Action Codes

Frits Vaandrager
sws.cs.ru.nl

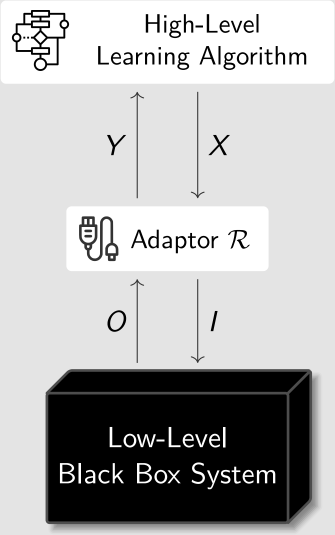
Thorsten Wißmann
thorsten-wissmann.de

Radboud University
Nijmegen, the Netherlands

Friedrich-Alexander-Universität
Erlangen-Nürnberg, Germany

ICALP 2023, Paderborn, July 13, 2023

Motivation: Adaptors for Automata Learning & Testing

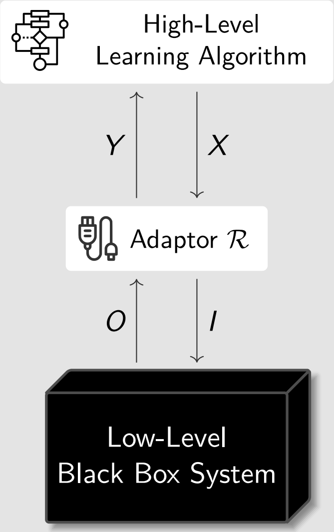


Used for example in papers learning...

USB/Smartcard readers, (D)TLS, SSH, Wi-Fi, ...

$\Longrightarrow$ Smaller Models!

Motivation: Adaptors for Automata Learning & Testing



Used for example in papers learning...

USB/Smartcard readers, (D)TLS, SSH, Wi-Fi, ...
$\Longrightarrow$ Smaller Models!

Goal: Formal theory for adaptors for Mealy machines

$\Longrightarrow$ Action Codes

$\Longrightarrow$ Works for general (sometimes deterministic) LTSs!

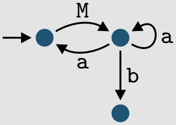
Contributions

- Formal definition of *Action Code*
- Abstraction operator with left and right adjoint
- Works for general LTSs ...
  & deterministic systems
  } best of both worlds!
- Compositionality: multiple layers of abstraction
- Formal Proofs in Coq
  `tinyurl.com/icalp23`
  ← In the paper, click on the icon to see the coq formalization!
- Pseudo-Code for the adaptor implementing an action code

Definition: labelled transition systems LTS(A)

Tuple $\mathcal{M} = (Q, q_0, \to)$ consisting of

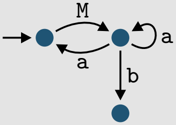
- a set Q of states
- an initial state $q_0 \in Q$
- a transition relation $\to \subseteq Q \times A \times Q$



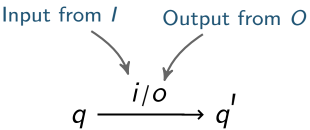
Definition: labelled transition systems LTS(A)

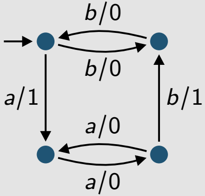
Tuple $\mathcal{M} = (Q, q_0, \to)$ consisting of

- a set Q of states
- an initial state $q_0 \in Q$
- a transition relation $\to \subseteq Q \times A \times Q$



Mealy machines = LTSs for $A := I \times O$

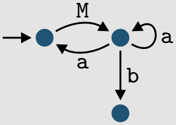




Definition: labelled transition systems $\mathrm{LTS}(A)$

Tuple $\mathcal{M} = (Q, q_0, \to)$ consisting of

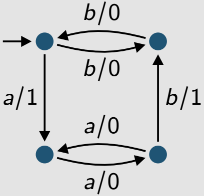
- a set Q of states
- an initial state $q_0 \in Q$
- a transition relation $\to \subseteq Q \times A \times Q$



Mealy machines $=$ LTSs for $A := I \times O$

Input from I — Output from O

$$q \xrightarrow{i/o} q'$$



Definition: Simulation $\mathcal{M} \sqsubseteq \mathcal{N}$ in $\mathrm{LTS}(A)$

Relation $S \subseteq Q^{\mathcal{M}} \times Q^{\mathcal{N}}$ such that $(q_0^{\mathcal{M}}, q_0^{\mathcal{N}}) \in S$ and

$$(q, p) \in S \ \&\ q \xrightarrow{a} q' \text{ in } \mathcal{M} \implies \exists\, p \xrightarrow{a} p' \text{ in } \mathcal{N} \text{ with } (q', p') \in S$$

$\Rightarrow$ How to compare LTSs of different alphabets A, B?

