A Term-based Approach for Generating Finite Automata from Interaction Diagrams

LVP MTV2 2023 - Erwan Mahe

Université Paris-Saclay, CEA-List, Fr-91120, Palaiseau
Interactions

Formal language (terms = binary trees)

- emissions
- receptions
- empty \( \emptyset \)
- strict, alt
- seq / par
- coreg
- loop
Previous works

Framework of equivalent formal semantics [TASE 2022 & SCP 2024]
(without concurrent regions):

▶ operational & denotational style, expressed as traces or multi-traces
▶ (multi-)trace analysis (Offline Runtime Verification) [FASE 2020 & SAC 2021]
▶ multi-trace analysis under partial observation [FSEN 2023]
Concurrent-regions

Adapting this concept to our framework [JOT 2024]:

Generalizes & subsumes weak sequencing & interleaving
Treated as any other operator
Motivation:

- now that we have a direct semantics
- links with other formalisms & translations
- a first candidate: finite automata

Still, interactions:

- may have non-regular trace languages
- due to non-strictly-sequential loops
- so we consider a subset

Interactions & Regular Languages
Literature on translating interactions to FA
Related works on translating Interactions to FA

For UML-SD in:
Runtime monitoring of web service conversations, Simmonds, J. Gan, Y. et al., IEEE TSC 2009

Based on works on graphs of basic Message Sequence Charts:
Model checking of message sequence charts, Alur, R. & Yannakakis, M., CONCUR 1999
Literature on translating interactions to FA

Compositional translation mechanism

- Partial orders
- Linearization
- Composition using NFA operators
  \( (a_1.(a_2|(a_3^*)).a_4)^* \)
  enforces strict sequencing

Model checking
undecidable under asynchronous interpretation

CONCUR 1999
TSC 2009
Literature on translating interactions to FA

Strict and Weak sequencing issue

\[
\text{initial interaction:} \quad \text{loop}_S(\text{seq}(i_1, \text{seq(alt}(i_2, \text{loop}_S(i_3)), i_4))))
\]

\[
\text{synchronous translation:} \quad \text{loop}_S(\text{strict}(i_1, \text{strict(alt}(i_2, \text{loop}_S(i_3)), i_4))))
\]

\[
\text{asynchronous translation:} \quad \text{loop}_W(\text{seq}(i_1, \text{seq(alt}(i_2, \text{loop}_W(i_3)), i_4))))
\]

TSC 2009 only offers the synchronous translation because it uses NFA concatenation.
Ideal case

Our example is an ideal case:
- the asynchronous translation is possible
- all three trace languages are equivalent

This is because the MSC graph is \textit{local-choice}. 
Our example is an ideal case:

- the asynchronous translation is possible
- all three trace languages are equivalent

This is because the MSC graph is **local-choice**.
Ideal case

Our example is an ideal case:

▶ the asynchronous translation is possible
▶ all three trace languages are equivalent

This is because the MSC graph is local-choice.
Ideal case

Our example is an ideal case:

▶ the asynchronous translation is possible
▶ all three trace languages are equivalent

This is because the MSC graph is local-choice.
With a non-local-choice MSC graph we may:

- have distinct outputs from synchronous and asynchronous translations
Non-local-choice: Example 1

With a non-local-choice MSC graph we may:

- have distinct outputs from synchronous and asynchronous translations

```
loop5
m1a
m1b
m2b
m3a
m3b
```

```
alt
m2b
m3a
m4
```
With a non-local-choice MSC graph we may:

▶ have distinct outputs from synchronous and asynchronous translations

different trace languages depending on interpretation of “then”
circled above (correct w.r.t. initial interaction is asynchronous)
Non-local-choice : Example 2

With a non-local-choice MSC graph we may:

- not be able to perform the asynchronous translation
Non-local-choice: Example 2

With a non-local-choice MSC graph we may:

- not be able to perform the asynchronous translation
Non-local-choice : Example 2

With a non-local-choice MSC graph we may:

- not be able to perform the asynchronous translation

the “then” circled above cannot be interpreted asynchronously
(if so, includes non regular $\bigcup_{n \in \mathbb{N}} \{x^n.y^n\}$)
Combining the two previous example, we have a case in which:

- the asynchronous translation is impossible
- the synchronous translation has a trace language that is distinct from that of the initial interaction
Non-local-choice : Example 4

What about concurrent regions?
- it even becomes difficult to define the graph properly
Discussion on the compositional graph-based translations

Observation:
- only handles well local-choice specifications
- difficult to apply when using concurrent-region

But why is this difficult?:
- weak sequencing is treated differently from the other symbols of the interaction language (i.e., strict sequencing, alternative, interleaving etc.)
- this makes nesting difficult
- and also hybrid operators like coreg which is between weak sequencing and interleaving

In TSC 2009 seq can only appear in basic SDs (leaf sub-terms of the tree structure with only strict and seq operators) i.e., we cannot nest more complex operators.
Our new method for generating NFA from Interactions
Our new method for generating NFA from Interactions

**Principle of the method**

Our method takes advantage of our framework of equivalent semantics. It is similar to that (used to generate NFA from ERE) from: *Testing extended regular language membership incrementally by rewriting*, Roşu, G. & Viswanathan, M. RTA 2003

- each node of the NFA corresponds to a derivative of the initial interaction
- each edge corresponds to the execution of an action
Our new method for generating NFA from Interactions

