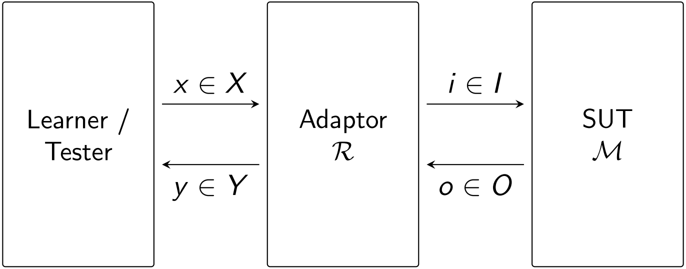
Action Codes

Frits Vaandrager and Thorsten Wißmann

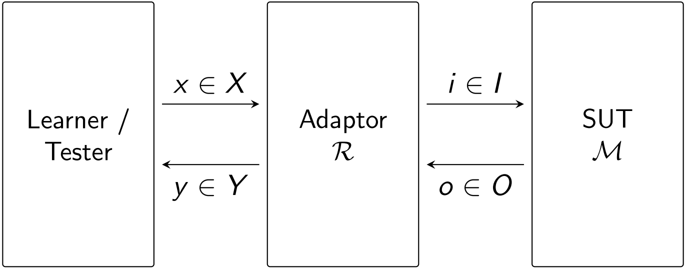
Radboud University Nijmegen

Oberseminar Informatik 8, FAU
May 2, 2023

Motivation: Adaptors used for Learning and Testing



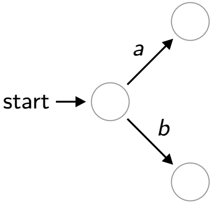
Motivation: Adaptors used for Learning and Testing



$\Rightarrow$ Generalization to:
Labelled Transition Systems for $A := I \times O$ and $B := X \times Y$.

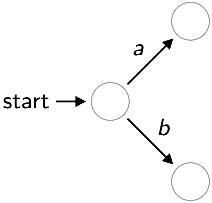
Labelled Transition Systems

Labeled transition systems (LTSs) constitute one of the most fundamental modeling mechanisms in Computer Science:



Labelled Transition Systems

Labeled transition systems (LTSs) constitute one of the most fundamental modeling mechanisms in Computer Science:

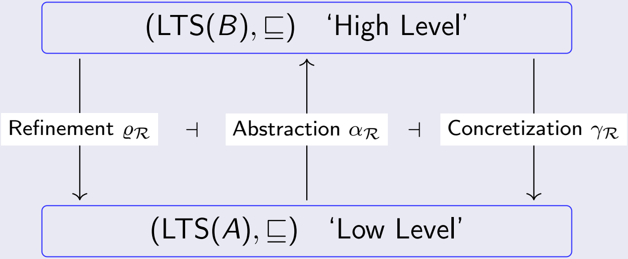


But our understanding of how to relate actions at different levels of abstraction is limited!

Overview

Goal

Find a notion of *action code* $\mathcal{R}$ from A to B, that translates between the LTSs living in different abstraction levels:



Simulation Relations

Definition

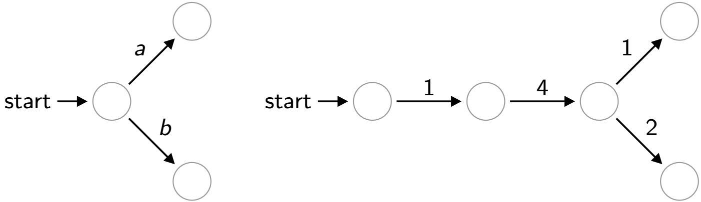
For $\mathcal{M}, \mathcal{N} \in \mathrm{LTS}(A)$, a *simulation* from $\mathcal{M}$ to $\mathcal{N}$ is a relation $S \subseteq Q^{\mathcal{M}} \times Q^{\mathcal{N}}$ such that

1. $q_0^{\mathcal{M}} \; S \; q_0^{\mathcal{N}}$ and
2. if $q_1 \; S \; q_2$ and $q_1 \xrightarrow{a}_{\mathcal{M}} q_1'$ then there exists a state q_2' such that $q_2 \xrightarrow{a}_{\mathcal{N}} q_2'$ and $q_1' \; S \; q_2'$.

