The Complexity of Transducer Synthesis from Multi-Sequential Specifications

Léo Exibard[1,2] Emmanuel Filiot[1,3] Ismaël Jecker[1]

Wednesday, December 5th, 2018

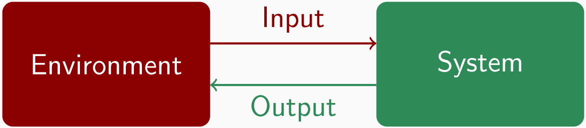
[1]Université libre de Bruxelles

[2]LIS
Aix-Marseille Université

[3]FNRS

From verification to synthesis

Reactive systems



Interaction $\rightsquigarrow$ $i_1 o_1 i_2 o_2 i_3 o_3 \cdots \in (I\,O)^\omega$ or $(I\,O)^*$

Verification

Check that a system satisfies a specification

System $\parallel$ Env $\models$ Specification

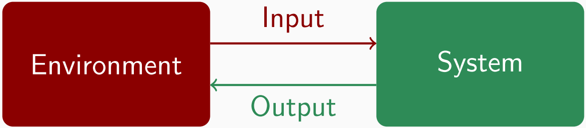
Synthesis

Generate a system from a specification

? $\parallel$ Env $\models$ Specification

From verification to synthesis

Reactive systems



Interaction $\rightsquigarrow$ $i_1 o_1 i_2 o_2 i_3 o_3 \cdots \in (I\,O)^\omega$ or $(I\,O)^*$

Verification

Check that a system satisfies a specification

System $\|$ Env $\models$ Specification

Synthesis

Generate a system from a specification

? $\|$ Env $\models$ Specification

$$? \parallel \text{Environment} \models \text{Specification}$$

➔ Generate a system from a specification

Implementing a specification

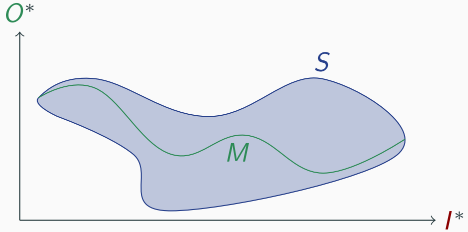
Input words I^*

Output words O^*

Implementation $M : I^* \to O^*$

Specification $S \subseteq I^* \times O^*$

M fulfils S, written $M \models S$, if for all $in \in \mathrm{dom}(S), (in, M(in)) \in S$



The realisability and synthesis problem

$\mathcal{S}$ = Class of specifications

$S \subseteq I^* \times O^*$

$\mathcal{M}$ = Class of target implementations

$M : I^* \to O^*$

Synthesis problem from $\mathcal{S}$ to $\mathcal{M}$

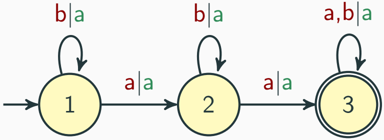
Input: Specification $S \in \mathcal{S}$

Output:
- Implementation $M \in \mathcal{M}$ s.t. $M \models S$ if it exists
- **No** otherwise

Realisability problem from $\mathcal{S}$ to $\mathcal{M}$

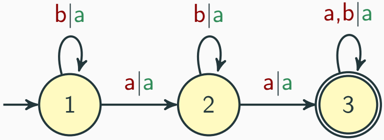
→ Corresponding decision problem

Finite transducers: automata with outputs

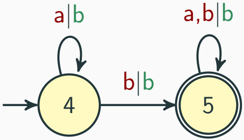


Replace every letter with an a when there are at least two a's

Finite transducers: automata with outputs

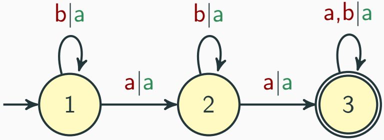


Replace every letter with an a when there are at least two a's

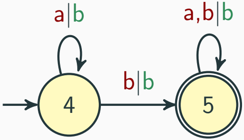


Replace every letter with a b when there is at least one b

Finite transducers: automata with outputs



Replace every letter with an a when there are at least two a's

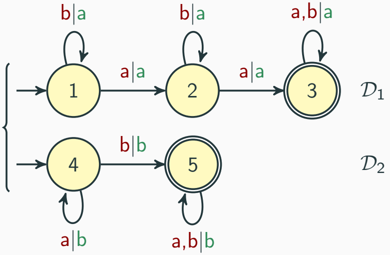


Replace every letter with a b when there is at least one b

Sequential transducer

The transition and output letter are *determined* by the input letter

Multi-sequential transducers



Running example

Multi-sequential transducer

Union of sequential transducers

$$\mathcal{T} = \biguplus_{i=1}^{k} \mathcal{D}_i$$

Multi-sequentiality

A relation is *multi-sequential* if it can be defined by a multi-sequential transducer