Central Definition

A

low-level symbols

— Code $\mathcal{R}$ —

B

high-level symbols

$$a_1\,a_2\,a_3\cdots a_n \triangleq b$$
$$a'_1\cdots a'_k \triangleq b'$$
$$\bar{a}_1\,\bar{a}_2\cdots \bar{a}'_\ell \triangleq \bar{b}$$

Central Definition

A
low-level symbols

Code $\mathcal{R}$

B
high-level symbols

$$\begin{aligned} a_1 a_2 a_3 \cdots a_n &\mathrel{\hat{=}} b \\ a'_1 \cdots a'_k &\mathrel{\hat{=}} b' \\ \bar{a}_1 \bar{a}_2 \cdots \bar{a}'_\ell &\mathrel{\hat{=}} \bar{b} \end{aligned}$$

Definition: Action code $\mathcal{R}$ from A to B

Partial map $\mathcal{R}\colon B \rightharpoonup A^+$ which is prefix-free, i.e.

$$\mathcal{R}(b_1) \le \mathcal{R}(b_2) \implies b_1 = b_2 \qquad \forall\, b_1, b_2 \in B \qquad (p)$$

'prefix of' relation on words

Central Definition

A — low-level symbols — Code $\mathcal{R}$ — **B** — high-level symbols

$$a_1 a_2 a_3 \cdots a_n \mathrel{\hat{=}} b$$
$$a'_1 \cdots a'_k \mathrel{\hat{=}} b'$$
$$\bar{a}_1 \bar{a}_2 \cdots \bar{a}'_\ell \mathrel{\hat{=}} \bar{b}$$

Definition: Action code $\mathcal{R}$ from A to B

Partial map $\mathcal{R}\colon B \rightharpoonup A^+$ which is prefix-free, i.e.

$$\mathcal{R}(b_1) \le \mathcal{R}(b_2) \implies b_1 = b_2 \qquad \forall\, b_1, b_2 \in B \qquad (p)$$

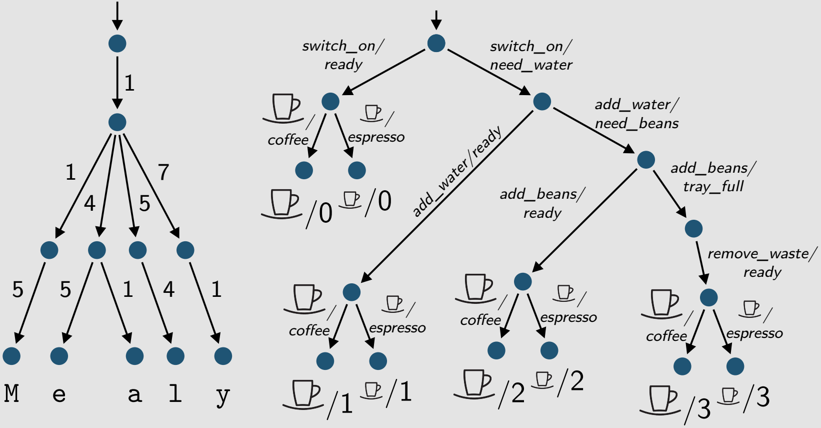
'prefix of' relation on words

Reasonable properties for deterministic systems:

- $\rightharpoonup$: at most one low-level representation
- A^+: low-level representation is non-empty
- (p) prefix-freeness allows deterministic decoding

Theorem: Characterization

Action codes as prefix-free $\mathcal{R}\colon B \rightharpoonup A^+$ $\Longleftrightarrow$ tree-shaped deterministic LTS(A) with leaves labelled in B