We write $\mathcal{M} \sqsubseteq \mathcal{N}$ if there exists a simulation from $\mathcal{M}$ to $\mathcal{N}$.

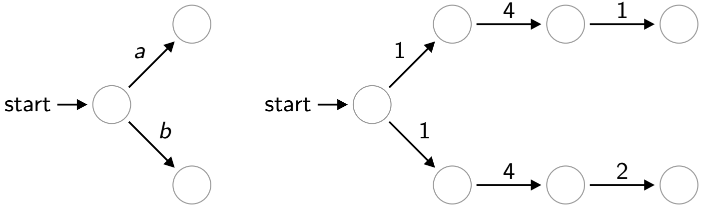
Changing the Level of Abstraction

Input a may be implemented by three consecutive inputs 1; 4; 1, and input b by 1; 4; 2 (ASCII encodings in octal format).

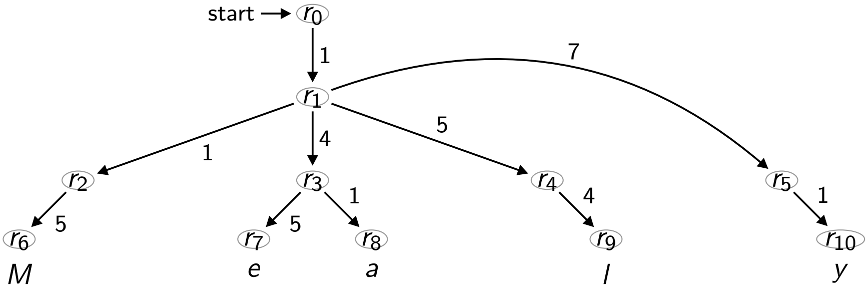


Action refinements?

An action refinement that replaces a by $1;\ 4;\ 1$ and b by $1;\ 4;\ 2$ leads to an incorrect refinement:



Our Solution: Action Codes



Formal Definition

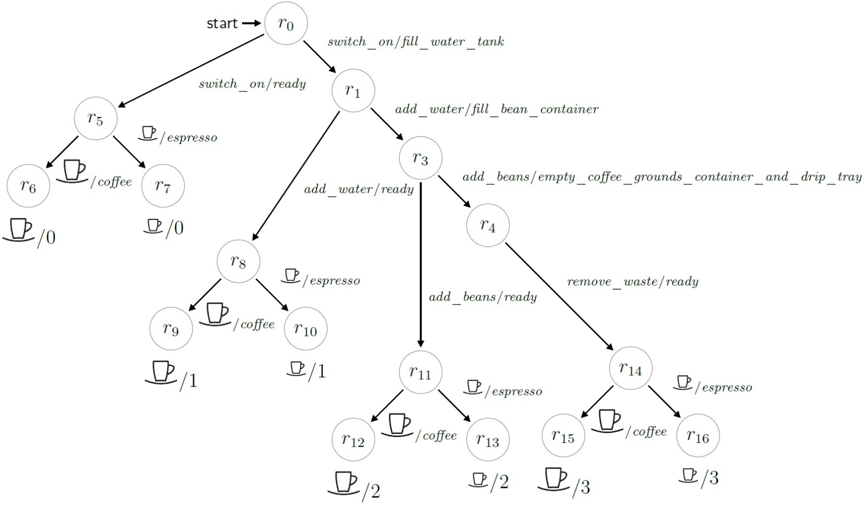
Definition (Action code)

For sets of action labels A and B, a (tree-shaped) action code $\mathcal{R}$ from A to B is a pair $\mathcal{R} = \langle \mathcal{M}, l \rangle$, with

1. $\mathcal{M} = \langle R, r_0, \longrightarrow \rangle \in \mathrm{LTS}(A)$ a deterministic, tree-shaped LTS
2. with L being the set of non-root leaves $L \subseteq R \setminus \{r_0\}$, and
3. an injective function $\ell \colon L \to B$.

