\mathsf{T} \cdot [\![p]\!] = \mathsf{T}\ [\![p]\!] \cdot p$

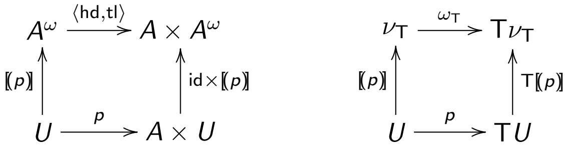
reflection $\quad [\![\omega_\mathsf{T}]\!] = \mathrm{id}_{\nu_\mathsf{T}}$

fusion $\quad [\![p]\!] \cdot h = [\![q]\!] \quad \text{if} \quad p \cdot h = \mathsf{T}\ h \cdot q$

Ex: Functions over streams

$$\langle \mathsf{hd}, \mathsf{tl} \rangle : A \times A^\omega \longleftarrow A^\omega$$

is final, i.e. characterised by the following universal property: from any other coalgebra p there is a unique morphism $[\![p]\!]$ st



$$k = [\![p]\!] \quad \Leftrightarrow \quad \omega_\mathsf{T} \cdot k = \mathsf{T}\ k \cdot p$$

from where one derives the usual toolkit:

$$\begin{array}{rrcl}
\text{cancelation} & \omega_\mathsf{T} \cdot [\![p]\!] & = & \mathsf{T}\ [\![p]\!] \cdot p \\
\text{reflection} & [\![\omega_\mathsf{T}]\!] & = & \mathsf{id}_{\nu_\mathsf{T}} \\
\text{fusion} & [\![p]\!] \cdot h & = & [\![q]\!] \quad \text{if} \quad p \cdot h = \mathsf{T}\ h \cdot q
\end{array}$$

Ex: Functions over streams

Behaviour is specified under all observers

Example:

$$
\begin{array}{ccc}
A^\omega & \xrightarrow{\langle \mathsf{hd},\mathsf{tl} \rangle} & A \times A^\omega \\
\uparrow{\scriptstyle \mathsf{rep}} & & \uparrow{\scriptstyle \mathsf{id} \times \mathsf{rep}} \\
A & \xrightarrow{\ \triangle\ } & A \times A
\end{array}
$$

$$\mathsf{rep} \triangleq [\![(\triangle)]\!]$$

Definition by coinduction

$$
\begin{array}{ll}
 & (\mathsf{id} \times \mathsf{rep})\cdot \triangle = \langle \mathsf{hd}, \mathsf{tl}\rangle \cdot \mathsf{rep} \\
\Leftrightarrow & \qquad \{ \ \triangle \text{ definition } \} \\
 & (\mathsf{id} \times \mathsf{rep}) \cdot \langle \mathsf{id}, \mathsf{id}\rangle = \langle \mathsf{hd}, \mathsf{tl}\rangle \cdot \mathsf{rep} \\
\Leftrightarrow & \qquad \{ \ \times \text{ abs and fusion } \} \\
 & \langle \mathsf{id}, \mathsf{rep}\rangle = \langle \mathsf{hd} \cdot \mathsf{rep}, \mathsf{tl} \cdot \mathsf{rep}\rangle \\
\Leftrightarrow & \qquad \{ \ \text{structural equality } \} \\
 & \mathsf{hd} \cdot \mathsf{rep} = \mathsf{id} \quad \wedge \quad \mathsf{tl} \cdot \mathsf{rep} = \mathsf{rep} \\
\Leftrightarrow & \qquad \{ \ \text{going pointwise } \} \\
 & \mathsf{hd}\ (\mathsf{rep}\ a) = a \quad \wedge \quad \mathsf{tl}\ (\mathsf{rep}\ a) = \mathsf{rep}\ a
\end{array}
$$

Exercise: define merge and twist.

