Certified Symbolic Transducer with the Application of String Solver

Shuanglong Kan
Barkhausen Institut, Germany

Abstract

Finite-State Automata (FA) and Finite-State Transducers (FT) are extensively utilized in programming languages and software engineering applications. For instance, regular expressions and their variations play a pivotal role in programming languages like JavaScript, Python, and others. Formalizing FAs and FTs in Coq, Isabelle/HOL, and other proof assistants are a popular topics. However, all these formalization are not practical in real-world applications. One of the reasons the transition labels are only single characters in the alphabet, for instance $q \xrightarrow{a} q'$ is a transition and a is a single character. For real-world applications, the alphabet an FA or FT may be enormous or even *infinite*. This classic way of transition definition can yield transitions explosion.

A more practical way is to formalize symoblic FAs [1] and FTs [2], in which transition labels are symbolic and can be infinite. The work of [3] has done the work for FAs, but FTs have not yet been formalized, which pose more challenges on proving the correctness of the formalization.

In this paper, we aim to filling this gap. We gave a symbolic transducer formalization in Isabelle/HOL. The formalization is refinement-based and extensible with different symbolic representations of transition labels. In order to evaluate its effect and efficiency, we applied it to a SMT string solver for modeling replacement operations in modern programming languages. The experimental results show the formalized symbolic transducer can efficient solver string constraints.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Dummy keyword

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Shuanglong Kan*: (Optional) author-specific funding acknowledgements

Acknowledgements I want to thank ...

1 Introduction

Automata and Transducers are crucial concepts in formal languages and have widely applications in programming languages and software engineering. For instance, [1] has shown the correspondence between regular expressions in modern regular expressions and variants of FAs and FTs. Other industrial usage of FAs and FTs is AWS access control polices checking.

Even though there various formalization of FAs and FTs in Coq, Isabelle, and other proof assistants. They are mainly based on classic definitions of FAs and FTs. There some drawbacks when come to practical applications. (1) transition labels are non-symbolic and usually finite. A classic and normal definition of a transition is $q \xrightarrow{a} q'$, where a is a character in an finite alphabet. This simple way will yield transition explosions. For instance, if the alphabet Σ is of the size $10,000$, then a transition from q to q' that accept any characters in Σ need to split into $10,000$ transitions. Automata product in this way will be very inefficient. Moreover, The alphabet are usually finite. For practical applications, infinite alphabet are often necessary. For instance, if the alphabet is all integers.

Symblic FAs and FTs [2,3,4] are extensions of classic FAs and FTs that make their application more practical. The transition labels are represented by algebra. For instances, intervals $('a' -' z')$, boolean algebras $(x\%2 == 0)$, or others. This symbolic way is more succinct and its support for infinite alphabet extend the expressive power of FAs and FTs.

Formalizing FAs and FTs in Isabelle/HOL, Coq, and other proof assists are more challenging compare with formalizing classic ones. Moreover, how to make the formalization extensible is also an important point to think, because, symbolic FAs and FTs support different algebras symbolic representations. For new algebra representations, we do not want repeat some proof works.

Fortunately, symblic FAs has been formalized in Isabelle/HOL [certistr] and experiments in [certistr] illustrates the efficiency and effective of symbolic FAs. Unfortunately FTs are not formalized yet. FTs are more powerful and expressive than symbolic FAs. For instance, when FTs are support, replacement operation in modern programming languages, such as Javascript, python and others can be modelled as FTs. And CertiStr can be extended to support replacement operations.

In this paper, we formalize symblic FTs based on symblic FAs. In order to solve the potential extension problem of FTs to support additional transition label theories, the formalization is refinement-based, in which at the abstraction level, transition labels are modeled as a general concept: sets. We can project any theory to sets. *extend this.* The most important operation that we formalized and proved is the application operation on an input language. More precisely, given an FA $\mathcal{A}$ (to represent a regular language) as the input language and a FT $\mathcal{T}$, the application operation $\mathcal{F}\ \mathcal{A}$ is the of output languages of FT after receiving the input language

In the next refinement level, the states and transitions are stored in efficient data structure based on Isabelle/HOL refinement framework [2]. For transition labels, we implemented an interval algebra, which are efficient for creating, checking, interaction. In the future the refinement of transition label from sets to other algebras instead of intervals can be done easily by implementing the common interfaces with interval algebras.