- Decidable for functions [Choffrut and Schützenberger, 1986]
- Membership in **PTime** [Jecker and Filiot, 2015]

Transducer realisability problem
Known results

$\mathcal{M}$ = sequential transducers

$\mathcal{S}$	**Complexity**
MSO	Nonelementary [Büchi and Landweber, 1969]
LTL	2-ExpTime-c [Pnueli and Rosner, 1989]
Finite Transducers	ExpTime-c

Transducer realisability problem
Known results

$\mathcal{M}$ = sequential transducers

$\mathcal{S}$	**Complexity**
MSO	Nonelementary [Büchi and Landweber, 1969]
LTL	2-ExpTime-c [Pnueli and Rosner, 1989]
Finite Transducers	ExpTime-c

Question: Class of transducers with better complexity?

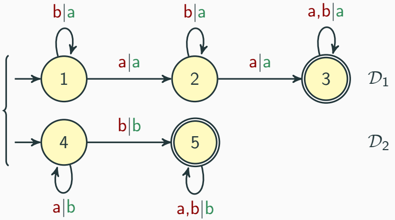
Realisability of multi-sequential specifications

$\mathcal{S}$ = Multi-seq. transducers

Unions of sequential transducers $\mathcal{T} = \biguplus_{i=1}^{k} \mathcal{D}_i$

$\mathcal{M}$ = Seq. transducers

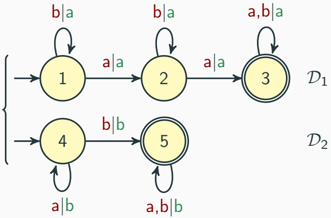
Output letter and transition is *determined* by input letter



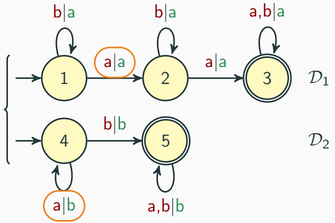
Theorem

Sequential transducer synthesis from multi-sequential specifications is **PSpace**-complete.

Realisability of multi-sequential specifications

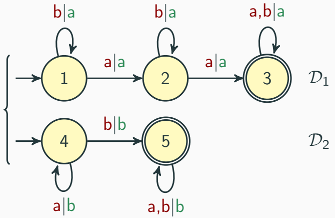


Realisability of multi-sequential specifications



➔ On input a, need to *drop* one transducer

Realisability of multi-sequential specifications

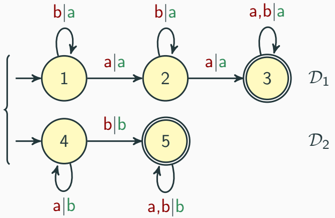


➔ On input a, need to *drop* one transducer

Critical prefix *u*

At least two runs on *u* disagree on their output

Realisability of multi-sequential specifications



➔ On input a, need to *drop* one transducer

Critical prefix u

At least two runs on u disagree on their output

Residual property

For all critical prefix u, there exists $P \subsetneq \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ s.t.:

1. All transducers in P produce the same output on u
2. The domain is still covered: $u^{-1}\mathrm{dom}(\mathcal{T}) = \bigcup_{i \in P} u^{-1}\mathrm{dom}(\mathcal{D}_i)$
3. The residual specification $u^{-1} \left[\!\left[\biguplus_{i \in P} \mathcal{D}_i \right]\!\right]$ is realisable

Realisability of multi-sequential specifications

Theorem

Sequential transducer realisability from multi-sequential specifications is **PSpace**-complete.

Easiness

The *residual property* can be checked in **PSpace**.