Using the operational-style semantics

\[
\begin{align*}
\text{act} & \quad a \xrightarrow{a} \emptyset \\
\text{loops} & \quad \text{loops}(i_1) \xrightarrow{a} \text{strict}(i_1', \text{loops}(i_1)) \\
\text{strict} & \quad i_2 \xrightarrow{a} i_2' \quad i_1 \xrightarrow{L} i_1' \\
\text{strict-right} & \quad i_1, i_2 \xrightarrow{a} i_2' \\
\text{alt-choice} & \quad i_j \xrightarrow{a} i_j' \quad i_k \xrightarrow{a} i_k' \\
\text{alt} & \quad \text{all}(i_1, i_2) \xrightarrow{a} i_j' \\
\text{loop} & \quad i_1 \xrightarrow{a} i_1' \\
\text{f-left} & \quad f(i_1, i_2) \xrightarrow{a} f(i_1', i_2) \\
\text{cr-right} & \quad \text{cr}_T(i_1, i_2) \xrightarrow{a} \text{cr}_T(i_1', i_2') \\
\text{all-delay} & \quad \text{all}(i_1, i_2) \xrightarrow{a} \text{all}(i_1', i_2')
\end{align*}
\]
Our new method for generating NFA from Interactions

Using the denotational-style semantics & rewriting

Semantic equivalence proof (Coq)
https://github.com/erwanM974/coq_interaction_semantics_equivalence_with_coregions

TRS convergence proof (TTT2/CSI)
https://github.com/erwanM974/hibou_trs_simplify_empty
Our new method for generating NFA from Interactions

Concise NFA generation

Our method:

▶ in practice rewriting is done on-the-fly when computing derivatives
▶ we can compute concise NFA directly without requiring costly NFA minimization/reduction techniques

Why it matters:

▶ NFA can be used in various practical settings (monitoring, model checking etc.)
▶ the less states they have the more efficiently they can be used
▶ equivalent DFA may have exponentially more states
▶ NFA state minimization is PSPACE-complete
Experimentations
Experimentations

Implementation:

- interaction manipulation tool:
  HIBOU https://github.com/erwanM974/hibou_label

- nfa toolbox:
  AUTOUR https://github.com/erwanM974/autour_core

Outline: 2 sets of experiments:

1. assessing the reduced nature of generated NFA and the performance of the translation
   https://github.com/erwanM974/hibou_nfa_generation

2. exploiting the generated NFA for runtime verification
   https://github.com/erwanM974/hibou_nfa_trace_analysis
Digital locks usecase
NFA generation assessment with networks of digital locks

| locks  | $|Q|$ nfa$_s$ | $\downarrow$ nfa$_s$ | $|Q|$ min$_{NFA}$ | $\downarrow$ min$_{NFA}$ | $|Q|$ min$_{DFA}$ | $\downarrow$ min$_{DFA}$ |
|--------|--------------|------------------|-----------------|-------------------|-----------------|-------------------|
| 1 lock | 8            | 230 $\mu$s      | 8               | 30s               | 14              | 130 $\mu$s       |
| 4 locks| 105          | 7400 $\mu$s     | N/A             | timeout           | 312             | 8000 $\mu$s      |
| 8 locks| 2881         | 1s               | N/A             | timeout           | 16274           | 2s               |

(a) Comparing nfa$_s$ with equivalent minimal NFA and DFA

| locks  | $|Q|$ nfa$_s$ | $\downarrow$ nfa$_s$ | $|Q|$ compo | $\downarrow$ compo | $|Q|$ min$_{DFA}$ | $\downarrow$ min$_{DFA}$ |
|--------|--------------|------------------|-----------|-------------------|-----------------|-------------------|
| 1 lock | 8            | 230 $\mu$s      | 13        | 370 $\mu$s        | 14              | 120 $\mu$s       |
| 4 locks| 97           | 3500 $\mu$s     | 223       | 7300 $\mu$s       | 298             | 8100 $\mu$s      |
| 8 locks| 1624         | 0.38s           | 5904      | 0.15s             | 9374            | 2.3s             |

(b) Comparing nfa$_s$ with compositional NFA compo generation method
Experimentations

Applicative usecases

Platoon

LVP - MTV2 : Generating NFA from Interactions
Applicative usecases
Applicative usecases

ABP
### Application to runtime verification

<table>
<thead>
<tr>
<th>NFA</th>
<th>Traces</th>
<th>Rates in $a.s^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Q</td>
<td>$</td>
</tr>
<tr>
<td>ABP [33]</td>
<td>68</td>
<td>7300$\mu$s</td>
</tr>
<tr>
<td>Platoon [3]</td>
<td>90</td>
<td>2900$\mu$s</td>
</tr>
<tr>
<td>HR [21]</td>
<td>101</td>
<td>4300$\mu$s</td>
</tr>
</tbody>
</table>
Conclusion & Perspectives
Conclusion:

▶ A novel, expressive and adaptive (may include new symbols) method to translate interactions to NFA
▶ Solves issues related to nesting and hybridation of weak sequencing
▶ Allows generating “small” NFA

Perspectives:

▶ application to model checking
▶ application to monitor synthesis
▶ characterize the generated NFA (under which conditions can they be the minimal NFA)
Thanks

any questions ?