Lição 1: Sistemas de Transição e Comportamento

Luís Soares Barbosa

Sumário

O objectivo desta Lição é introduzir a noção de sistema de transição como modelo para a descrição de sisteams reactivos, assim como as noções associadas de simulação e bissimulação que formam a base para o desenvolvimento de um cálculo para este tipo de sistemas.

Antes, porém, é percorrido um caminho relativamente longo de motivação onde o aluno é convidado a revisitar dois exemplos de modelos de sistemas reactivos que encontrou em pontos anteriores do seu curso: as funções sobre estruturas de dados infinitas e os autómatos. Esses exemplos desempenham duas funções: por um lado contribuem para a identificação de alguns elementos chave na caracterização matemática de um sistema reactivo, por outro exemplificam um roteiro de trabalho. Roteiro que inclui a procura de modelos formais adequados, de notações apropriadas para os descrever e de cálculos para sobre eles raciocinar. Apenas o primeiro aspecto é tratado nesta Lição. Os dois últimos ocuparão, de certa forma, o resto do curso.

1 Interação, Observação e Comportamento

Um sistema é observável sempre que (e só quando) interage com outros sistemas identificados como o seu ambiente. Donde a identificação

$$\text{observação} \quad \equiv \quad \text{interacção}$$

Por exemplo, a interacção de uma *função* $f : I \longrightarrow O$ com o seu ambiente ocorre em dois momentos bem identificados: o da *invocação* e o da *disponibilização do resultado*. O *comportamento* da função, muitas vezes sugestivamente designado por *comportamento 'entrada-saída'*, é precisamente o conjunto formado pelos pares dessas observações, um por cada argumento possível.

O objecto de estudo deste curso, contudo, são sistemas que interagem com o seu ambiente *ao longo* de toda a computação, e não apenas no início e no final desta. Nesta Lição vamos procurar definir um *modelo* para esse tipo de sistemas, que permita abstrair a estrutura que lhes é comum e estuda-los como um objecto matemático.

Comecemos, para isso, com a pergunta óbvia: *que modelos conhecemos já para sistemas deste tipo?*

O conceito de *função*, detalhadamente estudado na disciplina de Cálculo de Programas, parece desadequado: o modelo de interacção subjacente é *one-step*. No entanto, o que se passa com as funções que manipulam estruturas *infinitas*? Consideremos, por exemplo a função merge : $A^\omega \times A^\omega \longrightarrow A^\omega$ que faz a combinação de duas *streams*. De um ponto de vista denotacional trata-se de uma função como as outras: é invocada e produz um resultado. No entanto, esses dois pontos de interacção (à 'entrada' e à 'saída') têm uma duração indefinida precisamente porque os valores em jogo não são finitos. Todos sabemos que na prática este problema é ultrapassado por recurso a mecanismos de avaliação que permitem decompor estas interacções em sequências (infinitas) de interacções mais elementares. No caso concreto são lidos os valores à cabeça das

Notas de apoio às aulas de *Interacção e Concorrência*, Lic. em Ciências da Computação, 2014-15.

streams, utilizados para produzir a cabeça do resultado, iterando o processo, de seguida, sobre o *resto* das *streams* argumento. A avaliação de merge remete, pois, para um processo de interacção continuada.

Um outro modelo deste tipo de interacção foi estudado na disciplina de Métodos de Programação III, associado ao problema do reconhecimento de linguagens. Trata-se da noção de *autómato*. Aqui a interacção é a aceitação de um símbolo terminal na frase em reconhecimento, símbolos esses que constituem a matéria das observações possíveis. Vamos, de seguida, rever alguns elementos básicos destes dois exemplos e ver até onde nos podem levar.

Caso 1: Funções sobre *streams*

A função merge acima referida manipula *streams*, *i.e.*,, sequências infinitas de valores, que designamos por A^ω. O que é típico deste tipo de dados é a impossibilidade de construção: ao contrário das sequências finitas, das árvores binárias ou de qualquer tipo *indutivo*, as *streams* não são especificadas por um conjunto de *construtores* mas por um conjunto de *observadores*. Tudo o que podemos fazer sobre um valor do tipo A^ω é decompo-lo no resultado das duas observações elementares que sobre ele podemos realizar: a identificação da cabeça e a da cauda. As observações podem ser feitas simultaneamente e, por isso, os dois observadores referidos — hd e tl — são compostos por um *split*:

$$\langle \mathsf{hd}, \mathsf{tl} \rangle : A^\omega \longrightarrow A \times A^\omega \tag{1}$$

Por analogia com o caso bem conhecido dos tipos *indutivos* estudados em Cálculo de Programas dizemos:

- A função $\langle \mathsf{hd}, \mathsf{tl} \rangle$ é a estrutura de observação do tipo de dados infinito A^ω.
- A *forma* dessa observação é dada pelo *functor* $\mathsf{T} : X \longrightarrow A \times X$ para o qual a função $\langle \mathsf{hd}, \mathsf{tl} \rangle$ é uma *coalgebra*. O conjunto A^ω é o *portador* dessa coalgebra.

Nota 1 **[Coalgebras]**

No caso geral, o funtor pode ser visto como uma *lente* que se aplica sobre o sistema que se quer observar.