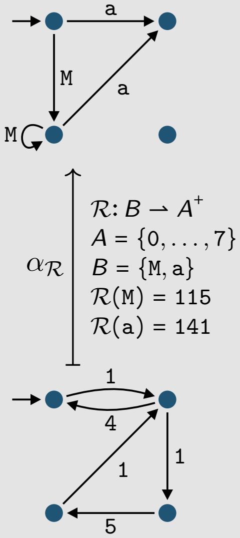


Contraction/Abstraction in LTSs

Definition: abstraction $\alpha_{\mathcal{R}}\colon \mathsf{LTS}(A) \to \mathsf{LTS}(B)$

For an action code $\mathcal{R}\colon B \rightharpoonup A^+$:

$$\text{Transition } q \xrightarrow{b} q' \text{ in } \alpha_{\mathcal{R}}(\mathcal{M})$$
$$\Longleftrightarrow$$
$$\text{Run } q \to \overset{\mathcal{R}(b)}{\cdots} \to q' \text{ in } \mathcal{M}$$



Contraction/Abstraction in LTSs

Definition: abstraction $\alpha_{\mathcal{R}}\colon \mathrm{LTS}(A) \to \mathrm{LTS}(B)$

For an action code $\mathcal{R}\colon B \rightharpoonup A^+$:

$$\text{Transition } q \xrightarrow{b} q' \text{ in } \alpha_{\mathcal{R}}(\mathcal{M})$$

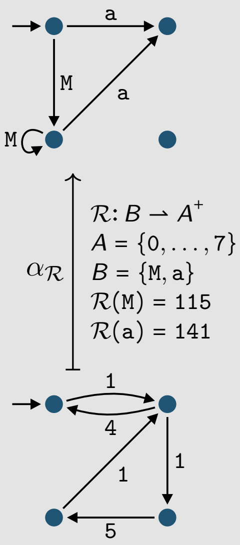
$$\Longleftrightarrow$$

$$\text{Run } q \to \overset{\mathcal{R}(b)}{\cdots} \to q' \text{ in } \mathcal{M}$$

Theorem

The contraction operator $\alpha_{\mathcal{R}}\colon \mathrm{LTS}(A) \to \mathrm{LTS}(B)$

- preserves determinism
- is monotone w.r.t. simulation order $\sqsubseteq$



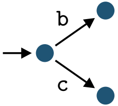
Construction in the other way?

Trivial refinement: replace every transition $q \xrightarrow{b} q'$ in $\mathcal{N} \in \mathrm{LTS}(B)$ with a run $q \rightarrow \overset{\mathcal{R}(b)}{\cdots} \rightarrow q'$.

Construction in the other way?

Trivial refinement: replace every transition $q \xrightarrow{b} q'$ in $\mathcal{N} \in \mathsf{LTS}(B)$ with a run $q \to \overset{\mathcal{R}(b)}{\cdots} \to q'$.

Construction $\mathsf{LTS}(B) \to \mathsf{LTS}(A)$ for $A = \{0, \ldots, 7\}$ and $B = \{\mathtt{b}, \mathtt{c}\}$

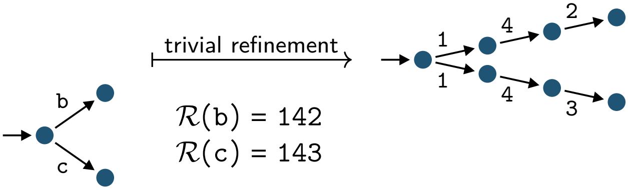


$\mathcal{R}(\mathtt{b}) = 142$
$\mathcal{R}(\mathtt{c}) = 143$

Construction in the other way?

Trivial refinement: replace every transition $q \xrightarrow{b} q'$ in $\mathcal{N} \in \mathsf{LTS}(B)$ with a run $q \to \overset{\mathcal{R}(b)}{\cdots} \to q'$.

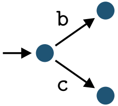
Construction $\mathsf{LTS}(B) \to \mathsf{LTS}(A)$ for $A = \{0, \ldots, 7\}$ and $B = \{\mathrm{b}, \mathrm{c}\}$



Construction in the other way?

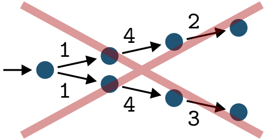
Trivial refinement: replace every transition $q \xrightarrow{b} q'$ in $\mathcal{N} \in \mathrm{LTS}(B)$ with a run $q \to \overset{\mathcal{R}(b)}{\cdots} \to q'$.