$\mathrm{Code}(A, B)$ = all action codes from A to B.

Action Code for Coffee Machine



An Equivalent Formal Definition

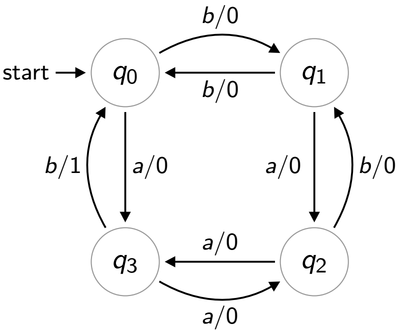
Definition

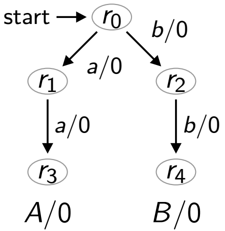
A (map-based) action code from A to B is a partial map $f\colon B \rightharpoonup A^+$ which is *prefix-free*, by which we mean that for all $b, b' \in \mathrm{dom}(f)$,

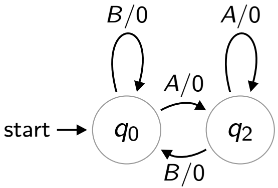
$$f(b) \leq f(b') \qquad \text{implies} \qquad b = b'. \tag{1}$$

$\leq$ is the 'prefix of' relation on words

Example of Contraction







Definition of Contraction

Definition

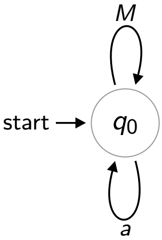
Suppose $\mathcal{M} = \langle Q, q_0, \longrightarrow \rangle \in \mathsf{LTS}(A)$ and an action code $\mathcal{R}\colon A \rightharpoonup B$. Then the contraction $\alpha_{\mathcal{R}}(\mathcal{M})$ is the LTS with states $Q^{\alpha(\mathcal{M})} \subseteq Q^{\mathcal{M}}$ defined inductively by the next two rules, for all $q, q' \in Q^{\mathcal{M}}$, $b \in B$:

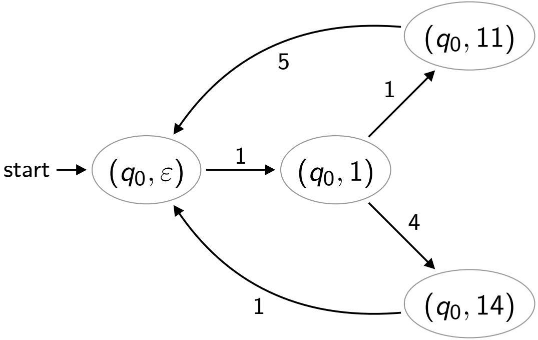
$$\frac{}{q_0 \in Q^{\alpha(\mathcal{M})}} \qquad (1_\alpha)$$

$$\frac{q \in Q^{\alpha(\mathcal{M})}, \quad b \in \mathrm{dom}(\mathcal{R}), \quad q \xrightarrow{\mathcal{R}(b)}_{\mathcal{M}} q'}{q \xrightarrow{b}_{\alpha(\mathcal{M})} q', \qquad q' \in Q^{\alpha(\mathcal{M})}} \qquad (2_\alpha)$$

The initial state $q_0^{\alpha(\mathcal{M})} := q_0^{\mathcal{M}}$ is the same as in $\mathcal{M}$.

Example of Refinement





Definition of Refinement

Definition

Let $\mathcal{R}\colon B \rightharpoonup A^+$ be an action code $\mathcal{R} \in \mathrm{Code}(A, B)$, we define the *refinement* operator $\varrho_{\mathcal{R}}\colon \mathrm{LTS}(B) \to \mathrm{LTS}(A)$ as follows: For $\mathcal{M} \in \mathrm{LTS}(B)$, the system $\varrho_{\mathcal{R}}(\mathcal{M}) \in \mathrm{LTS}(A)$ has a set of states

