securityandtrust.lu

SPiCE Workshop: Team Ping!

Flexible State-Merging with Python

10th October 2016

Chris Hammerschmidt, `firstname.lastname@uni.lu`

Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

Context for our Participation

Core Assumptions

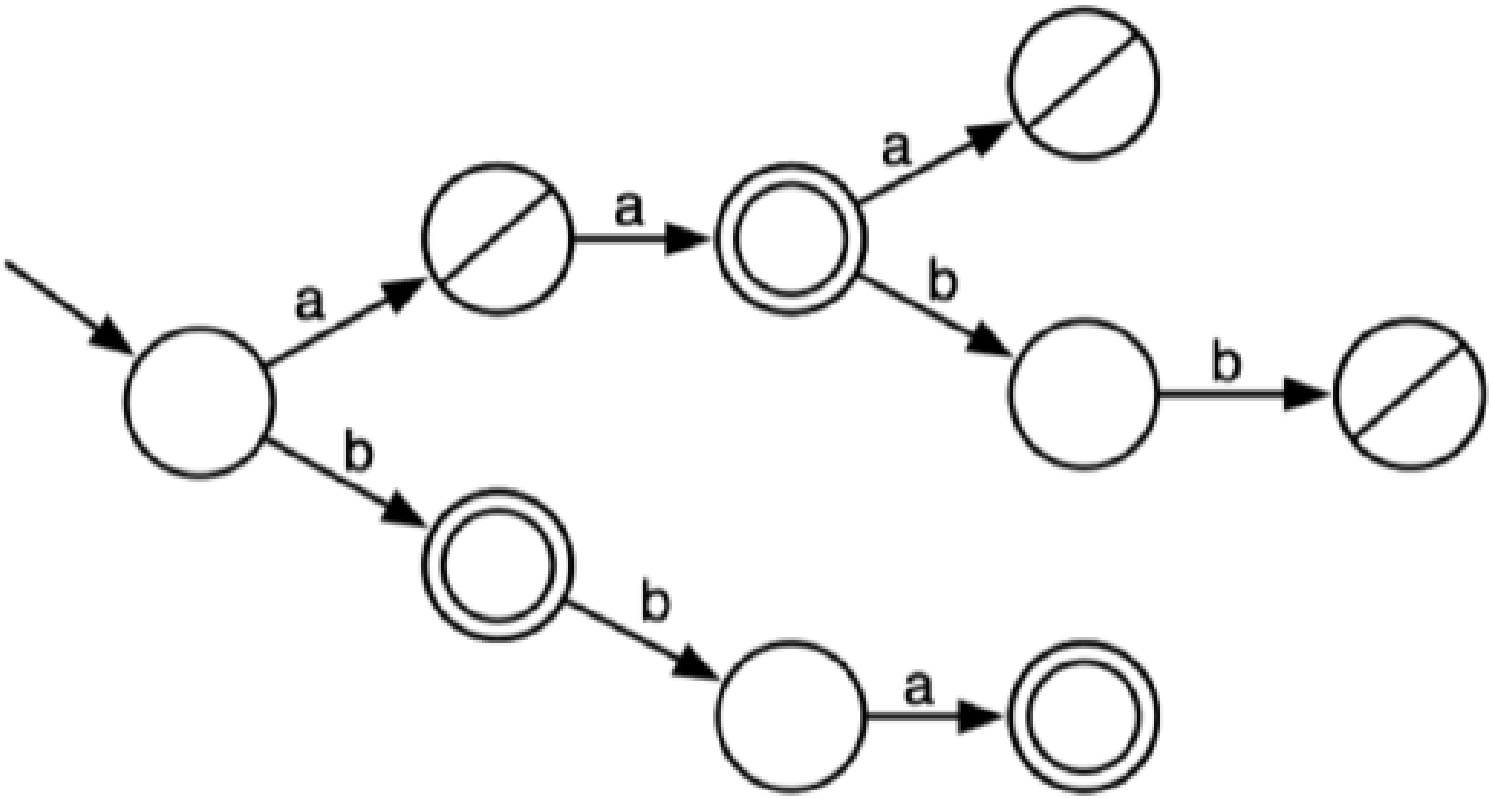
securityandtrust.lu

We live in a deterministic world where everything is regular

We assume: Everything is generated by a PDFA.