Construction $\mathrm{LTS}(B) \to \mathrm{LTS}(A)$ for $A = \{0, \ldots, 7\}$ and $B = \{\mathtt{b}, \mathtt{c}\}$



$\xmapsto{\text{trivial refinement}}$

$\mathcal{R}(\mathtt{b}) = 142$
$\mathcal{R}(\mathtt{c}) = 143$

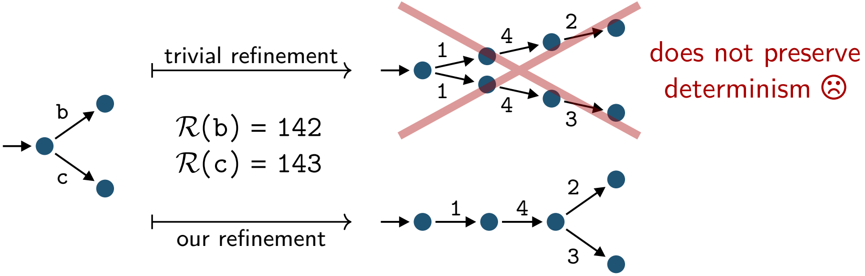


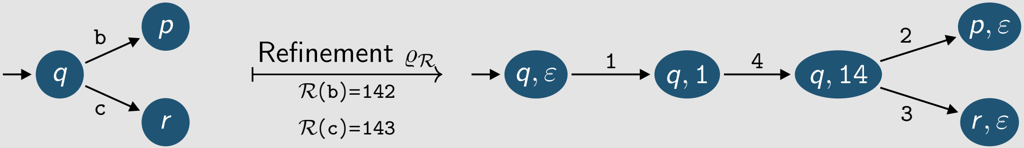
does not preserve determinism ☹

Construction in the other way?

Trivial refinement: replace every transition $q \xrightarrow{b} q'$ in $\mathcal{N} \in \mathrm{LTS}(B)$ with a run $q \to \overset{\mathcal{R}(b)}{\cdots} \to q'$.

Construction $\mathrm{LTS}(B) \to \mathrm{LTS}(A)$ for $A = \{0,\ldots,7\}$ and $B = \{\mathtt{b}, \mathtt{c}\}$





Definition: $\varrho_{\mathcal{R}} \colon \mathsf{LTS}(B) \to \mathsf{LTS}(A)$

For $\mathcal{N} = (Q, q_0, \to)$ in $\mathsf{LTS}(B)$, define $\varrho_{\mathcal{R}}(\mathcal{N})$ by:

- States: $\bar{Q} := \{(q, w) \in Q \times A^* \mid w = \varepsilon \text{ or } (q \xrightarrow{b} \text{ and } w \le \mathcal{R}(b))\}$
- Initial: (q_0, ε)
- Transitions:
  $\{(q, w) \xrightarrow{a} (q, wa) \mid (q, wa) \in \bar{Q}\} \cup \{(q, w) \xrightarrow{a} (q', \varepsilon) \mid q \xrightarrow{b} q', \mathcal{R}(b) = wa\}$

Theorem: for $\mathcal{R} \colon B \rightharpoonup A^*$, refinement $\varrho_{\mathcal{R}} \colon \mathsf{LTS}(B) \to \mathsf{LTS}(A)$

- preserves determinism
- is monotone w.r.t. simulation order $\sqsubseteq$

High-level $\mathcal{N} \in \mathrm{LTS}(B)$
and code $\mathcal{R}\colon B \rightharpoonup A^+$

Refinement $\varrho_{\mathcal{R}}(\mathcal{N}) \in \mathrm{LTS}(A)$

Add as few transitions as necessary
for representing $\mathcal{N}$'s transitions

High-level $\mathcal{N} \in \mathsf{LTS}(B)$ and code $\mathcal{R}\colon B \rightharpoonup A^+$

Refinement $\varrho_{\mathcal{R}}(\mathcal{N}) \in \mathsf{LTS}(A)$

Add as few transitions as necessary for representing $\mathcal{N}$'s transitions

Theorem: Galois connection $\varrho \dashv \alpha$

If $\mathcal{R}\colon B \rightharpoonup A^+$ is defined for all actions in $\mathcal{N} \in \mathsf{LTS}(B)$ and $\mathcal{M} \in \mathsf{LTS}(A)$ is deterministic, then:

$$\underbrace{\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M}}_{\text{in } \mathsf{LTS}(A)} \iff \underbrace{\mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})}_{\text{in } \mathsf{LTS}(B)}$$