óculos: ◯⌒◯
(*functor*)

estrutura de observação: ◯⌒◯ sistema $\xleftarrow{f}$ sistema
(*coalgebra*)

Diferentes tipos de lentes potenciam diferentes tipos de observações: As lentes podem ser constantes, caso em que o espaço de observações associado é finito e, portanto, desinteressante para esta discussão, como em

- *'opacas'*: ◯⌒◯ $X = \mathbf{1}$
- *black & white*: ◯⌒◯ $X = \mathbb{B}$
- *coloridas*: ◯⌒◯ $X = \mathbb{N}$

ou 'recursivas', no sentido em que parte do sistema é retornada como componente da observação, originando espaços de observações infinitos. Por exemplo,

- *lentes parciais: 'aborta ou prossegue'*: ◯⌒◯ $X = X + \mathbf{1}$

- *lentes que identificam atributos: 'cabeça & cauda'*: $\bigcirc\frown\bigcirc\ X\ =\ A\times X$
- *lentes em que essa identificação é não-determinística*: $\bigcirc\frown\bigcirc\ X\ =\ \mathcal{P}(A\times X)$

- Uma outra estrutura de observação sobre um conjunto portador arbitrário U, por exemplo

$$p\ =\ \langle \mathsf{at}, \mathsf{m}\rangle : U \longrightarrow A\times U \tag{2}$$

define uma nova coalgebra para T. Coalgebras do mesmo tipo podem ser relacionadas através de funções entre os respectivos portadores que preservam a estrutura. I.e.,

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

ou, equacionalmente,

$$\mathsf{at} = \mathsf{at}'\cdot h \quad \mathsf{e} \quad h\cdot \mathsf{m} = \mathsf{m}'\cdot h \tag{3}$$

- O comportamento de $\langle \mathsf{at}, \mathsf{m}\rangle$, a partir de um valor inicial u, é dado por *observações* sucessivas:

$$[\![p]\!]\ u\ =\ <\mathsf{at}\ u,\ \mathsf{at}\ (\mathsf{m}\ u),\ \mathsf{at}\ (\mathsf{m}\ (\mathsf{m}\ u)), ...> \tag{4}$$

originando uma sequência infinita de valores de A, *i.e.*,, um elemento de A^ω.

- De facto a coálgebra em (1) tem um estatuto muito particular: o seu portador é o conjunto de todos os *comportamentos* (*i.e.*,, estruturas de *observações*) possíveis para T-coálgebras. Tal coálgebra é dita *final* e caracterizada pela seguinte propriedade universal: a partir de qualquer outra T-coálgebra p existe um único morfismo $[\![p]\!]$ tal que o seguinte diagrama comuta:

$$\begin{array}{ccc} A^\omega & \xrightarrow{\omega_\mathsf{T}} & A\times A^\omega \\ {\scriptstyle [\![p]\!]}\uparrow & & \uparrow{\scriptstyle \mathsf{T}[\![p]\!]} \\ U & \xrightarrow{p} & \mathsf{T}U \end{array}$$

onde $\omega_\mathsf{T}\ =\ \langle \mathsf{hd}, \mathsf{tl}\rangle$. Daí a classificação de A^ω como um tipo *coindutivo*. Em Cálculo de Programas esta propriedade é expressa pela bem conhecida *lei universal dos anamorfismos*:

$$k = [\![p]\!] \ \Leftrightarrow\ \omega_\mathsf{T}\cdot k = \mathsf{T}\ k\cdot p \tag{5}$$

de onde se deduz o habitual *tool-kit*:

$$\text{cancelamento} \quad \omega_\mathsf{T}\cdot [\![p]\!]\ =\ \mathsf{T}\ [\![p]\!]\cdot p \tag{6}$$

$$\text{reflexão} \quad [\![\omega_\mathsf{T}]\!]\ =\ \mathsf{id}_{\nu_\mathsf{T}} \tag{7}$$

$$\text{fusão} \quad [\![p]\!]\cdot h\ =\ [\![q]\!]\ \text{ if }\ p\cdot h\ =\ \mathsf{T}\ h\cdot q \tag{8}$$

Exercício 1

Recorde o cálculo de tipos *indutivos* que estudou anteriormente e explicite o seu paralelo com os conceitos aqui revistos: *cf.*, construtores, álgebra, álgebra inicial, catamorfismo, etc.

A modelação e o cálculo de funções sobre *streams* (ou, mais genericamente, sobre qualquer tipo coindutivo) é baseada na equivalência subjacente à propriedade (5) [BM97]. A implicação da esquerda para a direita oferece um princípio de *modelação* (ou *definição*):

O sistema é modelado através da especificação do seu comportamento sob todos os observadores.