Realisability of multi-sequential specifications

Theorem

Sequential transducer realisability from multi-sequential specifications is **PSpace**-complete.

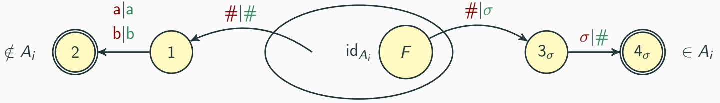
Easiness

The *residual property* can be checked in **PSpace**.

Hardness

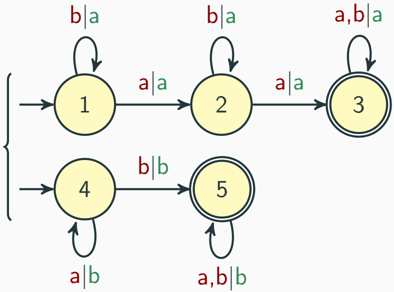
$\rightsquigarrow$ Emptiness problem of the intersection of n DFAs

$$S: \quad w\#\sigma \mapsto w\sigma\# \quad \text{if } \exists i, w \in L(A_i) \qquad (\sigma \in \{a,b\})$$
$$\quad w\#\sigma \mapsto w\#\sigma \quad \text{if } \exists i, w \notin L(A_i)$$



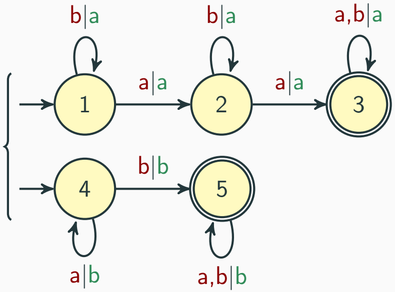
2-sequential transducer for one A_i

Asynchronicity



Our running example

Asynchronicity

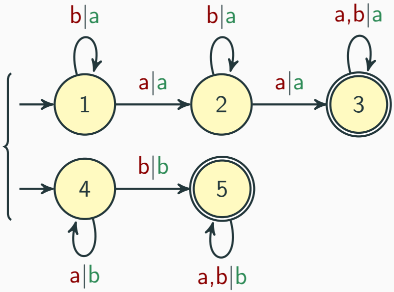


Our running example

Waiting two steps allows to determine whether:

- There is at least one b
- There are at least two a's

Asynchronicity



Our running example

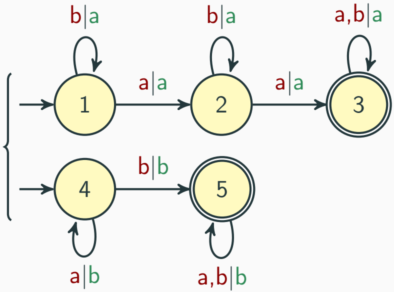
Asynchronous transducer

At every transition, reads a letter, outputs a (possibly empty) word.

Waiting two steps allows to determine whether:

- There is at least one b
- There are at least two a's

Asynchronicity



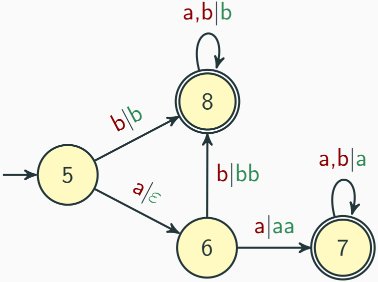
Our running example

Asynchronous transducer

At every transition, reads a letter, outputs a (possibly empty) word.

Waiting two steps allows to determine whether:

- There is at least one b
- There are at least two a's



An asynchronous implementation

Asynchronous transducer realisability problem
Known results

$\mathcal{M}$ = Unambiguous functional transducers

Feasible for any asynchronous specification [Kobayashi, 1969]

$\mathcal{M}$ = Sequential transducers

$\mathcal{S}$ (async. transducers)	**Complexity**
Nondeterministic	Undecidable [Carayol and Löding, 2014]
Finite-valued	3-ExpTime [Filiot et al., 2016]
Multi-sequential	PSpace-c

Asynchronous transducer realisability problem
Known results

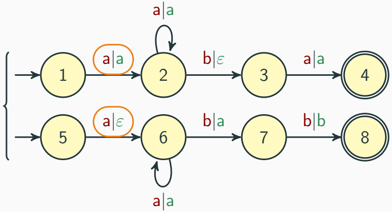
$\mathcal{M}$ = Unambiguous functional transducers

