Learning minimal separating DFA for Compositional Verification

Karsten Fix

February 23, 2017

Compositional Verification

System

- consist of Components M_1 and M_2
- shall satisfy a Property P
- can be describe by regular Laguages $\mathcal{L}(M_1), \mathcal{L}(M_2), \mathcal{L}(P)$.

To verify this, there's an inference rule, that says:

$$\frac{\mathcal{L}(M_1) \cap \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(P) \qquad \mathcal{L}(M_2) \subseteq \mathcal{L}(\mathcal{A})}{\mathcal{L}(M_1) \cap \mathcal{L}(M_2) \subseteq \mathcal{L}(P)}$$

Intuitively: We can find an Assumption $\mathcal{A}$ for M_2.

Compositional Verification

This premise of the interference rule:

$$\mathcal{L}(M_1) \cap \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(P)$$

can be rewritten as:

$$\mathcal{L}(\mathcal{A}) \subseteq \overline{\mathcal{L}(M_1) \cap \overline{\mathcal{L}(P)}}$$

Substitution:

$$\mathcal{L}(M_2) \subseteq \mathcal{L}(\mathcal{A}) \subseteq \overline{\mathcal{L}(M_1) \cap \overline{\mathcal{L}(P)}}$$

Then $\mathcal{A}$ is separating DFA for $\mathcal{L}(M_2)$ and $\mathcal{L}(M_1) \cap \overline{\mathcal{L}(P)}$.

Separating DFA

Definition

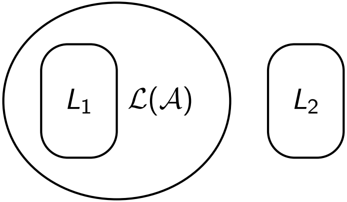
Let $L_1, L_2 \subseteq \Sigma^*$ be *disjoint* regular languages. Then a DFA $\mathcal{A}$ is called **separating** DFA for L_1 and L_2, if it statisfies:

1. $L_1 \subseteq \mathcal{L}(\mathcal{A})$
2. $\mathcal{L}(\mathcal{A}) \cap L_2 = \emptyset$

Or equivalently: $L_1 \subseteq \mathcal{L}(\mathcal{A}) \subseteq \overline{L_2}$

Separating DFA

That means: $\mathcal{A}$ accepts at least all words of L_1 and rejects all words of L_2.



3DFA

Definition

A 3DFA $\mathcal{C}$ is defined like a DFA:

$$\mathcal{C} = (Q, \Sigma, \delta, q_0, \underbrace{Acc, Rej, Dont}_{Q})$$

but all states are partitioned into three sets:

- $Acc \subseteq Q$: accepting states
- $Rej \subseteq Q$: rejecting states
- $Dont \subseteq Q$: Don't care states

That means: $Acc \cap Rej \cap Dont = \emptyset$

3DFA

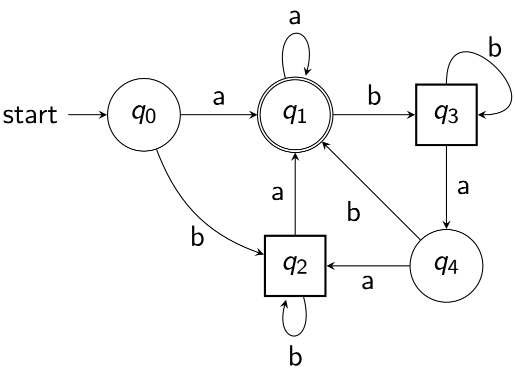
Given a 3DFA $\mathcal{C}$ a string $w \in \Sigma^*$ is:

- **accepted** by $\mathcal{C}$ if $\hat{\delta}(q_0, w) \in Acc$
- **rejected** by $\mathcal{C}$ if $\hat{\delta}(q_0, w) \in Rej$
- called **don't care** string if $\hat{\delta}(q_0, w) \in Dont$

3DFA - Visualisation

A 3DFA will be visualised, using squares for the don't care states. Rejecting and accepting states are visualised as circles, as usual.

An Example:



3DFA $\rightarrow$ DFA $\mathcal{C}^+$

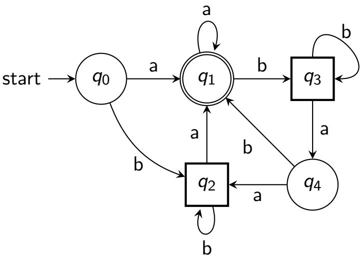
Definition

We define a DFA $\mathcal{C}^+$, where the don't care states become accepting states:

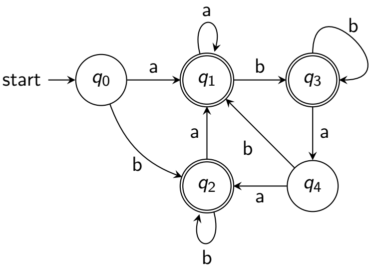
$$\mathcal{C}^+ = (Q, \Sigma, q_0, \delta, Acc \cup Dont)$$

3DFA $\rightarrow$ DFA $\mathcal{C}^+$

Example 3DFA $\mathcal{C}$:



DFA $\mathcal{C}^+$:



3DFA $\rightarrow$ DFA $\mathcal{C}^-$

Definition

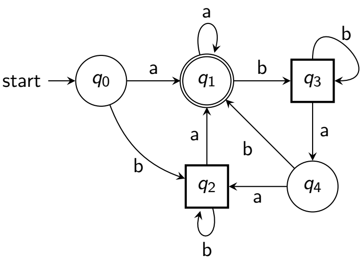
We define a DFA $\mathcal{C}^-$, where only the accepting states are accepting:

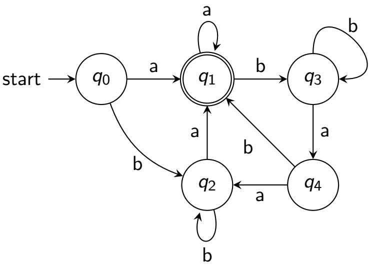
$$\mathcal{C}^- = (Q, \Sigma, q_0, \delta, Acc)$$

3DFA $\rightarrow$ DFA $\mathcal{C}^-$

Example 3DFA $\mathcal{C}$:

DFA $\mathcal{C}^-$:





Consistency

Definition

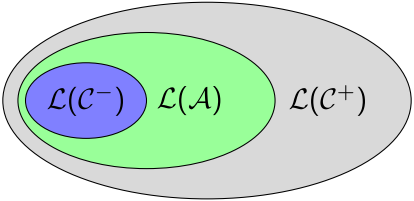
Let $\mathcal{A}$ be a DFA, then it will be called **consistent** with a 3DFA $\mathcal{C}$, if both are accepting and rejecting the same words. Means:

1. $\mathcal{L}(\mathcal{C}^-) \subseteq \mathcal{L}(\mathcal{A})$
2. $\mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{C}^+)} = \emptyset$

Or equivalently: $\mathcal{L}(\mathcal{C}^-) \subseteq \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}^+)$

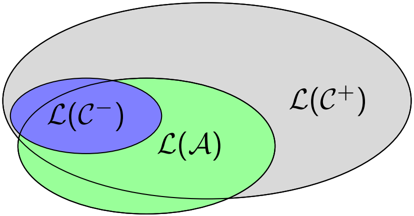
Consistency - Visualisation

DFA $\mathcal{A}$ consistent with a 3DFA $\mathcal{C}$:



Consistency - Visualisation

DFA $\mathcal{A}$ inconsistent with a 3DFA $\mathcal{C}$:



Soundness

Definition

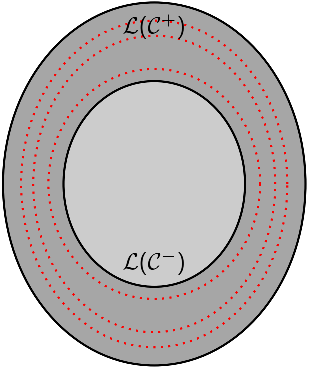
A 3DFA $\mathcal{C}$ is called **sound** with respect to L_1 and L_2, if any with $\mathcal{C}$ consistent DFA $\mathcal{A}$ separates L_1 and L_2.

Remember

- Consistency: $\mathcal{L}(\mathcal{C}^-) \subseteq \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}^+)$
- Separating: $L_1 \subseteq \mathcal{L}(\mathcal{A}) \subseteq \overline{L_2}$

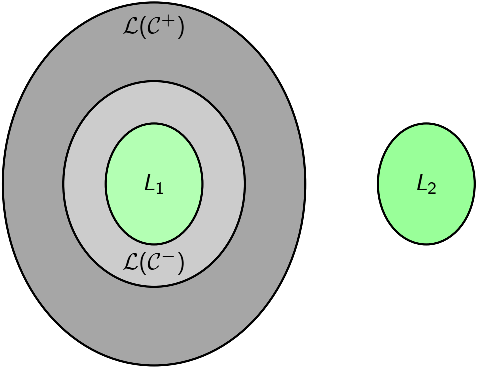
Soundness - Visualisation

Any DFA $\mathcal{A}$ consistent with 3DFA $\mathcal{C}$:



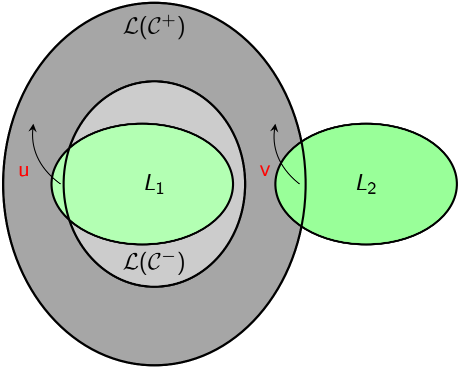
Soundness - Visualisation

is separating DFA for L_1 and L_2, so $\mathcal{C}$ is sound:



Soundness - Visualisation

An unsound 3DFA $\mathcal{C}$:



Completeness

Definition

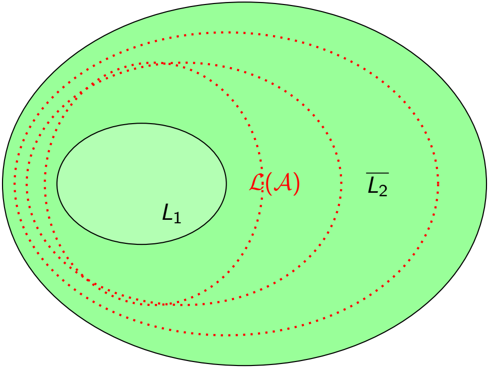
A 3DFA $\mathcal{C}$ is called **complete** with respect to L_1 and L_2, if any separating DFA $\mathcal{A}$ for L_1 and L_2 is consistent with $\mathcal{C}$.

Remember

- Separating: $L_1 \subseteq \mathcal{L}(\mathcal{A}) \subseteq \overline{L_2}$
- Consistency: $\mathcal{L}(\mathcal{C}^-) \subseteq \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}^+)$

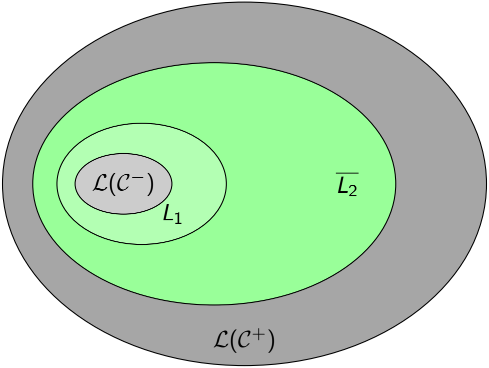
Completeness - Visualisation

Any DFA $\mathcal{A}$ separating L_1 and L_2:



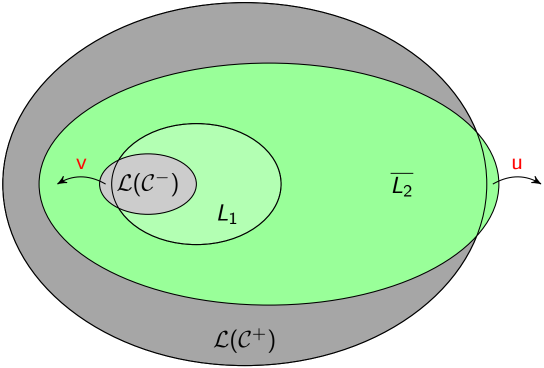
Completeness - Visualisation

is consistent with 3DFA $\mathcal{C}$, so it is complete:



Completeness - Visualisation

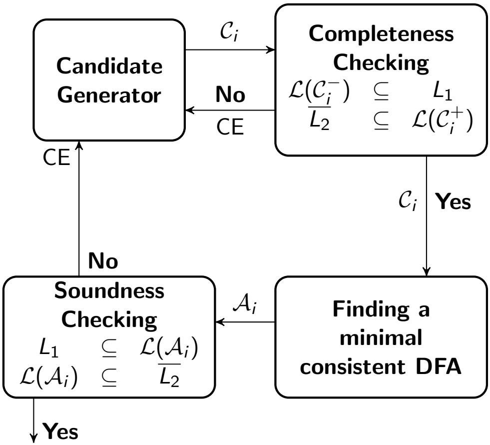
An incomplete 3DFA $\mathcal{C}$:



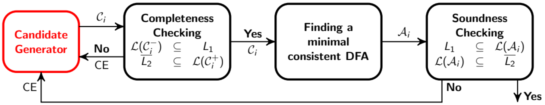
Summary

1. DFA $\mathcal{A}$ is separating DFA if: $L_1 \subseteq \mathcal{L}(\mathcal{A}) \subseteq \overline{L_2}$
2. $\mathcal{L}(\mathcal{C}^-)$: are all words a 3DFA $\mathcal{C}$ accepts
3. $\overline{\mathcal{L}(\mathcal{C}^+)}$: are all words a 3DFA $\mathcal{C}$ rejects
4. DFA $\mathcal{A}$ is consistent with 3DFA $\mathcal{C}$ if: $\mathcal{L}(\mathcal{C}^-) \subseteq \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{C}^+)$
5. 3DFA is sound if $L_1 \subseteq \mathcal{L}(\mathcal{C}^-)$ and $\mathcal{L}(\mathcal{C}^+) \subseteq \overline{L_2}$
6. 3DFA is complete if $\mathcal{L}(\mathcal{C}^-) \subseteq L_1$ and $\overline{L_2} \subseteq \mathcal{L}(\mathcal{C}^+)$

