State-merging DFA induction algorithms with mandatory merge constraints From MSM to ASM

Bernard Lambeau, Christophe Damas and Pierre Dupont



Computing Science and Engineering Department Université catholique de Louvain – Belgium

September 23, 2008

(UCL Machine Learning Group)

From MSM to ASM

Requirements Engineering (RE)

- It has been claimed that the hardest part in building a software system is deciding precisely what the system should do
- One can automate parts of this RE process by learning behavior models from scenarios
- Scenarios are strings of possible events which can be generalized to form a language of acceptable behaviors
- Such languages are conveniently represented by finite-state machines

- Scenarios describe interactions between the software-to-be and its environment
- Scenarios are typical examples of system usage provided by an end-user involved in the requirements elicitation process





Synthesis of behavior models and DFA induction A win-win situation

- Regular languages are considered to be powerful enough
- DFAs offer a convenient representation for model checking and code generation
- The typical size of such DFAs is about 20...100 states
 - \blacktriangleright \Rightarrow hard to design exactly by a software analyst
 - ➤ ⇒ not problematic for state-of-the-art DFA induction algorithm (RPNI, BlueFringe)
- Typical alphabet size $\approx 10...20$
- The end-user can really be used as an oracle in practice

State-merging induction with membership queries

Our previous work: the QSM algorithm [Damas et al. 05], [Dupont et al. 08]

- An extension to RPNI or BlueFringe (also known as redBlue) with membership queries
- The limited amount of positive and negative scenarios provided initially by an end-user can be enriched by asking membership queries



A high-level Message Sequence Chart Flow-charting of various scenarios



イロト イヨト イヨト ・ 注

A high-level Message Sequence Chart Flow-charting of various scenarios



This information defines Mandatory Merge Constraints between some states of the prefix tree acceptor (PTA)



(UCL Machine Learning Group)

State-merging DFA induction

Prefix-Tree Acceptor (PTA)



Quotient automaton



```
Algorithm STATE-MERGING DFA INDUCTION ALGORITHM

Input: A positive and negative sample (S_+, S_-)

Output: A DFA A consistent with (S_+, S_-)

// Compute a PTA, let N denote the number of its states

PTA \leftarrow Initialize (S_+, S_-); \pi \leftarrow \{\{0\}, \{1\}, ..., \{N-1\}\}

// Main state-merging loop

while (B_i, B_j) \leftarrow ChoosePair (\pi) do

\pi_{new} \leftarrow Merge(\pi, B_i, B_j)

if Compatible (PTA/\pi_{new}, S_-) then

\perp \pi \leftarrow \pi_{new}
```

return PTA/π

The Merge function also reduces non-determinism



э

Image: A matrix

Tree invariant

Tree invariant property

- At least one of the 2 states implied in a merging operation is the root of a (sub)-tree
- True for RPNI, BlueFringe (= redBlue), etc
- Simplification of the actual implementation



Incompatibility constraints



- Augmented PTA with positively accepting states (= grey) and negatively accepting states (= black)
- The Merge function reduces non-determinism and checks such coloring constraints
 - States having different colors may not be merged
 - States having the same color can be merged
- Coloring constraints define incompatibility between states from positive and negative information or additional domain knowledge [Coste et al. 04], [Dupont et al. 08]

Mandatory merge constraints



- Another kind of domain knowledge defines mandatory merge constraints between states sharing the same labels
- Labeling constraints are the logical counterpart to the coloring constraints
 - States with the same label must be merged
 - States with different labels can be merged
- A fully labeled PTA does not define a trivial induction problem
 - Without coloring constraints (such as those provided by the negative sample) all states will be merged

MSM algorithm

Algorithm MSM Input: A non-empty initial positive and negative sample (S_+, S_-) Input: Labeling and coloring constraints Output: A DFA *A* consistent with (S_+, S_-) and all constraints // Compute a PTA, let *N* denote the number of its states

```
return PTA/\pi
```

MSM does not satisfy the tree invariant property

MSM is a straightforward extension to standard state-merging algorithms **However...**

MSM does not satisfy the tree invariant property

MSM is a straightforward extension to standard state-merging algorithms **However...**

- Labeling constraints can force to merge states such that the resulting automaton has a general graph structure
- The tree invariant property is no longer satisfied
- Recursive merging to reduce non-determinism naturally stops even for general graphs



Experiments on synthetic data



э

15.

・ロト ・ 四ト ・ ヨト ・ ヨト

From MSM to ASM

Requirements engineering case study



(UCL Machine Learning Group)

From MSM to ASM

< 17 ▶

DFA induction from a positive DFA and a negative sample

Algorithm ASM Input: A positive DFA A_+ and a negative sample S_- Output: A DFA A consistent with (A_+, S_-) // Augment the automaton A_+ with states // marked/added from $S_ M \leftarrow \text{Augment}(A_+, S_-)$ // Compute the natural order on M $\pi \leftarrow \text{NatOrder}(M)$ // Main state-merging loop

 $\pi \leftarrow \text{Generalize}(\pi)$

return M/π

(I) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1))

DFA induction from positive and negative DFAs

```
Algorithm ASM*
Input: A positive DFA A_+ and a negative DFA A_- such that L(A_+) \cap L(A_-) = \emptyset
Output: A DFA A consistent with (A_+, A_-)
// Augment the automaton A_+ with states
// marked/added from S_
M \leftarrow \text{Product}(A_{\perp}, A_{\perp})
   Compute the natural order on M
\pi \leftarrow \text{NatOrder}(M)
   Main state-merging loop
\pi \leftarrow \text{Generalize}(\pi)
```

return M/π

Product DFA



From MSM to ASM

イロト イヨト イヨト イヨト

Take home message

- Mandatory merge constraints are introduced to model domain knowledge, for instance, from a Requirements Engineering perspective
- Mandatory merge constraints form the logical counterpart to incompatibility constraints
- The MSM algorithm deals with both types of constraints
- MSM is a straightforward extension to RPNI or BlueFringe but
 - without satisfying the tree-invariant property
 - using recursive merging extended to general graphs
- MSM gives rise to ASM* to induce DFAs from prior positive and negative DFAs
 - interesting from a practical viewpoint
 - may require a new theoretical framework

(I) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1)) < ((1))

Future work

- MSM implementation with the BlueFringe search order (easy)
- MSM as an extension to QSM for active learning with queries (somewhat more challenging)
- Other applicative contexts where mandatory merge constraints are natural
- Further analyze ASM*
 - theoretically: characteristic samples?
 - practically: experimental protocol?