Active Learning of Automata for JSON-Streaming Validation

Highlights of Logic, Games, and Automata

Véronique Bruyère Guillermo A. Pérez Gaëtan Staquet

Theoretical computer science
Computer Science Department
Science Faculty
University of Mons

Formal Techniques in Software Engineering
Computer Science Department
Science Faculty
University of Antwerp

June 29, 2022

UMONS Université de Mons

fnrs LA LIBERTÉ DE CHERCHER

University of Antwerp

What is a JSON document?

```
{
    "day": 29,
    "place": {
        "city": "Paris",
        "country": "France"
    },
    "authors": ["Bruyère", "Pérez", "Staquet"]
}
```

What is a JSON document?

```
{
    "day": 29,
    "place": {
        "city": "Paris",
        "country": "France"
    },
    "authors": ["Bruyère", "Pérez", "Staquet"]
}
```

- An object is an unordered collection of key-value pairs.

What is a JSON document?

```
{
    "day": 29,
    "place": {
        "city": "Paris",
        "country": "France"
    },
    "authors": ["Bruyère", "Pérez", "Staquet"]
}
```

- An object is an unordered collection of key-value pairs.
- An array is an ordered collection of values.

Adding constraints to a document

We want a document to satisfy some constraints, given in a JSON schema.

Adding constraints to a document

We want a document to satisfy some constraints, given in a JSON schema.

We want to decide whether a document d satisfies the schema S. Simple idea: explore d and S in parallel and check that each value in d is correct for S.

Adding constraints to a document

We want a document to satisfy some constraints, given in a JSON schema.

We want to decide whether a document d satisfies the schema S.
Simple idea: explore d and S in parallel and check that each value in d is correct for S.

What if S contains Boolean operations?
$\hookrightarrow$ We may need to backtrack and process the same value of d multiple times.

Adding constraints to a document

We want a document to satisfy some constraints, given in a JSON schema.

We want to decide whether a document d satisfies the schema S. Simple idea: explore d and S in parallel and check that each value in d is correct for S.

What if S contains Boolean operations?
$\hookrightarrow$ We may need to backtrack and process the same value of d multiple times.

What if we are in a streaming context, where we receive d little by little and we do not want to store the whole document?