High-level $\mathcal{N} \in \mathrm{LTS}(B)$
and code $\mathcal{R}\colon B \rightharpoonup A^+$

Refinement $\varrho_\mathcal{R}(\mathcal{N}) \in \mathrm{LTS}(A)$

Add as few transitions as necessary
for representing $\mathcal{N}$'s transitions

$\sqsubseteq$

Concretization $\gamma_\mathcal{R}(\mathcal{N}) \in \mathrm{LTS}(A)$

Accumulate prefix $\in A^*$ while in $\mathcal{R}$,
otherwise go to sink-state

Theorem: Galois connection $\varrho \dashv \alpha$

If $\mathcal{R}\colon B \rightharpoonup A^+$ is defined for all actions in $\mathcal{N} \in \mathrm{LTS}(B)$ and $\mathcal{M} \in \mathrm{LTS}(A)$ is deterministic, then:

$$\underbrace{\varrho_\mathcal{R}(\mathcal{N}) \sqsubseteq \mathcal{M}}_{\text{in } \mathrm{LTS}(A)} \iff \underbrace{\mathcal{N} \sqsubseteq \alpha_\mathcal{R}(\mathcal{M})}_{\text{in } \mathrm{LTS}(B)}$$

High-level $\mathcal{N} \in \mathsf{LTS}(B)$ and code $\mathcal{R}\colon B \rightharpoonup A^+$

Refinement $\varrho_{\mathcal{R}}(\mathcal{N}) \in \mathsf{LTS}(A)$

Add as few transitions as necessary for representing $\mathcal{N}$'s transitions

$\sqsubseteq$

Concretization $\gamma_{\mathcal{R}}(\mathcal{N}) \in \mathsf{LTS}(A)$

Accumulate prefix $\in A^*$ while in $\mathcal{R}$, otherwise go to sink-state

Theorem: Galois connection $\varrho \dashv \alpha$

If $\mathcal{R}\colon B \rightharpoonup A^+$ is defined for all actions in $\mathcal{N} \in \mathsf{LTS}(B)$ and $\mathcal{M} \in \mathsf{LTS}(A)$ is deterministic, then:

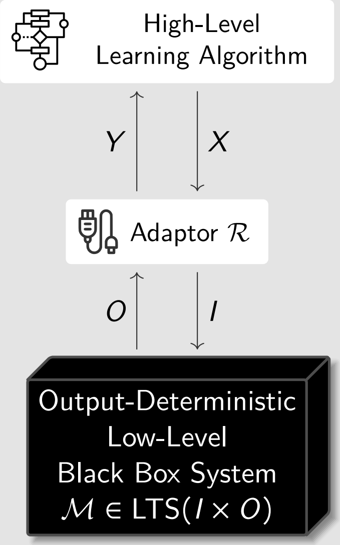
$$\underbrace{\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M}}_{\text{in } \mathsf{LTS}(A)} \iff \underbrace{\mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})}_{\text{in } \mathsf{LTS}(B)}$$

Theorem: Galois connection $\alpha \dashv \gamma$

For all $\mathcal{N} \in \mathsf{LTS}(B)$, $\mathcal{M} \in \mathsf{LTS}(A)$:

$$\underbrace{\alpha_{\mathcal{R}}(\mathcal{M}) \sqsubseteq \mathcal{N}}_{\text{in } \mathsf{LTS}(B)} \iff \underbrace{\mathcal{M} \sqsubseteq \gamma_{\mathcal{R}}(\mathcal{N})}_{\text{in } \mathsf{LTS}(A)}$$

(+ versions for various degrees of determinism)



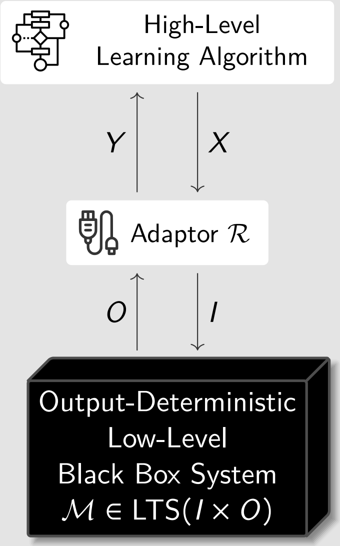
For deterministic Mealy machines:

$$\mathcal{R}\colon \underbrace{X \times Y}_{B} \rightharpoonup (\underbrace{I \times O}_{A})^{+}$$