Proof by coinduction: merge $(a^\omega, b^\omega) = (ab)^\omega$

$$\mathsf{merge} \cdot (\mathsf{rep} \times \mathsf{rep}) = \mathsf{twist}$$

$$= \quad \{ \text{ merge definition } \}$$

$$[\![\langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle]\!] \cdot (\mathsf{rep} \times \mathsf{rep}) = [\![\langle \pi_1, \mathsf{s} \rangle]\!]$$

$$\Leftarrow \quad \{ \text{ fusion } \}$$

$$\langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle \cdot (\mathsf{rep} \times \mathsf{rep}) = \mathsf{id} \times (\mathsf{rep} \times \mathsf{rep}) \cdot \langle \pi_1, \mathsf{s} \rangle$$

$$= \quad \{ \times \text{ abs and reflection } \}$$

$$\langle \mathsf{hd} \cdot \mathsf{rep} \cdot \pi_1, \mathsf{s} \cdot ((\mathsf{tl} \cdot \mathsf{rep}) \times \mathsf{rep}) \rangle = \mathsf{id} \times (\mathsf{rep} \times \mathsf{rep}) \cdot \langle \pi_1, \mathsf{s} \rangle$$

$$= \quad \{ \mathsf{tl} \cdot \mathsf{rep} = \mathsf{rep} \text{ e } \mathsf{hd} \cdot \mathsf{rep} = \mathsf{id} \}$$

$$\langle \pi_1, \mathsf{s} \cdot (\mathsf{rep} \times \mathsf{rep}) \rangle = \mathsf{id} \times (\mathsf{rep} \times \mathsf{rep}) \cdot \langle \pi_1, \mathsf{s} \rangle$$

$$= \quad \{ \times \text{ abs } \}$$

$$\langle \pi_1, \mathsf{s} \cdot (\mathsf{rep} \times \mathsf{rep}) \rangle = \langle \pi_1, (\mathsf{rep} \times \mathsf{rep}) \cdot \mathsf{s} \rangle$$

$$= \quad \{ \mathsf{s} \text{ natural: } (f \times g) \cdot \mathsf{s} = \mathsf{s} \cdot (g \times f) \}$$

$$\langle \pi_1, \mathsf{s} \cdot (\mathsf{rep} \times \mathsf{rep}) \rangle = \langle \pi_1, \mathsf{s} \cdot (\mathsf{rep} \times \mathsf{rep}) \rangle$$

Ex: Automata

Definition

$A = \langle \Sigma, S, s_o, F, T \rangle$
where

- Σ is an alphabet
- $S = \{s_0, s_1, s_2, ...\}$ is a set of states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of final states
- $T \subseteq S \times \Sigma \times S$ is the transition relation usually given as a Σ-indexed family of realtions over S:

$$s \xrightarrow{a} s' \;\Leftrightarrow\; \langle s', a, s \rangle \in T$$

- deterministic
- finite
- image finite

Ex: Automata

Definition

$A = \langle \Sigma, S, s_o, F, T \rangle$
where

- Σ is an alphabet
- $S = \{s_0, s_1, s_2, ...\}$ is a set of states
- $s_0 \in S$ is the initial state
- $F \subseteq S$ is the set of final states
- $T \subseteq S \times \Sigma \times S$ is the transition relation usually given as a Σ-indexed family of realtions over S:

$$s \xrightarrow{a} s' \;\Leftrightarrow\; \langle s', a, s \rangle \in T$$

- deterministic
- finite
- image finite

Ex: Automata

automaton behaviour $\Leftrightarrow$ accepted language

Recall that finite automata recognize regular languages, i.e. generated by

- $L_1 + L_2 \triangleq L_1 \cup L_2$ (union)
- $L_1 \cdot L_2 \triangleq \{st \mid s \in L_1, t \in L_2\}$ (concatenation)
- $L^* \triangleq \{\epsilon\} \cup L \cup (L \cdot L) \cup (L \cdot L \cdot L) \cup \ldots$ (iteration)

Ex: Automata

There is a syntax to specify such languages:

$$E \;::=\; \epsilon \mid a \mid E+E \mid E\,E \mid E^*$$

where $a \in \Sigma$.

- which regular expression specifies $\{a, bc\}$?
- and $\{ca, cb\}$?

and an algebra of regular expressions:

$$\begin{aligned}(E_1 + E_2) + E_3 &= E_1 + (E_2 + E_3)\\ (E_1 + E_2)\,E_3 &= E_1\,E_3 + E_2\,E_3\\ E_1\,(E_2\,E_1)^* &= (E_1\,E_2)^*\,E_1\end{aligned}$$

Ex: Automata

There is a syntax to specify such languages:

$$E \quad ::= \quad \epsilon \mid a \mid E + E \mid E\,E \mid E^*$$

where $a \in \Sigma$.