Adding constraints to a document

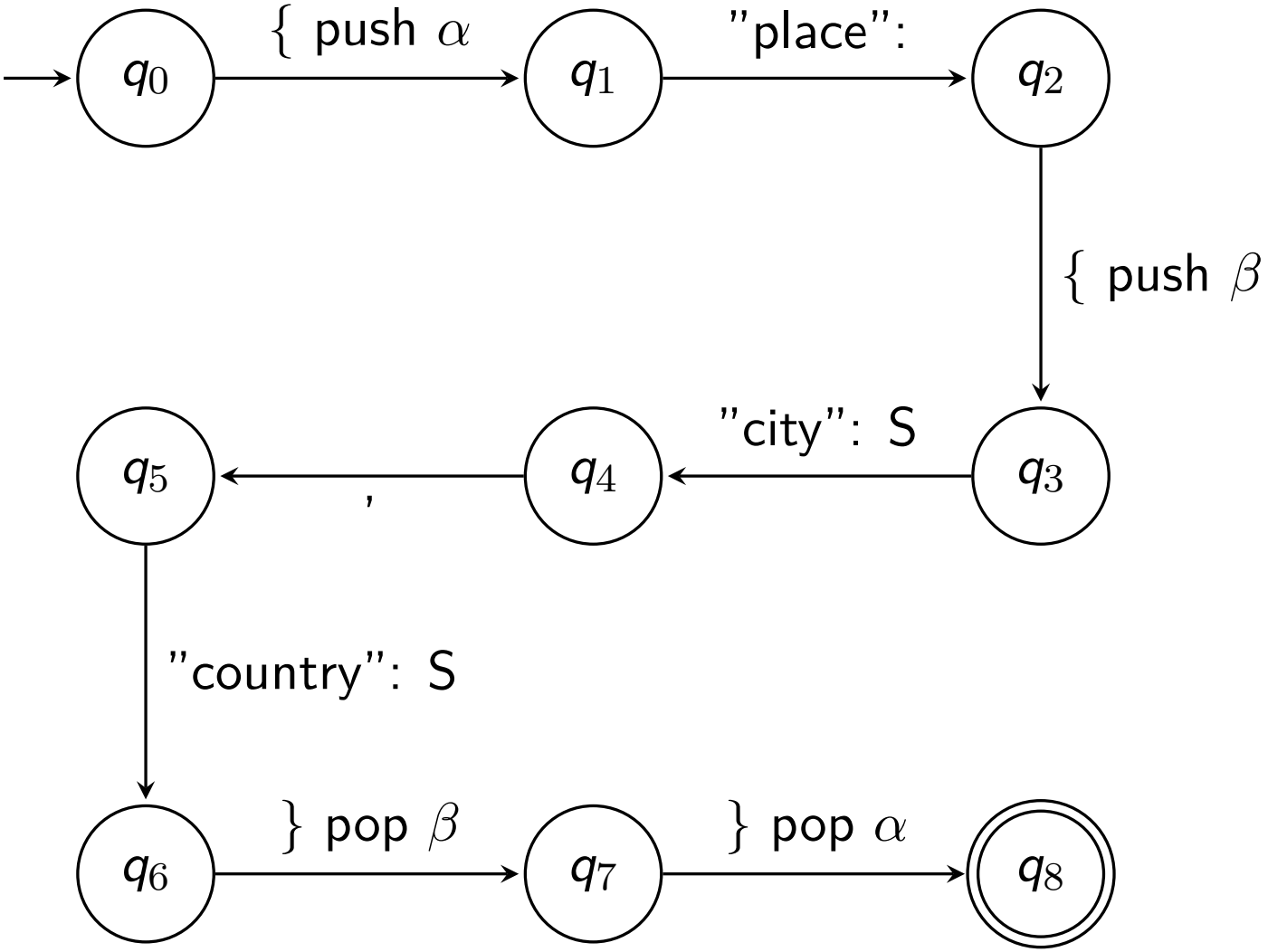
We want a document to satisfy some constraints, given in a JSON schema.

We want to decide whether a document d satisfies the schema S.
Simple idea: explore d and S in parallel and check that each value in d is correct for S.