Theorem: Adaptor Algorithm

If $\mathcal{R}$ determinate and has a strategy for every input X, then:

$$\text{Adaptor behaves like } \alpha_{\mathcal{R}}(\mathcal{M})$$



For deterministic Mealy machines:

$$\mathcal{R}\colon \underbrace{X \times Y}_{B} \rightharpoonup (\underbrace{I \times O}_{A})^{+}$$

Theorem: Adaptor Algorithm

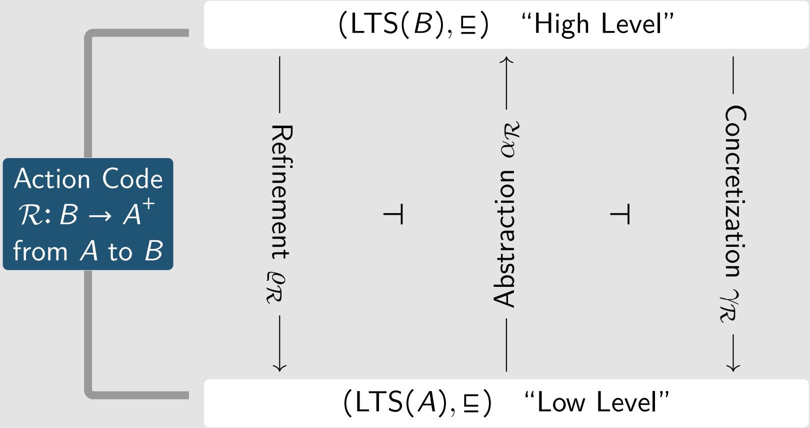
If $\mathcal{R}$ determinate and has a strategy for every input X, then:

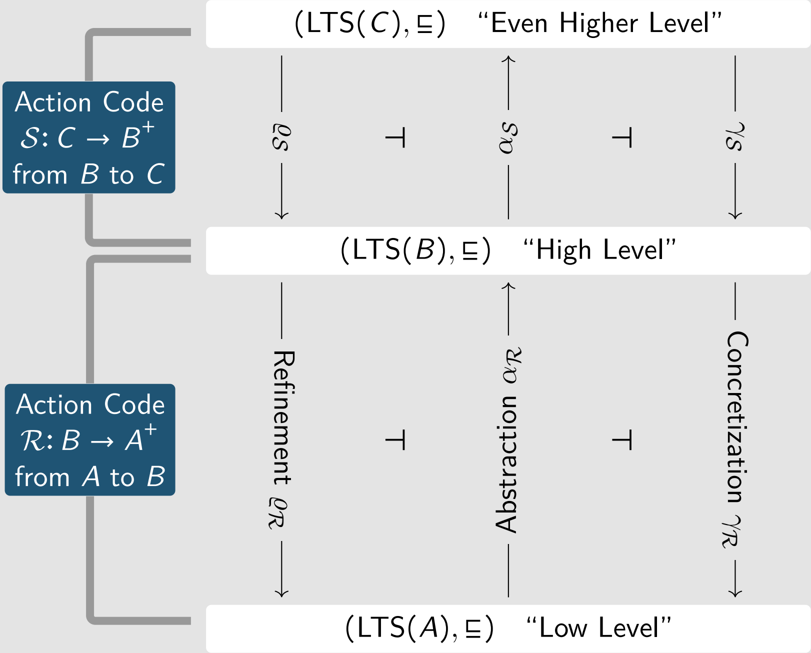
$$\text{Adaptor behaves like } \alpha_{\mathcal{R}}(\mathcal{M})$$