The paper is organized as follows. Section 2 presents the formalization of symbolic FTs. Section 3 presents the Core operations of symbolic FTs.

2 Formalization of Symbolic FTs

We firstly present a mathematical form of FTs.

▶ **Definition 1** (Label Theory). *A label theory*

▶ **Definition 2** (Symbolic FTs). *A symbolic FT is defined a quadruple (Q, Δ, I, F), where*

- *Q is a finite set of states,*
- *Δ is a finite set of transitions of the form (q, ϕ, f, q') or in a more succinct form $q \xrightarrow{(\phi,f)} q'$. ϕ is the input of the transition, which is a predicate, a symbolic representation of input characters. f is a label theory that generates the output of this transition,*
- *I is the set of initial states,*
- *F is the set of accepting states.*

▶ **Lemma 3** (Lorem ipsum). *Vestibulum sodales dolor et dui cursus iaculis. Nullam ullamcorper purus vel turpis lobortis eu tempus lorem semper. Proin facilisis gravida rutrum. Etiam sed sollicitudin lorem. Proin pellentesque risus at elit hendrerit pharetra. Integer at turpis varius libero rhoncus fermentum vitae vitae metus.*

Proof. Cras purus lorem, pulvinar et fermentum sagittis, suscipit quis magna.

```
record ('q, 'a, 'b) NFT =
         Q :: "'q set"
         Δ :: "('q, 'a, 'b) LTTS"
         I :: "'q set"
         F :: "'q set"
```

Figure 1 The definitions of *FTs*

Listing 1 Useless code.

```
for i:=maxint to 0 do
begin
    j:=square(root(i));
end;
```

Just some paragraph within the proof. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

▷ Claim 4. content...

Proof. content...

1. abc abc abc ◁

◀

▶ **Corollary 5** (Curabitur pulvinar, [2]). *Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.*

▶ **Proposition 6.** *This is a proposition*

Proposition 6 and Proposition 6 ...

2.1 Curabitur dictum felis id sapien

Curabitur dictum Corollary 5 felis id sapien Corollary 5 mollis ut venenatis tortor feugiat. Curabitur sed velit diam. Integer aliquam, nunc ac egestas lacinia, nibh est vehicula nibh, ac auctor velit tellus non arcu. Vestibulum lacinia ipsum vitae nisi ultrices eget gravida turpis laoreet. Duis rutrum dapibus ornare. Nulla vehicula vulputate iaculis. Proin a consequat neque. Donec ut rutrum urna. Morbi scelerisque turpis sed elit sagittis eu scelerisque quam condimentum. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nec faucibus leo. Cras ut nisl odio, non tincidunt lorem. Integer purus ligula, venenatis et convallis lacinia, scelerisque at erat. Fusce risus libero, convallis at fermentum in, dignissim sed sem. Ut dapibus orci vitae nisl viverra nec adipiscing tortor condimentum [1]. Donec non suscipit lorem. Nam sit amet enim vitae nisl accumsan pretium.

2.2 Proin ac fermentum augue

Proin ac fermentum augue. Nullam bibendum enim sollicitudin tellus egestas lacinia euismod orci mollis. Nulla facilisi. Vivamus volutpat venenatis sapien, vitae feugiat arcu fringilla ac. Mauris sapien tortor, sagittis eget auctor at, vulputate pharetra magna. Sed congue, dui nec vulputate convallis, sem nunc adipiscing dui, vel venenatis mauris sem in dui. Praesent a pretium quam. Mauris non mauris sit amet eros rutrum aliquam id ut sapien. Nulla aliquet fringilla sagittis. Pellentesque eu metus posuere nunc tincidunt dignissim in tempor dolor. Nulla cursus aliquet enim. Cras sapien risus, accumsan eu cursus ut, commodo vel velit. Praesent aliquet consectetur ligula, vitae iaculis ligula interdum vel. Integer faucibus faucibus felis.