o que é feito, de novo por dualidade com o caso indutivo, por especificação do *património genético* da função. Por exemplo, o *gene* de uma função gen que a partir de um valor do tipo A gera uma *stream* que repete indefinidamente esse valor é a função *diagonal* $\triangle = \langle \mathsf{id}, \mathsf{id} \rangle$. Assim,

$$\begin{array}{ccc} A^\omega & \xrightarrow{\langle \mathsf{hd},\mathsf{tl}\rangle} & A \times A^\omega \\ {\scriptstyle \mathsf{gen}}\uparrow & & \uparrow{\scriptstyle \mathsf{id}\times\mathsf{gen}} \\ A & \xrightarrow{\triangle} & A \times A \end{array} \tag{9}$$

$$\mathsf{gen} \ \triangleq \ [\!(\triangle)\!] \tag{10}$$

Alguma 'massagem' na equação que traduz a comutatividade do diagrama (9), torna bem claro o princípio de *modelação* referido acima:

$$\begin{array}{ll} & (\mathsf{id} \times \mathsf{gen}) \cdot \triangle \ = \ \langle \mathsf{hd}, \mathsf{tl} \rangle \cdot \mathsf{gen} \\ \equiv & \quad \{ \ \triangle \text{ definição} \ \} \\ & (\mathsf{id} \times \mathsf{gen}) \cdot \langle \mathsf{id}, \mathsf{id} \rangle \ = \ \langle \mathsf{hd}, \mathsf{tl} \rangle \cdot \mathsf{gen} \\ \equiv & \quad \{ \ \times \text{ absorção e fusão} \ \} \\ & \langle \mathsf{id}, \mathsf{gen} \rangle \ = \ \langle \mathsf{hd} \cdot \mathsf{gen}, \mathsf{tl} \cdot \mathsf{gen} \rangle \\ \equiv & \quad \{ \ \text{igualdade estrutural} \ \} \\ & \mathsf{hd} \cdot \mathsf{gen} = \mathsf{id} \ \ \wedge \ \ \mathsf{tl} \cdot \mathsf{gen} = \mathsf{gen} \\ \equiv & \quad \{ \ \text{introduzindo variáveis} \ \} \\ & \mathsf{hd} \ (\mathsf{gen} \ a) = a \ \ \wedge \ \ \mathsf{tl} \ (\mathsf{gen} \ a) = \mathsf{gen} \ a \end{array}$$

Exercício 2

Considere as seguintes definições das funções merge, acima referida, e twist : $A \times A \longrightarrow A^\omega$:

$$\begin{array}{rcl} \mathsf{merge} & \triangleq & [\!(\langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle)\!] \\ \mathsf{twist} & \triangleq & [\!(\langle \pi_1, \mathsf{s} \rangle)\!] \end{array}$$

Explique o seu propósito, trace os diagramas correspondentes e reescreva-as numa versão com variáveis, codificando-as em HASKELL.

Por outro lado, e votando à propriedade (5), recorde-se que a implicação da direita para a esquerda fornece um princípio de *prova*, dito de *coindução*. Como habitualmente, as propriedades universais aplicam-se regra geral através de uma *lei de fusão* — neste caso a equação (8). Mostremos, por exemplo, que

$$\mathsf{merge} \ (a^\omega, b^\omega) \ = \ (ab)^\omega \tag{11}$$

que, na notação *pointfree* usada em Cálculo de Programas, se escreve

$$\mathsf{merge} \cdot (\mathsf{gen} \times \mathsf{gen}) \ = \ \mathsf{twist} \tag{12}$$

Temos, então,

$$
\begin{array}{ll}
 & \mathsf{merge} \cdot (\mathsf{gen} \times \mathsf{gen}) \;=\; \mathsf{twist} \\
= & \quad \{ \text{ merge definição } \} \\
 & [\![\langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle]\!] \cdot (\mathsf{gen} \times \mathsf{gen}) \;=\; [\![\langle \pi_1, \mathsf{s} \rangle]\!] \\
\Leftarrow & \quad \{ \text{ fusão } \} \\
 & \langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle \cdot (\mathsf{gen} \times \mathsf{gen}) \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle \\
= & \quad \{ \text{ } \times \text{ absorção and reflexão } \} \\
 & \langle \mathsf{hd} \cdot \mathsf{gen} \cdot \pi_1, \mathsf{s} \cdot ((\mathsf{tl} \cdot \mathsf{gen}) \times \mathsf{gen}) \rangle \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle \\
= & \quad \{ \text{ } \mathsf{tl} \cdot \mathsf{gen} = \mathsf{gen} \text{ e } \mathsf{hd} \cdot \mathsf{gen} = \mathsf{id} \} \\
 & \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle \\
= & \quad \{ \text{ } \times \text{ absorção } \} \\
 & \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \langle \pi_1, (\mathsf{gen} \times \mathsf{gen}) \cdot \mathsf{s} \rangle \\
= & \quad \{ \text{ s é natural, } \textit{i.e.,,} \; (f \times g) \cdot \mathsf{s} = \mathsf{s} \cdot (g \times f) \} \\
 & \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle
\end{array}
$$

Caso 2: Autómatos

Definição 1 *Um* autómato *(ver, por exemplo, [HU79, How91]) A sobre um 'alfabeto' Σ é um tuplo $A = \langle S, s_o, F, T \rangle$, onde $S = \{s_0, s_1, s_2, ...\}$ é um conjunto de estados no qual se identifica um estado inicial $s_0 \in S$ e um subconjunto de estados finais $F \subseteq S$. A dinâmica do autómato é dada pela relação de transição $T \subseteq S \times \Sigma \times S$.*