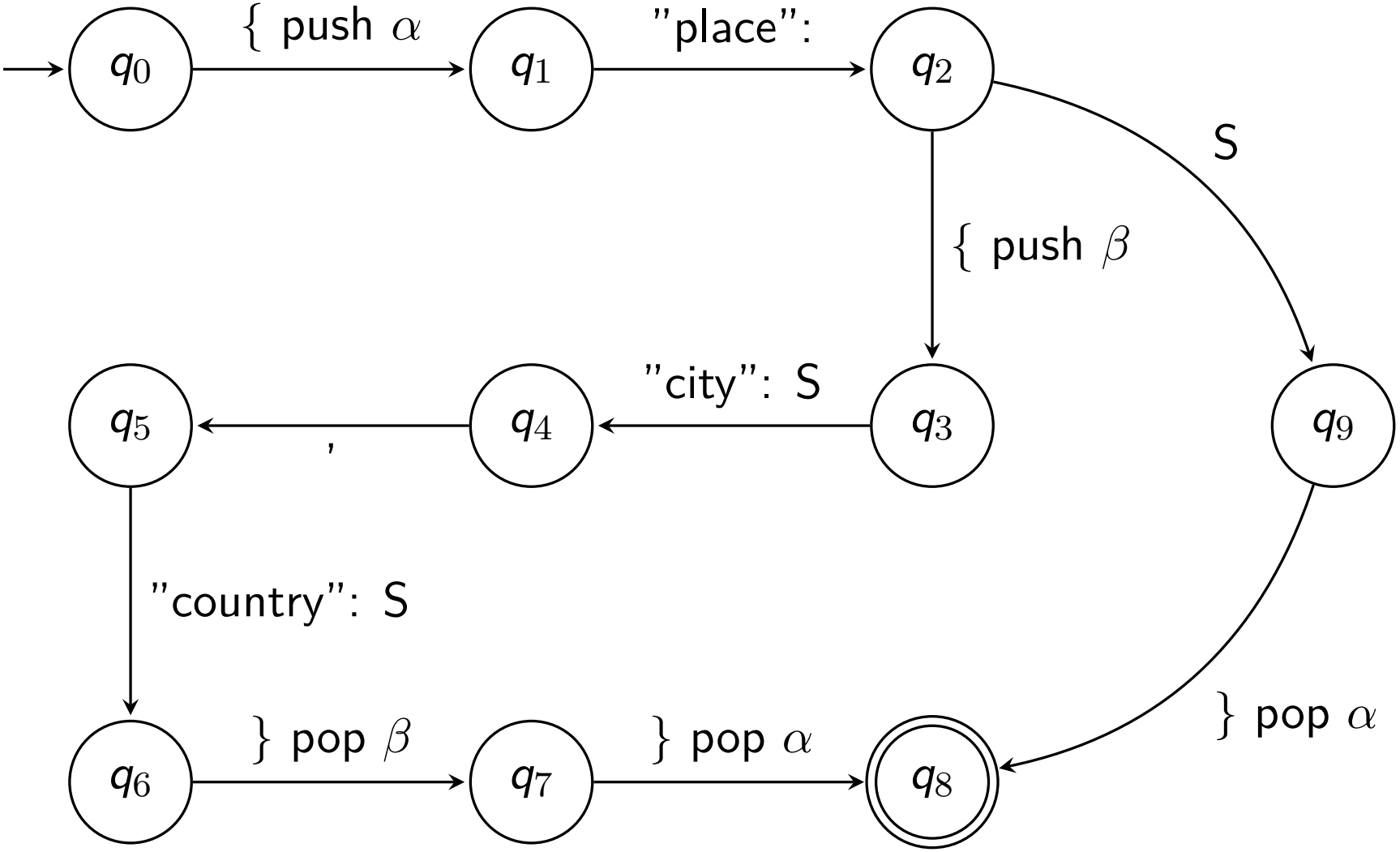
What if S contains Boolean operations?
$\hookrightarrow$ We may need to backtrack and process the same value of d multiple times.

What if we are in a streaming context, where we receive d little by little and we do not want to store the whole document?
$\hookrightarrow$ Use pushdown automata-based techniques!