Recipe: Learning

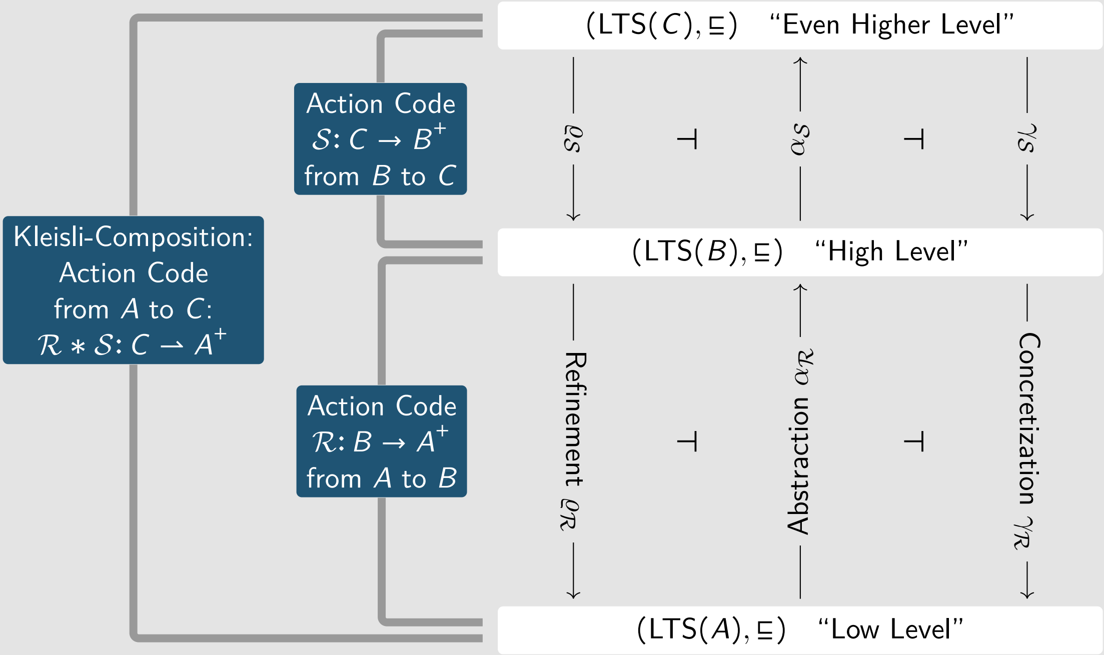
- $\Longrightarrow$ Apply learning algorithm to adaptor's interface
- $\Longrightarrow$ Yields a high-level model $\mathcal{N} = \alpha_{\mathcal{R}}(\mathcal{M})$ $\in \mathrm{LTS}(X \times Y)$
- $\Longrightarrow$ Bounds for the unknown $\mathcal{M}$:

$$\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M} \sqsubseteq \gamma_{\mathcal{R}}(\mathcal{N})$$