Por convenção T é geralmente escrita como uma família Σ-indexada de relações binárias sobre S:

$$s \xrightarrow{a} s' \;\equiv\; \langle s', a, s \rangle \in T \tag{13}$$

Recorde-se, ainda, que um *autómato* A é classificado como

- *Finito*, se S finito.
- Com *imagem finita*, se os estados acessíveis por um mesmo símbolo a partir de cada estado do autómato formam um conjunto finito.
- *Determinístico*, se a relação T é simples, *i.e.,,* se para cada par $\langle s, a \rangle$ há no máximo uma transição $s \xrightarrow{a} s'$

comportamento de um autómato $\equiv$ linguagem aceite

Uma linguagem não é mais que um conjunto de sequências de símbolos em Σ. Conforme estudado anteriormente, autómatos finitos, *determinísticos* ou não, reconhecem a mesma classe de linguagens, ditas *regulares*, *i.e.,,* linguagens geradas por aplicação das operações:

- $L_1 + L_2 \;\triangleq\; L_1 \cup L_2$ (reunião)
- $L_1 \cdot L_2 \;\triangleq\; \{st |\; s \in L_1, t \in L_2\}$ (concatenação)
- $L^* \;\triangleq\; \{\epsilon\} \cup L \cup (L \cdot L) \cup (L \cdot L \cdot L) \cup ...$ (iteração)

Em Métodos de Programação III foi, também, estudado uma notação para *especificar* este tipo de linguagens: as *expressões regulares* sobre Σ, geradas pela seguintes gramática

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

onde $a \in \Sigma$.

Cada construtor de expressões regulares é interpretado como uma operação nas linguagens sobre Σ. Em particular, $+$ é interpretado como a união de linguagens, a justaposição como concatenação, a estrela como iteração, enquanto a cada símbolo em Σ e a ϵ corresponde o respectivo conjunto singular com a sequência unitária formada por esse símbolo. Por exemplo a linguagem $\{a, bc\}$ pode ser especificada pela expressão regular $a + bc$. Similarmente as expressões regulares $(a + b)c$ e $ac + bc$ especificam a linguagem $\{ac, bc\}$.

Com base nesta interpretação é possível encontrar um conjunto de *leis equacionais* entre este tipo de expressões, que permitem *compara-las* e *transformar umas nas outras*. I.e., é possível definir uma *álgebra de expressões regulares*. Por exemplo,

$$(E_1 + E_2) + E_3 \ = \ E_1 + (E_2 + E_3) \tag{14}$$
$$(E_1 + E_2)\,E_3 \ = \ E_1\,E_3 + E_2\,E_3 \tag{15}$$
$$E_1\,(E_2\,E_1)^* \ = \ (E_1\,E_2)^*\,E_1 \tag{16}$$

Dispomos, ainda, de um processo de resolução de *equações* sobre expressões regulares. Trata-se da Regra de Arden que estabelece $X = S^*\,T$ como solução da equação

$$X \ = \ (S\,X) + T \tag{17}$$

Solução é única se $\epsilon \notin S$.

Reflexão

Revimos dois exemplos de sistemas reactivos, *i.e.,*que asseguram uma interacção continuada com o exterior, a partir de material discutido em disciplinas prévias do curso. Em ambos os casos procuramos identificar as noções aplicáveis de *observação* e *comportamento*. Preocupamo-nos, também, por dispor de uma *notação* para para os **modelar** e de um *cálculo* para sobre eles **raciocinar**.

No caso geral, porém, os sistemas reactivos não se restringem às funções sobre estruturas de dados infinitas nem aos autómatos usados no reconhecimento de linguagens. De facto,

- Muitos sistemas reactivos apresentam capacidades múltiplas de interacção com o ambiente: existem várias portas de 'entrada' e 'saída', e não apenas duas como caso das funções (mesmo das funções sobre *streams* ...). Além disso, uma mesma entrada numa mesma porta pode não conduzir sempre à mesma reacção.
- A maior parte dos sistemas reactivos que nos são tecnologicamente familiares (por exemplo, os sistemas operativos ou a internet) nunca terminam a sua computação. Ao contrários dos autómatos, não exsitem, em geral, estados *finais*.
- Em todo o caso, será sempre necessário distinguir entre terminação e situações anómalas (por exemplo, *deadlock*).
- No caso geral, o não-determinismo tem de ser tomado a sério, enquanto, como é sabido, para cada autómato finito não-determinístico existe um autómatos determinístico com igual comportamento (*i.e.*,, que reconhece a mesma liguagem regular). Esta noção de equivalência baseada no reconhecimento da mesma linguagem é cega para o não-determinismo.

Precisamos, pois, de um modelo mais geral para os *sistemas reactivos*. Para ser útil, contudo, ele deverá, tal como os casos particulares aqui revistos,

- Captar as situações/fenómenos reais que classificamos como *sistemas reactivos*.
- Oferecer noções precisas de *observação* (*interacção*) e *comportamento*.
- Ter associada uma notação para podermos construir os *modelos* que os descrevem.
- Ter subjacente um *cálculo*, associado a noções de *equivalência* apropriadas, para podermos estabelecer e verificar as suas propriedades e transformar os seus modelos.