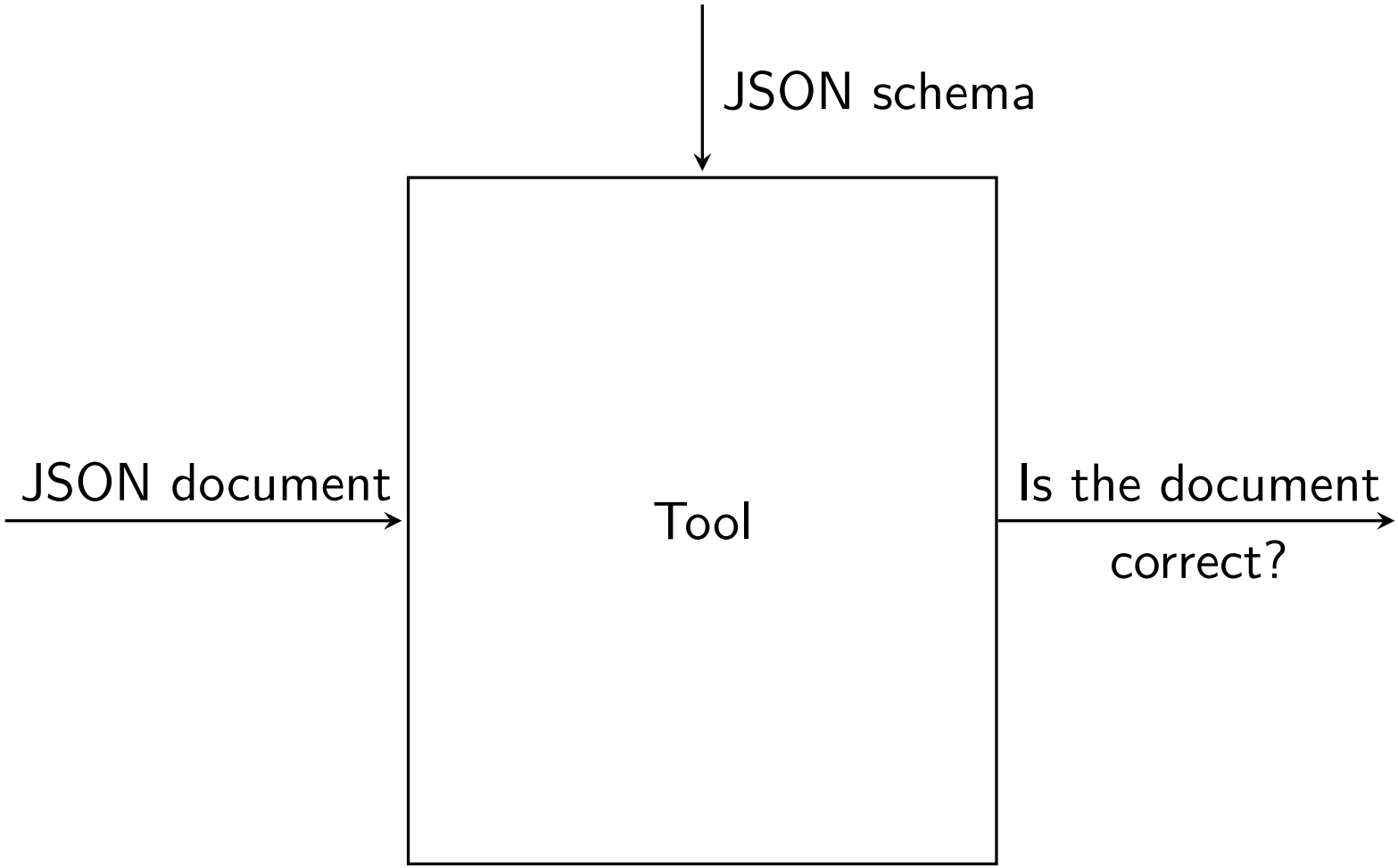
(Visibly) pushdown automaton



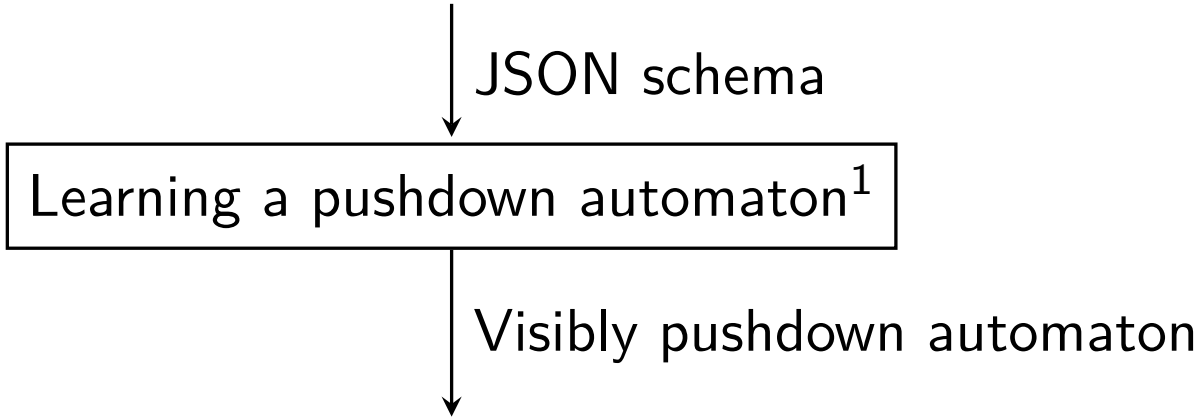
(Visibly) pushdown automaton



Our framework

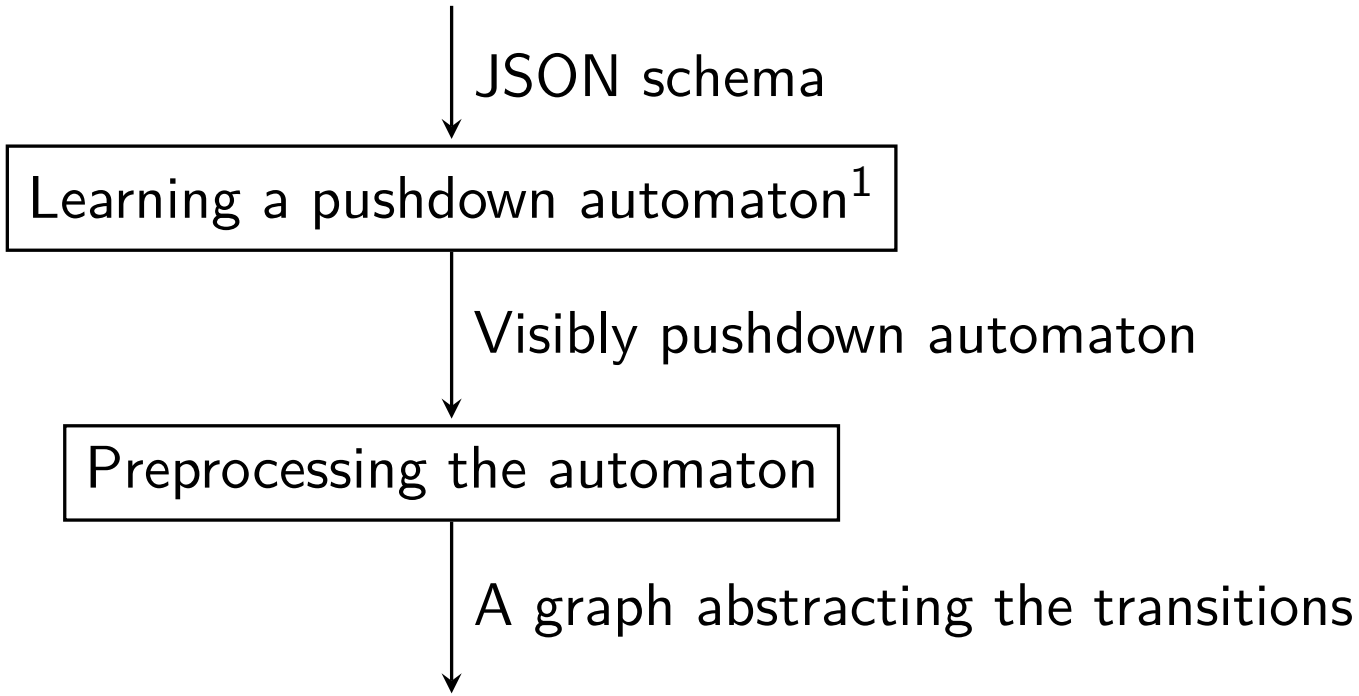


Our framework



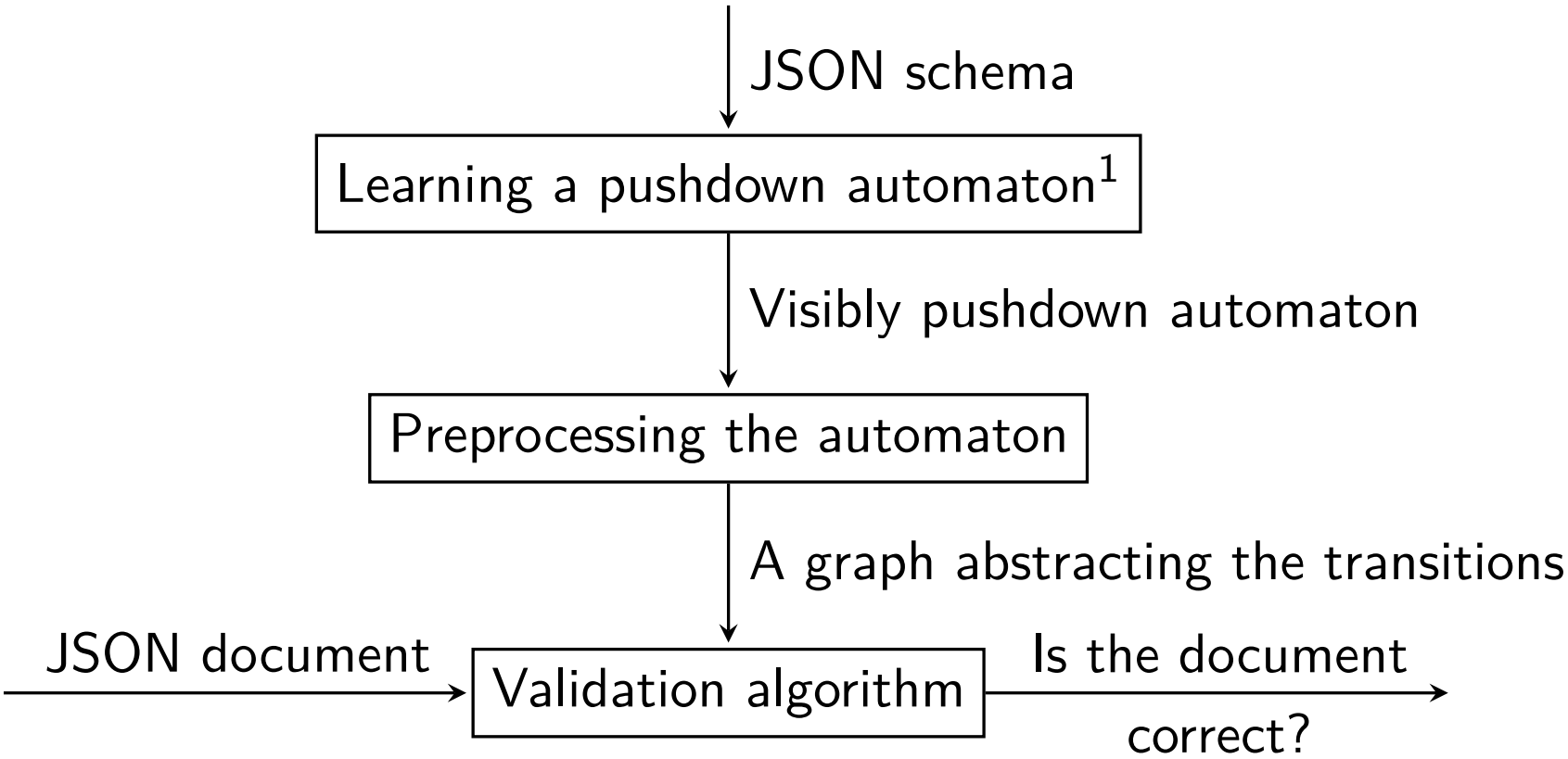
[1] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

Our framework



[1] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

Our framework



[1] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

Main contributions

1. For any schema S, we proved there always exists a visibly pushdown automaton that can decide whether a document satisfies S.
   $\hookrightarrow$ But fixing an order on the key-value pairs yields a much smaller automaton.

[2] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

Main contributions

1. For any schema S, we proved there always exists a visibly pushdown automaton that can decide whether a document satisfies S.