$$Q^{\varrho(\mathcal{M})} := \{(q, w) \in Q^{\mathcal{M}} \times A^* \mid w = \varepsilon \text{ or } \exists q \xrightarrow{b}_{\mathcal{M}} : w \lneqq \mathcal{R}(b)\}$$

and the initial state $(q_0^{\mathcal{M}}, \varepsilon)$. The transition relation $\longrightarrow_{\varrho(\mathcal{M})}$ is defined by the following rules:

$$\frac{(q, wa) \in Q^{\varrho(\mathcal{M})}}{(q, w) \xrightarrow{a}_{\varrho(\mathcal{M})} (q, wa)} \qquad (1_\varrho)$$

$$\frac{q \xrightarrow{b}_{\mathcal{M}} q' \quad wa = \mathcal{R}(b)}{(q, w) \xrightarrow{a}_{\varrho(\mathcal{M})} (q', \varepsilon)} \qquad (2_\varrho)$$

Galois Connection

Theorem (Galois connection)

Consider an action code $\mathcal{R} \in \mathrm{Code}(A, B)$ *and LTSs* $\mathcal{N} \in \mathrm{LTS}(A)$ *and* $\mathcal{M} \in \mathrm{LTS}(B)$*:*

1. *If* $\mathrm{dom}(\mathcal{R}) = B$*, then* $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M}$ *implies* $\mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})$*.*
2. *If* $\mathcal{M}$ *is deterministic, then* $\mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})$ *implies* $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M}$*.*

Action code composition

Definition

For codes $\mathcal{R} \in B \rightharpoonup A^+$ and $\mathcal{S} \in C \rightharpoonup B^+$:

$$(\mathcal{R} * \mathcal{S})\colon C \rightharpoonup A^+ \text{ as the Kleisli composition}$$

Action code composition

Definition

For codes $\mathcal{R} \in B \rightharpoonup A^+$ and $\mathcal{S} \in C \rightharpoonup B^+$:

$$(\mathcal{R} * \mathcal{S})\colon C \rightharpoonup A^+ \text{ as the Kleisli composition}$$

Theorem

1. contraction (α) commutes with code composition
2. refinemenet (ϱ) commutes with code composition if $im(\mathcal{S}) \subseteq dom(\mathcal{R})^+$

concretization γ does not commute with $*$ in general.

Mealy Machines

Definition

For a non-empty sets of inputs I and outputs O, a (non-deterministic) Mealy machine $\mathcal{M} \in \mathrm{LTS}(I \times O)$ is an LTS where the labels are pairs of an input and an output.

Winning Action Codes

Definition (Winning)

Let $\mathcal{R} \in \text{Code}(I \times O, X \times Y)$ be an action code and let $x \in X$. Then

1. A leaf $r \in R$ is winning for x if $l(r) = (x, y)$, for some $y \in Y$.
2. An internal state $r \in R$ is winning for x with input $i \in I$ if $r \xrightarrow{i}$ and, for each transition of the form $r \xrightarrow{i/o} r'$, r' is winning for x.
3. An internal state $r \in R$ is winning for $x \in X$ if it is winning for x with some input $i \in I$.
4. $\mathcal{R}$ has a winning strategy if r^0 is winning for all leaf labels.

From a Winning Action Code to an Adaptor

function Adaptor($\mathcal{R}$)
    while *true*
        $x \leftarrow$ `Receive-from-learner`()
        $r \leftarrow r^0$
        while r is internal   ▷ We maintain loop invariant that r is winning for x
            $i \leftarrow$ input such that r is winning for x with i
            `Send-to-SUT`(i)
            $o \leftarrow$ `Receive-from-SUT`()
            $r \leftarrow$ state reached by i/o-transition from r
        end while
        $(x, y) \leftarrow I(r)$
        `Send-to-learner`(y)
    end while
end function

Conclusions

1. A notion of code that relates abstract actions related to low-level models in which these actions are refined by sequences of concrete actions.
2. This may help with the systematic design of adaptors during learning and testing, and the subsequent interpretation of obtained results.
3. Almost all results, examples, and counter-examples formalized in Coq (except Adaptor-Pseudocode)
4. Paper will also present concretization operator and corresponding Galois connection.