2 Sistemas de Transição

Caracterização Relacional

Definição 2 *Um* sistema de transição etiquetado, *que abreviaremos por* lte, *sobre um conjunto de nomes $\mathcal{N}$ é um par $\langle S, T\rangle$ onde*

- *$S = \{s_0, s_1, s_2, ...\}$ é um conjunto de estados,*
- *e $T \subseteq S \times \mathcal{N} \times \mathcal{S}$ uma relação de transição, geralmente apresentada como uma família $\mathcal{N}$-indexada de relações binárias*

$$s \xrightarrow{a} s' \;\equiv\; \langle s', a, s\rangle \in T \tag{18}$$

A definição de *lte* relaxa a dada na Definição (1), eliminando a componente de estados inicial e finais. Assim, uma noção de *morfismo* entre estes sistemas deverá apenas respeitar a estrutura remanescente, em particular a relação de transição.

Definição 3 *Um* morfismo *entre dois sistemas de transição etiquetados sobre $\mathcal{N}$, $\langle S, T\rangle$ e $\langle S', T'\rangle$, é uma função $h : S \longrightarrow S'$ entre os respectivos espaços de estados tal que*

$$s \xrightarrow{a} s' \quad \Rightarrow \quad h\ s \xrightarrow{a} h\ s' \tag{19}$$

A intuição subjacente a (19) é de que um morfismo deve *preservar* as transições.

Caracterização Coalgébrica

É bem conhecido o isomorfismo entre relações binárias e funções para o conjunto potência, sob o qual a relação de transição T na definição 2 se pode escrever como uma função

$$\mathsf{next} : S \times \mathcal{N} \longrightarrow \mathcal{P}\mathcal{S} \tag{20}$$

que a cada par $\langle$estado, etiqueta$\rangle$ faz corresponder o conjunto de estados para os quais o sistema pode transitar — daí o seu nome.

Note-se que, por *currying*, a função next pode ser escrita como uma coálgebra

$$\overline{\mathsf{next}} : S \longrightarrow (\mathcal{P}S)^{\mathcal{N}} \tag{21}$$

para o functor $\mathsf{T}\,X \;=\; (\mathcal{P}X)^{\mathcal{N}}$

Como vimos na discussão das funções sobre *streams*, a noção de coálgebra, que em Cálculo de Programas desempenhava um papel essencialmente técnico na análise e codificação de funções recursivas (como uma das componentes dos hilomorfismos), tem, de facto, um alcance muito mais vasto. De uma forma geral, coálgebras são modelos abstractos de estruturas de transição genéricas, captadas por observações, do mesmo modo que a noção dual, e mais familiar, de álgebra, modela processos de construção.

Podemos, assim, chegar a uma definição alternativa de *lte*:

Definição 4 *Um* sistema de transições etiquetado *sobre um conjunto de nomes $\mathcal{N}$ é um par $\langle S, \mathsf{next}\rangle$ onde $S = \{s_0, s_1, s_2, ...\}$ é um conjunto de estados e $\mathsf{next} : S \times \mathcal{N} \longrightarrow \mathcal{PS}$ é uma função de transição de estados.*

A noção correspondente de *morfismo* é, de novo, pedida de empréstimo à teoria das coálgebras [Rut00]:

Definição 5 *Um morfismo $h : \langle S, \mathsf{next}\rangle \longrightarrow \langle S', \mathsf{next}'\rangle$ é uma função $h : S \longrightarrow S'$ entre os respectivos espaços de estados tal que o seguinte diagrama comuta*

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

i.e.,,

$$\mathcal{P}h \cdot \mathsf{next} \quad = \quad \mathsf{next}' \cdot (h \times \mathsf{id}) \tag{22}$$

Introduzindo variáveis, a equação (22) vem

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

Mais interessante é, contudo, reparar nas consequências desta definição de morfismo em termos da dinâmica das transições:

Lema 1 *Um morfismo entre sistemas de transição definido pela equação* (22) preserva *e* reflecte *as transições do sistema original,* i.e.,,

$$s' \in \mathsf{next}\ \langle s, a\rangle \Rightarrow h\ s' \in \mathsf{next}'\ \langle h\ s, a\rangle \qquad \textit{preservação} \tag{23}$$

$$r' \in \mathsf{next}'\ \langle h\ s, a\rangle \Rightarrow \exists_{s' \in \mathsf{next}\ \langle s,a\rangle}\ .\ r' = h\ s' \qquad \textit{reflexão} \tag{24}$$

Prova. Exercício 1 □

Comparando estas duas caracterizações dos sistemas de transição, concluímos que

- Ao nível das estruturas coincidem:

$$\langle s, a, s'\rangle \in T \quad \equiv \quad s' \in \mathsf{next}\ \langle s, a\rangle \tag{25}$$

- Ao nível dos morfismos a definição relacional é mais *geral*, *i.e.*,, relaxada, correspondendo, no lado coalgébrico, à inequação:

$$\mathcal{P}h \cdot \mathsf{next} \quad \subseteq \quad \mathsf{next}' \cdot (h \times \mathsf{id}) \tag{26}$$

Em qualquer caso a questão que, de um ponto de vista de engenharia, nos interessa é saber

Como operacionalizar a noção de *morfismo* para comparar sistemas de transição?

Ora qualquer *comparação* tem subjacente uma *pre-ordem*, o que nos conduz às noções de *simulação* e *bissimulação*.