Feasible for any asynchronous specification [Kobayashi, 1969]

$\mathcal{M}$ = Sequential transducers

$\mathcal{S}$ (async. transducers)	**Complexity**
Nondeterministic	Undecidable [Carayol and Löding, 2014]
Finite-valued	3-ExpTime [Filiot et al., 2016]
Multi-sequential	PSpace-c

Asynchronous transducer realisability problem

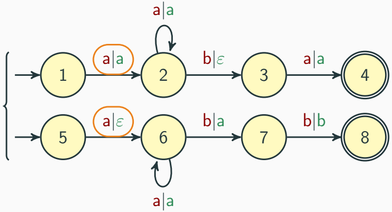


Delay

$$\mathrm{del}(u_1, u_2) = (\ell^{-1}u_1, \ell^{-1}u_2)$$

$$\ell = u_1 \wedge u_2$$

Asynchronous transducer realisability problem

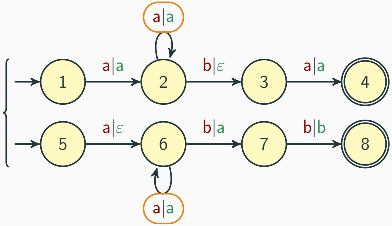


Delay

$\mathrm{del}(u_1, u_2) = (\ell^{-1}u_1, \ell^{-1}u_2)$

$$\mathrm{del}(a, \varepsilon) \quad = \quad (a, \varepsilon)$$

Asynchronous transducer realisability problem

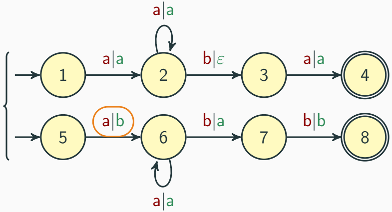


Delay

$$\mathrm{del}(u_1, u_2) = (\ell^{-1}u_1, \ell^{-1}u_2)$$

$$\begin{array}{ccc} \mathrm{del}(a, \varepsilon) & = & (a, \varepsilon) \\ \parallel & & \\ \mathrm{del}(aa, a) & = & (a, \varepsilon) \end{array}$$

Asynchronous transducer realisability problem

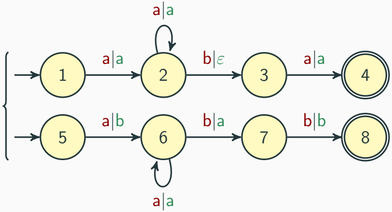


Delay

$\mathrm{del}(u_1, u_2) = (\ell^{-1}u_1, \ell^{-1}u_2)$

$$\begin{array}{ccc} \mathrm{del}(a, b) & = & (a, b) \\ \not\Downarrow & & \\ \mathrm{del}(aa, ba) & = & (aa, ba) \end{array}$$

Asynchronous transducer realisability problem



Delay

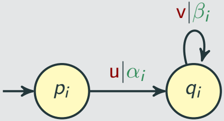
$\mathrm{del}(u_1, u_2) = (\ell^{-1}u_1, \ell^{-1}u_2)$

$$\begin{array}{ccc} \mathrm{del}(a, b) & = & (a, b) \\ \nparallel & & \\ \mathrm{del}(aa, ba) & = & (aa, ba) \end{array}$$

Critical loop

Triple (u, v, X) s.t.:

1. For all $\mathcal{D}_i \in X$,



2. For all $\mathcal{D}_i \notin X$, no run on u
3. For two transducers $\mathcal{D}_i, \mathcal{D}_j \in X$, delays accumulate: $\mathrm{del}(\alpha_i, \alpha_j) \neq \mathrm{del}(\alpha_i\beta_i, \alpha_j\beta_j)$

Asynchronous transducer realisability problem

Recursive characterisation

$\mathcal{T} = \uplus_{i=1}^{k} \mathcal{D}_i$ is realisable iff for all critical loops (u, v, X), there exists $Y \subsetneq X$ s.t.:

1. Delays do not accumulate:
   $\forall \mathcal{D}_i, \mathcal{D}_j \in Y, \mathrm{del}(\alpha_i, \alpha_j) = \mathrm{del}(\alpha_i\beta_i, \alpha_j\beta_j)$
2. The domain is still covered: $u^{-1}\mathrm{dom}(\mathcal{T}) = \bigcup_{i \in P} u^{-1}\mathrm{dom}(\mathcal{D}_i)$
3. The residual specification $(u, \ell)^{-1} \left[\!\left[\biguplus_{i \in Y} \mathcal{D}_i \right]\!\right]$ is realisable

ℓ longest common prefix of the α_i's

➔ Can be easily checked in **ExpTime**

Asynchronous transducer realisability problem

Theorem

Asynchronous sequential transducer synthesis from multi-sequential specifications is **PSpace**-complete.

PSpace-easiness: a non-recursive characterisation

Witness of non-satisfaction

- Unfolding of the recursive characterisation
- Reformulation of delay difference

➔ Can be found in **PSpace**

PSpace-hardness

➔ Similar to the synchronous case

Conclusion

Multi-sequential specifications

- Membership decidable in **PTime**
- Sequential realisability is **PSpace**-c both in synchronous and asynchronous cases

➔ Improvement of the general case:

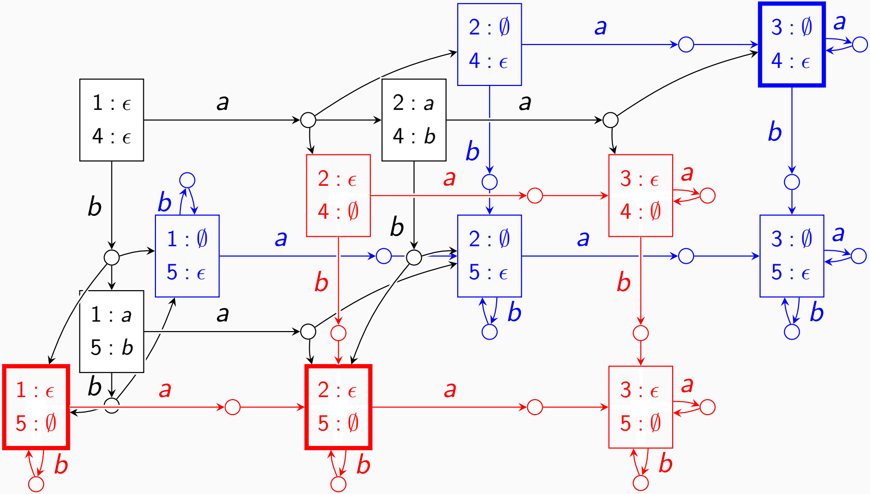
  - synchronous = **ExpTime-c**
  - asynchronous = **undecidable**

Synthesis game

➔ Practical synthesis algorithm

- Suitable for any type of specification defined by transducers (might not terminate)

The synthesis game



Büchi, J. R. and Landweber, L. H. (1969).
Solving Sequential Conditions by Finite-State Strategies.
Transactions of the American Mathematical Society, 138:295–311.

Carayol, A. and Löding, C. (2014).
Uniformization in Automata Theory.
In *Proceedings of the 14th Congress of Logic, Methodology and Philosophy of Science Nancy, July 19-26, 2011*, pages 153–178, London. College Publications.

Choffrut, C. and Schützenberger, M. P. (1986).
Décomposition de fonctions rationnelles.
In *2nd Annual Symposium on Theoretical Aspects of Computer Science, STACS*, pages 213–226.

Filiot, E., Jecker, I., Löding, C., and Winter, S. (2016).
On equivalence and uniformisation problems for finite transducers.
In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, Rome, Italy*, pages 125:1–125:14.

Jecker, I. and Filiot, E. (2015).
Multi-sequential word relations.
In *Proceedings of the 19th International Conference on Developments in Language Theory, DLT 2015, Liverpool, UK, July 27-30*, pages 288–299.

Kobayashi, K. (1969).
Classification of formal languages by functional binary transductions.
Information and Control, 15(1):95–109.

Pnueli, A. and Rosner, R. (1989).
On the synthesis of a reactive module.
In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, pages 179–190, New York, NY, USA. ACM.