- Ut vitae diam augue.
- Integer lacus ante, pellentesque sed sollicitudin et, pulvinar adipiscing sem.
- Maecenas facilisis, leo quis tincidunt egestas, magna ipsum condimentum orci, vitae facilisis nibh turpis et elit.

▶ Remark 7. content...

3 Pellentesque quis tortor

Nec urna malesuada sollicitudin. Nulla facilisi. Vivamus aliquam tempus ligula eget ornare. Praesent eget magna ut turpis mattis cursus. Aliquam vel condimentum orci. Nunc congue, libero in gravida convallis [3], orci nibh sodales quam, id egestas felis mi nec nisi. Suspendisse tincidunt, est ac vestibulum posuere, justo odio bibendum urna, rutrum bibendum dolor sem nec tellus.

▶ **Lemma 8** (Quisque blandit tempus nunc). *Sed interdum nisl pretium non. Mauris sodales consequat risus vel consectetur. Aliquam erat volutpat. Nunc sed sapien ligula. Proin faucibus sapien luctus nisl feugiat convallis faucibus elit cursus. Nunc vestibulum nunc ac massa pretium pharetra. Nulla facilisis turpis id augue venenatis blandit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.*

Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

4 Morbi eros magna

Morbi eros magna, vestibulum non posuere non, porta eu quam. Maecenas vitae orci risus, eget imperdiet mauris. Donec massa mauris, pellentesque vel lobortis eu, molestie ac turpis. Sed condimentum convallis dolor, a dignissim est ultrices eu. Donec consectetur volutpat eros, et ornare dui ultricies id. Vivamus eu augue eget dolor euismod ultrices et sit amet nisi. Vivamus malesuada leo ac leo ullamcorper tempor. Donec justo mi, tempor vitae aliquet non, faucibus eu lacus. Donec dictum gravida neque, non porta turpis imperdiet eget. Curabitur quis euismod ligula.

References

1 Edsger W. Dijkstra. Letters to the editor: go to statement considered harmful. *Commun. ACM*, 11(3):147–148, 1968. doi:10.1145/362929.362947.

2 Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

3 John E. Hopcroft, Wolfgang J. Paul, and Leslie G. Valiant. On time versus space and related problems. In *16th Annual Symposium on Foundations of Computer Science, Berkeley, California, USA, October 13-15, 1975*, pages 57–64. IEEE Computer Society, 1975. `doi:10.1109/SFCS.1975.23`.

4 Donald E. Knuth. Computer Programming as an Art. *Commun. ACM*, 17(12):667–673, 1974. `doi:10.1145/361604.361612`.

A Styles of lists, enumerations, and descriptions

List of different predefined enumeration styles:

- `\begin{itemize}...\end{itemize}`
- ...
- ...

1. `\begin{enumerate}...\end{enumerate}`
2. ...
3. ...

(a) `\begin{alphaenumerate}...\end{alphaenumerate}`
(b) ...
(c) ...

(i) `\begin{romanenumerate}...\end{romanenumerate}`
(ii) ...
(iii) ...

(1) `\begin{bracketenumerate}...\end{bracketenumerate}`
(2) ...
(3) ...

Description 1 `\begin{description} \item[Description 1] ...\end{description}`
Description 2 Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.
Description 3 ...

Proposition 12 and Proposition 12 ...

B Theorem-like environments

List of different predefined enumeration styles:

▶ **Theorem 9.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Lemma 10.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Corollary 11.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Proposition 12.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Conjecture 13.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Observation 14.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Exercise 15.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Definition 16.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

► **Example 17.** Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

► Note 18. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

► Note. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

► Remark 19. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

► Remark. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▷ Claim 20. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▷ Claim. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

Proof. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque. ◄

Proof. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque. ◁