3 Simulação e Bissimulação

A noção de simulação, que constitui o instrumento básico para comparação de sistemas de transição, é uma relação entre estados que capta a capacidade de um deles acompanhar a evolução do sistema a partir do outro, oferecendo em cada passo as mesmas capacidades de interacção. I.e., dados dois estados p e q (de um mesmo ou de dois diferentes sistemas de transição), *q simula p* se toda a transição a partir do último é correspondida por uma transição a partir do primeiro e, adicionalmente, essa capacidade é mantida ao longo de toda a evolução do(s) sistema(s)[1]

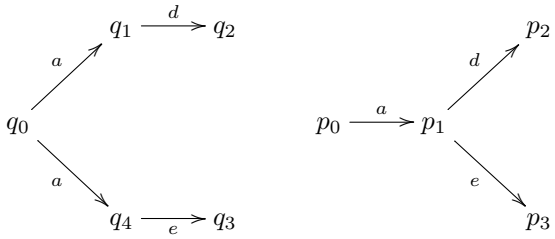
Definição 6 *Dados dois sistemas de transição $\langle S_1, T_1\rangle$ e $\langle S_2, T_2\rangle$ sobre $\mathcal{N}$, uma relação binária $R \subseteq S_1 \times S_2$ é uma* simulação *sse, sempre que $\langle p, q\rangle \in R$ e $a \in \mathcal{N}$,*

$$p \xrightarrow{a} p' \;\Rightarrow\; \exists_{q'}.\; q \xrightarrow{a} q' \;\wedge\; \langle p', q'\rangle \in R \tag{27}$$

Graficamente,

Dizemos que p é simulado por q, e escrevemos $p \lesssim q$, se existir uma simulação *R tal que $\langle p, q\rangle \in R$.*

Exemplo



$$q_0 \lesssim p_0 \qquad \text{por} \quad \{\langle q_0, p_0\rangle, \langle q_1, p_1\rangle, \langle q_4, p_1\rangle, \langle q_2, p_2\rangle, \langle q_3, p_3\rangle\}$$

Lema 2 *A relação de similaridade é uma pré-ordem (*i.e.*,, reflexiva e transitiva)*

Prova. A prova da reflexividade é trivial. Para a transitividade, considere-se

$$\begin{array}{ll} & \langle x, y\rangle \in S \cdot R \;\wedge\; x \xrightarrow{a} x' \\ \equiv & \quad \{\ \text{definição de composição relacional}\ \} \\ & \exists_z .\; \langle x, z\rangle \in S \wedge \langle z, y\rangle \in R \;\wedge\; x \xrightarrow{a} x' \\ \Rightarrow & \quad \{\ S \text{ e } R \text{ são bissimulações}\ \} \\ & \exists_z .\; \langle x, z\rangle \in S \wedge \langle z, y\rangle \in R \;\wedge \\ & \exists_{z'} .\; (z \xrightarrow{a} z' \;\wedge\; \langle x', z'\rangle \in S) \;\wedge \\ & \exists_{y'} .\; (y \xrightarrow{a} y' \;\wedge\; \langle z', y'\rangle \in R) \\ \Rightarrow & \quad \{\ \text{definição de composição relacional}\ \} \\ & y \xrightarrow{a} y' \;\wedge\; \langle x', y'\rangle \in S \cdot R \end{array}$$

[1] Formalmente, diremos que essa capacidade é fechada para a relação de transição do sistema ao qual p pertence.

□

Definição 7 *Dados dois sistemas de transição $\langle S_1, T_1\rangle$ e $\langle S_2, T_2\rangle$ sobre $\mathcal{N}$, uma relação binária $R \subseteq S_1 \times S_2$ é uma* bissimulação *sse, R e a sua conversa, R°, forem simulações. I.e., sse sempre que $(p, q) \in R$ e $a \in \mathcal{N}$,*

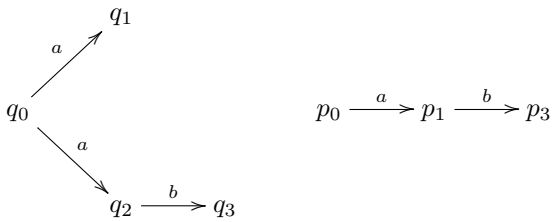
$$p \xrightarrow{a} p' \Rightarrow \exists_{q'}.\ q \xrightarrow{a} q' \ \wedge\ (p', q') \in R \tag{28}$$

$$q \xrightarrow{a} q' \Rightarrow \exists_{p'}.\ p \xrightarrow{a} p' \ \wedge\ (p', q') \in R \tag{29}$$

Dizemos que dois estados p e q são bissimilares, *escrevendo $p \sim q$ se existir uma* bissimulação *R tal que $\langle p, q\rangle \in R$.*

Atenção: a relação de bissimilaridade não é, como se poderia esperar, o fecho simétrico da similaridade. Por exemplo, no diagrama seguinte é fácil concluir que

$$q_0 \lesssim p_0,\ p_0 \lesssim q_0 \quad \text{mas} \quad p_0 \not\sim q_0$$



As definições de $\lesssim$ e $\sim$ têm implícito um princípio de prova, dito *coindutivo*, que será de extrema utilidade neste curso:

Para mostrar que $p \sim q$ (resp., $p \lesssim q$) é suficiente exibir uma *bis(simulação)* contendo $\langle p, q\rangle$.