Overview of L^{sep}



Teacher



The algorithm assumes a teacher that can answer:

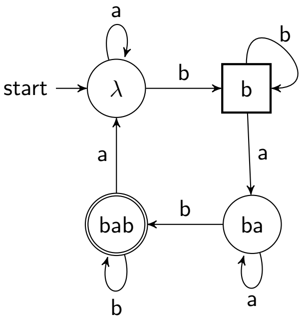
- **membership queries** $w \overset{?}{\in} L_1$, $w \overset{?}{\in} L_2$ with:
  - $+$ if $w \in L_1$
  - $-$ if $w \in L_2$
  - ? otherwise, i.e. don't care.

Candidate Generator

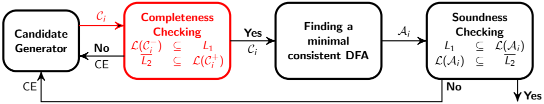
Based on the L^*-algorithm a 3DFA $\mathcal{C}_i$ is computed by asking **membership queries** and building an observation table with entries: $+$, $-$ and ?, depending on the answers of the teacher.

	λ	b
λ	$-$	?
b	?	?
ba	$-$	$+$
bab	$+$	$+$
a	$-$	?
$\vdots$	$\vdots$	$\vdots$

$\xrightarrow{\mathcal{C}_i}$



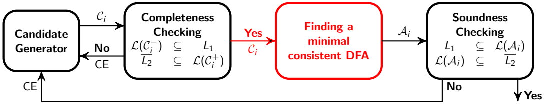
Teacher



The teacher also answers:

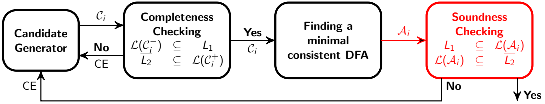
- **containment queries**, such as $\mathcal{L}(\mathcal{C}_i^-) \overset{?}{\subseteq} L_1$, $\overline{L_2} \overset{?}{\subseteq} \mathcal{L}(\mathcal{C}_i^+)$ with:
  - **Yes**, if both subset relations are true
  - **No**, if one relation is false. It also gives a counterexample (CE)

Minimal consistent DFA



L^{sep} translates the 3DFA $\mathcal{C}_i$ into a mealy automaton, which will be minimized with existing algorithms. After minimizing it, it will be translated into a consistent DFA $\mathcal{A}_i$.

Soundness Checking

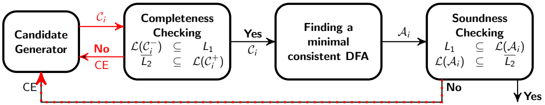


Now the DFA $\mathcal{A}_i$ is minimal, complete and consistent. The last step is to check for soundness, using **containment queries**:

- $L_1 \overset{?}{\subseteq} \mathcal{L}(\mathcal{A})$
- $\mathcal{L}(\mathcal{A}) \overset{?}{\subseteq} \overline{L_2}$

In case both subset relations are true, we have the minimal separating DFA for L_1 and L_2. Otherwise there is a CE sent to the Candidate Generator.

Runtime



Let n be the size of the minimal sound and complete 3DFA.
Let m be the size of the longest CE, then

- at most $n - 1$ incorrect 3DFAs
- $O(n^2 + n \cdot log(m))$ queries, for a complete and sound 3DFA

References

This talk is mainly based on

- Y.-F. Chen, A. Farzan, E. M. Clarke, Y.-K. Tsay, B.-Y. Wang: Learning Minimal Separating DFA for Compositional Verification. S.Kowalewski, A. Phillippou: TACAS 2009, LNCS 5505, pp. 31-45, 2009. Springer-Verlag Berlin-Heidelberg 2009.

References

and in addtion the following has been studied:

- Gupta, A., McMillilan, K.L., Fu, Z.: Automated assumption generation for compositional verification. In: Damm, W., Hermanns, H.(eds.) CAV 2007. LNCS, vol 4590, pp. 420-432. Springer, Heidelberg (2007)
- Angluin, D: Learning regular sets from queries and counterexamples. Information and Computation 75(2), 87-106 (1987)
- Paull, M.C., Unger, S.H.: Minimizing the number of states in incompletely specified sequential switching functions. IRE Transitions on Electronic Computers EC-8, 356366 (1959)

Thanks...

... for Listening and Attention.
Any Questions?