   $\hookrightarrow$ But fixing an order on the key-value pairs yields a much smaller automaton.
2. An algorithm using TTT to actively learn a visibly pushdown automaton[2] from a schema.

[2] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

Main contributions

1. For any schema S, we proved there always exists a visibly pushdown automaton that can decide whether a document satisfies S.
   $\hookrightarrow$ But fixing an order on the key-value pairs yields a much smaller automaton.
2. An algorithm using TTT to actively learn a visibly pushdown automaton[2] from a schema.
3. A new validation algorithm that does not need to backtrack nor to store the whole document.
   Idea: create a graph to abstract the transitions inside an object, and use a stack to store the needed information.

[2] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

Main contributions

1. For any schema S, we proved there always exists a visibly pushdown automaton that can decide whether a document satisfies S.
   $\hookrightarrow$ But fixing an order on the key-value pairs yields a much smaller automaton.
2. An algorithm using TTT to actively learn a visibly pushdown automaton[2] from a schema.
3. A new validation algorithm that does not need to backtrack nor to store the whole document.
   Idea: create a graph to abstract the transitions inside an object, and use a stack to store the needed information.

$\hookrightarrow$ Work in progress.

[2] Isberner, "Foundations of active automata learning: an algorithmic perspective", 2015.

References I

 Isberner, Malte. “Foundations of active automata learning: an algorithmic perspective”. PhD thesis. Technical University Dortmund, Germany, 2015. URL: http://hdl.handle.net/2003/34282.