Dito de outro modo, a relação de *similaridade* (resp., *bissimilaridade*) é a *maior* das simulações (resp., *bissimulações*) sobre o(s) sistema(s) de transição considerado(s). I.e.,

$$\lesssim \triangleq \bigcup\{S|\ S \text{ é uma simulação}\} \quad \text{e} \quad \sim \triangleq \bigcup\{S|\ S \text{ é uma bissimulação}\} \tag{30}$$

Nota 2 [O Teorema de Knaster-Tarski]

Estas relações podem ainda ser caracterizadas como *maiores* pontos fixos no recticulado das relações binárias sobre o espaço de estados relevante. Por exemplo, $\sim$ é o maior ponto fixo da função

$$\mathcal{F}(R)\,\langle p, q\rangle \ = \ \begin{cases} p \xrightarrow{a} p' \Rightarrow & \exists_{q'}.\ q \xrightarrow{a} q' \ \wedge\ (p', q') \in R \\ q \xrightarrow{a} q' \Rightarrow & \exists_{p'}.\ p \xrightarrow{a} p' \ \wedge\ (p', q') \in R \end{cases} \tag{31}$$

Como $\mathcal{F}$ é monótona, o resultado decorre do teorema de Knaster-Tarski, estudado em Matemática Discreta. Como recorda, trata-se um resultado clássico sobre a existência de pontos fixos de funções monótonas em recticulados completos, publicado originalmente em [Tar55], que a seguir se reproduz chamando a atenção para a elegância da prova.

Teorema 3.1 *Seja $(U, \sqsubseteq)$ um recticulado completo e $f : U \longrightarrow U$ uma função monótona,* i.e.*, tal que se $x \sqsubseteq y$, então $f(x) \sqsubseteq f(y)$. Então*

$$m = \bigsqcap \{x \in U |\ f(x) \sqsubseteq x\}$$
$$M = \bigsqcup \{x \in U |\ x \sqsubseteq f(x)\}$$

constituem, respectivamente, o menor e o maior ponto fixo de f.

Prova. Comecemos por provar que m é o menor ponto fixo de f. Seja $X = \{x \in U |\ f(x) \sqsubseteq x\}$ e tomemos um qualquer $x \in X$. Claramente, $m \sqsubseteq x$ e, sendo f monótona, $f(m) \sqsubseteq f(x)$. Por outro lado $f(x) \sqsubseteq x$, uma vez que $x \in X$. Logo podemos concluir que, para todo o $x \in X$, $f(m) \sqsubseteq x$. Isto significa que m é o *menor* pré-ponto fixo de f. Em particular $f(m) \sqsubseteq m$, o que, de novo pela monotonia de f, nos leva a $f(f(m)) \sqsubseteq f(m)$. Daqui concluimos que $f(m) \in X$ e, portanto, $m \sqsubseteq f(m)$. Mas então, $f(m) = m$ como queríamos. A segunda parte do teorema decorre desta: basta notar que se f é monótona em $(U, \sqsubseteq)$, então é ainda monótona no recticulado completo $(U, \sqsupseteq)$ formado pela ordem inversa. Se M é o menor ponto fixo de f em $(U, \sqsupseteq)$, será o *maior* ponto fixo da mesma função f em $(U, \sqsubseteq)$.

□

Lema 3

1. *A relação identidade* id *é uma bissimulação*
2. *A relação vazia $\perp$ é uma bissimulação.*
3. *O converso R° de uma bissimulação é-o igualmente*
4. *A composta $S \cdot R$ de duas bissimulações S e R é ainda uma bissimulação.*
5. *A união $\bigcup_{i \in I} R_i$ de uma família de bissimulações $\{R_i |\ i \in I\}$é uma bissimulação.*

Prova. Verifiquemos a afirmação 3. Para as restantes ver exercício 5.

$$\begin{array}{ll} & \langle y, x\rangle \in S^\circ \ \wedge\ y \xrightarrow{a} y' \\ \equiv & \quad \{\ \text{definição de composição relacional}\ \} \\ & \langle x, y\rangle \in S \ \wedge\ y \xrightarrow{a} y' \\ \Rightarrow & \quad \{\ S \text{ é bissimulação}\ \} \\ & \exists_{x'} .\ x \xrightarrow{a} x' \ \wedge\ \langle x', y'\rangle \in S \\ \equiv & \quad \{\ \text{definição de relação conversa}\ \} \\ & \exists_{x'} .\ x \xrightarrow{a} x' \ \wedge\ \langle y', x'\rangle \in S^\circ \end{array}$$

A prova completa-se por um cálculo análogo que se inicia pela hipótese $\langle y, x\rangle \in S^\circ \ \wedge\ x \xrightarrow{a} x'$.

□

Lema 4 *A relação de bissimilaridade é uma relação de equivalência.*

Prova.

- $p \sim q$ porque a identidade é bissimulação (lema 3).
- $p \sim q$ se $q \sim p$ porque o converso da bissimulação que testemunha $q \sim p$ é ainda uma bissimulação (de novo pelo lema 3).

- $p \sim r$ se $p \sim q$ e $q \sim r$ porque o par $\langle p, r \rangle$ pertence à composição das bisimulações que testemunham $p \sim q$ e $q \sim r$, composta essa que, ainda pelo 3, é ela própria uma bissimulação.