- which regular expression specifies $\{a, bc\}$?
- and $\{ca, cb\}$?

and an algebra of regular expressions:

$$\begin{aligned} (E_1 + E_2) + E_3 &= E_1 + (E_2 + E_3) \\ (E_1 + E_2)\,E_3 &= E_1\,E_3 + E_2\,E_3 \\ E_1\,(E_2\,E_1)^* &= (E_1\,E_2)^*\,E_1 \end{aligned}$$

After thoughts

(from the two examples of reactive systems discussed)

- characterise notions of observation and interaction
- syntax (support for modeling) and semantics (basis for calculation)

After thoughts

... need more general models and theories:

- Several interaction points ($\neq$ functions)
- Non termination (no final states as in automata)
- Need to distinguish normal from anomolous termination (eg deadlock)
- Non determinisim should be taken seriously: the notion of equivalence based on accepted language is blind wrt non determinism

Labelled Transition System

Relational characterization

A LTS over a set $\mathcal{N}$ of names is a pair $\langle S, T \rangle$ where

- $S = \{s_0, s_1, s_2, \ldots\}$ is a set of states
- $T \subseteq S \times \mathcal{N} \times \mathcal{S}$ is the transition relation, often given as an $\mathcal{N}$-indexed family of binary relations

$$s \xrightarrow{a} s' \Leftrightarrow \langle s', a, s \rangle \in T$$

Labelled Transition System

Relational characterization (morphism)

A morphism relating two LTS over $\mathcal{N}$, $\langle S, T\rangle$ and $\langle S', T'\rangle$, is a function$h : S' \longleftarrow S$ st

$$s \xrightarrow{a} s' \quad \Rightarrow \quad h\ s \xrightarrow{a} h\ s'$$

morphisms preserve transitions

Labelled Transition System

Coalgebraic characterization

A LTS over a set $\mathcal{N}$ of names is a pair $\langle S, \text{next} \rangle$ where

- $S = \{s_0, s_1, s_2, \ldots\}$ is a set of states
- $\text{next} : \mathcal{P}S \longleftarrow S \times \mathcal{N}$ is the transition function

Labelled Transition System

Coalgebraic characterization (morphism)

A morphism $h : \langle S', \text{next}' \rangle \longleftarrow \langle S, \text{next} \rangle$ is a function $h : S' \longleftarrow S$ st the following diagram commutes

$$
\begin{array}{ccc}
S \times \mathcal{N} & \xrightarrow{\text{next}} & \mathcal{P}S \\
{\scriptstyle h \times \text{id}} \downarrow & & \downarrow {\scriptstyle \mathcal{P}h} \\
S' \times \mathcal{N} & \xrightarrow{\text{next}'} & \mathcal{P}S'
\end{array}
$$

i.e.,

$$\mathcal{P}h \cdot \text{next} \quad = \quad \text{next}' \cdot (h \times \text{id})$$

or, going pointwise,

$$\{h\,x \mid x \in \text{next}\ \langle s, a \rangle\} \quad = \quad \text{next}'\ \langle h\,s, a \rangle$$

Labelled Transition System

Coalgebraic characterization (morphism)

A morphism $h : \langle S', \text{next}' \rangle \longleftarrow \langle S, \text{next} \rangle$

- preseves transitions:

$$s' \in \text{next}\ \langle s, a \rangle \Rightarrow h\ s' \in \text{next}'\ \langle h\ s, a \rangle$$

- reflects transitions:

$$r' \in \text{next}'\ \langle h\ s, a \rangle \Rightarrow \langle \exists\ s' \in S\ :\ s' \in \text{next}\ \langle s, a \rangle :\ r' = h\ s' \rangle$$

(why?)

Comparison

- Both definitions coincide at the object level:

$$\langle s, a, s' \rangle \in T \quad \Leftrightarrow \quad s' \in \text{next}\ \langle s, a \rangle$$

- Wrt morphisms, the relational definition is more general, corresponding, in coalgebraic terms to

$$\mathcal{P}h \cdot \text{next} \quad \subseteq \quad \text{next}' \cdot (h \times \text{id})$$

How can these notions of morphism be used to compare LTS?

Comparison

- Both definitions coincide at the object level:

$$\langle s, a, s' \rangle \in T \quad \Leftrightarrow \quad s' \in \text{next}\ \langle s, a \rangle$$

- Wrt morphisms, the relational definition is more general, corresponding, in coalgebraic terms to

$$\mathcal{P}h \cdot \text{next} \quad \subseteq \quad \text{next}' \cdot (h \times \text{id})$$

How can these notions of morphism be used to compare LTS?