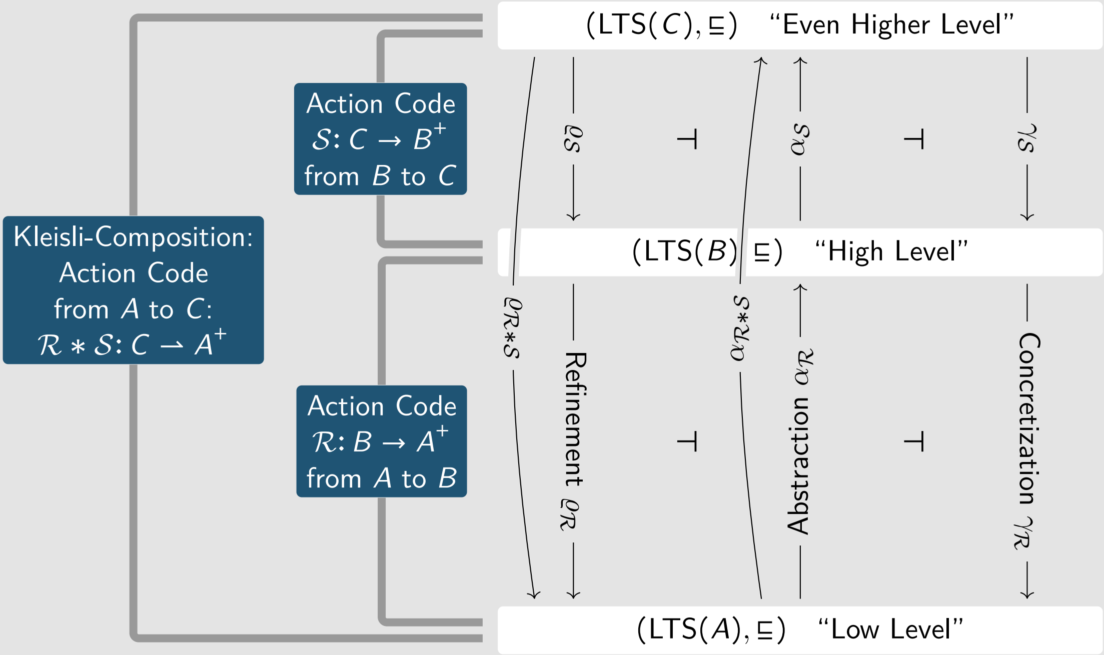

Action Code
$\mathcal{R}\colon B \to A^+$
from A to B
$(\mathsf{LTS}(B), \sqsubseteq)$ “High Level”
Refinement $\varrho_\mathcal{R}$
$\dashv$
Abstraction $\alpha_\mathcal{R}$
$\dashv$
Concretization $\gamma_\mathcal{R}$
$(\mathsf{LTS}(A), \sqsubseteq)$ “Low Level”



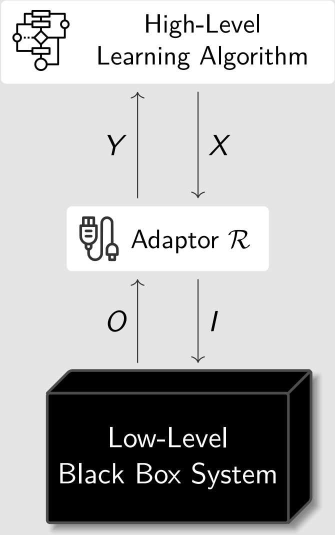
(LTS(C), ⊑) “Even Higher Level”
(LTS(B), ⊑) “High Level”
(LTS(A), ⊑) “Low Level”
Action Code
S: C → B+
from B to C
Action Code
R: B → A+
from A to B
ϱS
αS
γS
⊣
⊣
Refinement ϱR
Abstraction αR
Concretization γR
⊣
⊣



(LTS(C), ⊑) "Even Higher Level"
Action Code
$\mathcal{S}\colon C \to B^+$
from B to C
$\varrho_\mathcal{S}$
⊣
$\alpha_\mathcal{S}$
⊣
$\gamma_\mathcal{S}$
Kleisli-Composition:
Action Code
from A to C:
$\mathcal{R} * \mathcal{S}\colon C \to A^+$
(LTS(B), ⊑) "High Level"
Action Code
$\mathcal{R}\colon B \to A^+$
from A to B
Refinement $\varrho_\mathcal{R}$
⊣
Abstraction $\alpha_\mathcal{R}$
⊣
Concretization $\gamma_\mathcal{R}$
(LTS(A), ⊑) "Low Level"


(LTS(C), ⊑) "Even Higher Level"
Action Code
$\mathcal{S}\colon C \to B^+$
from B to C
$\varrho_\mathcal{S}$
⊣
$\alpha_\mathcal{S}$
⊣
$\gamma_\mathcal{S}$
Kleisli-Composition:
Action Code
from A to C:
$\mathcal{R} * \mathcal{S}\colon C \to A^+$
(LTS(B), ⊑) "High Level"
$\varrho_{\mathcal{R}*\mathcal{S}}$
$\alpha_{\mathcal{R}*\mathcal{S}}$
Refinement $\varrho_\mathcal{R}$
Abstraction $\alpha_\mathcal{R}$
Concretization $\gamma_\mathcal{R}$
Action Code
$\mathcal{R}\colon B \to A^+$
from A to B
(LTS(A), ⊑) "Low Level"

Conclusions



- Formal definition of *Action Code*
- Adjoint Operators for upper / lower bounds in learning
- Works for general LTSs …
  & deterministic systems
  } best of both worlds!
- Compositionality
- Formal Proofs in Coq `tinyurl.com/icalp23`

$\Longrightarrow$ Apply it in automata learning & testing!

Questions? Comments? Suggestions?

Vaandrager, Frits, Thorsten Wißmann. "Action Codes". *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Ed. by Kousha Etessami, Uriel Feige, Gabriele Puppis. Vol. 261. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, July 2023, 137:1–137:20. DOI: 10.4230/LIPIcs.ICALP.2023.137.

Vaandrager, Frits, Thorsten Wißmann. *Action Codes – Coq Formalization*. 2023. URL: https://gitlab.science.ru.nl/twissmann/action-codes-coq.