□

Lema 5 *O conjunto de todas as bissimulações entre dois sistemas de transição forma um* reticulado completo*, ordenado pela inclusão, cujo topo é a relação de* bissimilaridade $\sim$*.*

Prova. Exercício 6

□

Reflexão

Nesta lição introduzimos a noção de *sistema de transição etiquetado* como um *modelo* para os *sistemas reactivos*. Discutimos e caracterizamos as noções de *morfismo*, *simulação* e *bisimulação* a partir de dois pontos de vista: o da interpretação *relacional* e o da *coalgébrica*. Esta última faz a ponte com os *tipos coindutivos* [GH05], e, a partir destes, com os tipos indutivos e o método da programação funcional [BM97] anteriormente estudado em Cálculo de Programas. É, também, base para a modelação de sistemas reactivos *genéricos*, por alteração do functor que fixa a forma da coalgebra, *i.e.*,, a assinatura dos observadores [JR97, Rut00]. Não prosseguiremos, porém, aqui nesse caminho.

Questões

- Se temos um modelo semântico para os sistemas reactivos, com que *linguagem* os vamos descrever?
- Como estabelecer e provar as suas *propriedades*?

Referências

[BM97] R. Bird and O. Moor. *The Algebra of Programming*. Series in Computer Science. Prentice-Hall International, 1997.

[GH05] J. Gibbons and G. Hutton. Proof methods for corecursive programs. *Fundamenta Informatica*, 66(4):353–366, 2005.

[How91] Howie. *Automata and Languages*. Cambridge University Press, 1991.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata, Languages and Computation*. Addison-Wesley, 1979.

[JR97] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–159, 1997.

[Rut00] J. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. (Revised version of CWI Techn. Rep. CS-R9652, 1996).

[Tar55] A. Tarski. A lattice–theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

A Exercícios

Exercício 1

Conforme estudou, um *sistema de transição etiquetado* sobre um conjunto de acções $\mathcal{N}$ pode ser definido por uma função

$$\mathsf{next} : S \times \mathcal{N} \longrightarrow \mathcal{P}\mathcal{S}$$

onde S designa o conjunto de estados. Desta forma a notação $p \xrightarrow{a} p'$, vulgarmente usada para representar uma transição de estados, é definida por

$$p \xrightarrow{a} p' \quad \equiv \quad p' \in \mathsf{next}(p, a)$$

Recorde, ainda, que um morfismo entre sistemas de transição assim representados é uma função h entre os respectivos espaços de estados tal que o seguinte diagrama comuta:

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

1. Prove o lema 1, mostrando que um morfismo assim definido preserva e reflecte as transições do sistema original.
2. Confirme ou refute a seguinte afirmação: *a existência de um morfismo h entre os sistemas representados pelas funções next e next' é suficiente para mostrar que o primeiro é uma simulação do segundo.*
3. Confirme ou refute a seguinte afirmação: *dois estados u e v pertencentes ao espaço de estados dos sistemas representados pelas funções next e next', respectivamente, e tais que $u = h(v)$, sendo h um morfismo conforme definição acima, são bissimilares.*

Exercício 2

Dados dois sistemas de transição $\langle S_1, T_1 \rangle$ e $\langle S_2, T_2 \rangle$ sobre $\mathcal{N}$, diz-se que dois estados p e q são *mutuamente similares* sse

$$p \div q \;\equiv\; p \lesssim q \wedge q \lesssim p$$

1. Mostre que $\div$ é uma relação de equivalência.
2. Compare esta relação com a relação de bissimilaridade $\sim$ e com a noção canónica de equivalência entre autómatos.

Exercício 3

Um simulação é trivial se é vazia ou composta apenas por pares triviais (*i.e.*,, pares de estados a partir dos quais não existem transições) ou, ainda, se contém pelo menos um par trivial que não é acessível a partir de pelo menos um par não trivial contido em S. No sistema de transição seguinte

$$\{\langle 1, z, 2\rangle, \langle 1, x, 3\rangle, \langle 4, z, 5\rangle, \langle 6, z, 7\rangle, \langle 6, x, 8\rangle, \langle 6, x, 9\rangle\}$$

indique os pares triviais e enumere todas as simulações não triviais que podem nele ser definidas.

Exercício 4

Considere o sistema de transição caracterizado pela relação seguintes:

$$\{\langle 1, a, 2\rangle, \langle 1, a, 3\rangle, \langle 2, a, 3\rangle, \langle 2, b, 1\rangle, \langle 3, a, 3\rangle, \langle 3, b, 1\rangle, \langle 4, a, 5\rangle, \langle 5, a, 5\rangle, \langle 5, b, 6\rangle, \langle 6, a, 5\rangle, \langle 7, a, 8\rangle, \langle 8, a, 8\rangle, \langle 8, b, 7\rangle\}$$

Mostre ou refute a afirmação $1 \sim 4 \sim 6 \sim 7$.

Exercício 5

Complete a prova do lema 3.

Exercício 6

Prove o lema 5.

Exercício 7

Como estudou, as definições de *morfismo* entre sistemas de transição são distintas conforme se adopta a caracterização relacional clássica ou a coalgébrica. Recordando essas definições, mostre que

1. O grafo de um morfismo entre sistemas de transição na caracterização relacional é uma simulação
2. O grafo de um morfismo entre sistemas de transição na caracterização coalgébrica é uma bissimulação