Symbolic Automata = Automata + SMT solvers

Rajeev Alur
Loris D’Antoni
Margus Veanes

01/24/14, POPL14
Motivations

✔️ Automata and Transducers are great!!
✔️ Used in many applications (NLP, XML, program analysis, regex matching...)
✖️ Can only handle finite alphabets
✖️ Do not scale when the alphabet is very big (UTF16 has $2^{16}$ elements)
A simple analysis problem

Given a list of even numbers

```
Public Trans map_mod : IntList -> IntList {
  nil() to (nil [0])
  | cons(x) to (cons [mod (x+5) 26] (map_inc x))
}
```

Always outputs a list of odd numbers
Symbolic Finite Automaton (SFA)

\[ A = (Q, q_0, F, \delta, \sigma) \]

Input sort: in this case int

\[ \lambda x. x \mod 2 = 1 \]

\[ \lambda x. x \mod 2 = 0 \]

Separate theory for the input alphabet

SMT SOLVER
Symbolic Finite Automata (SFA)

Execution Example

λx. x mod 2=0
λx. x mod 2=1

1 2 5 3

p p q p p

p is final → accept the input
Generalizing Classical Constructions

**Requirements:**
- Input theory must be a **Boolean algebra**, and
- Decidable (SMT solver)
Hopcroft’s algorithm for Minimization

\[ P := \{F, Q \setminus F\} \]

\[ W := \{\text{if } |F| < |Q \setminus F| \text{ then } F \text{ else } Q \setminus F\} \]

\textbf{while} \( W \neq \{\} \)

\[ R := \text{pickFrom}(W) \]

\textbf{foreach} \( a \) in \( \Sigma \)

\[ S := \delta^{-1}(R,a) \]

\textbf{while} \( \exists T \in P. \ T \cap S \neq \{\} \land T \setminus S \neq \{\} \)

\[ P, W := \text{split}(P, P \cap S, P \setminus S) \]

\textbf{return} partitioned DFA

D’Antoni, Veanes. Minimization of Symbolic Automata [POPL14]
Symbolic Finite Transducers (SFT)

Execution Example

\[
\text{even}(x)/ [x+1, x-1]
\]

\[
\text{odd}(x)/ [x+2\%4]
\]

\[
\begin{array}{cccc}
1 & 2 & 5 & 3 \\
\end{array}
\]

\[
\begin{array}{cccc}
3 & 1 & 3 & 3 & 1 \\
\end{array}
\]
Advantages of Symbolic Automata

• Infinite alphabets
• Separation of concerns between automaton and alphabet
  – UTF16 abstracted using BDDs
  – Integer using predicates over integers
• Succinctness
  – One transition captures many symbols
• Same closure and decidability properties of finite automata
  – Boolean closure and decidable equivalence
  – Closure under composition
An interesting property

Is it true that this transducer always outputs a list of odd numbers?

It can be checked: `isEmpty(T \cdot \text{complement}(O))`
Applications [Fast, PLDI14]

// Increments all the elements of the list by 1
Public Trans map_inc : IntList -> IntList {
    nil() to (nil [0])
    | cons(x) to (cons [(inc i)] (map_inc x))
}

// Removes all the odd elements from the list
Public Trans filter_ev : IntList -> IntList {
    nil() to (nil [0])
    | cons(x) where (odd i) to (filter_ev x)
    | cons(x) where (even i) to (cons [i] (filter_ev x))
}

// Compose the four functions into a single one
Def map_filt_2 : IntList -> IntList := (compose (compose map_inc filter_ev) (compose map_inc filter_ev))

// Empty list language
Public Lang not_emp_list : IntList {
    nil()
}

// Non-empty list language
Public Lang not_emp_list : IntList {
    cons(x)
}

// Check whether map_filt_2 ever outputs a non-empty list
Def map_filt_2_rest : IntList -> IntList := (restrict_out map_filt_2 not_emp_list)
AssertTrue (is_empty_trans map_filt_2_rest)
Conclusion

Results
Minimization [POPL14]
Analysis of string encoders [VMCAI13, CAV13]
Analysis of List/Tree Manipulating Programs [PLDI14]
Symbolic Nested Word Automata [Submitted CAV14]
Crowd Sourcing Regular Expressions [Submitting to HCOMP14]

Future work
Learning
Extend to more complex models