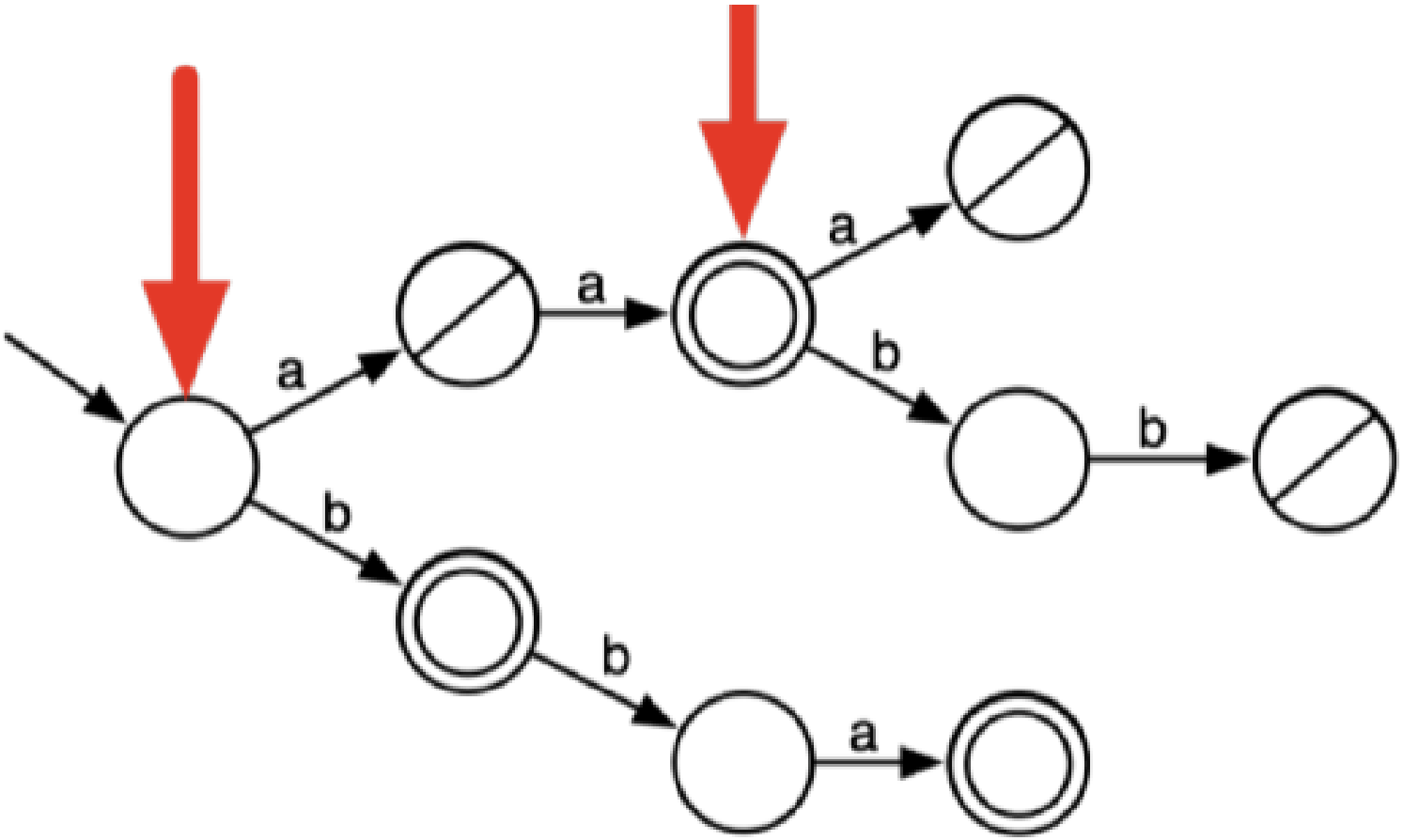
State-Merging for PDFA I

Reminder: Merging Conceptually



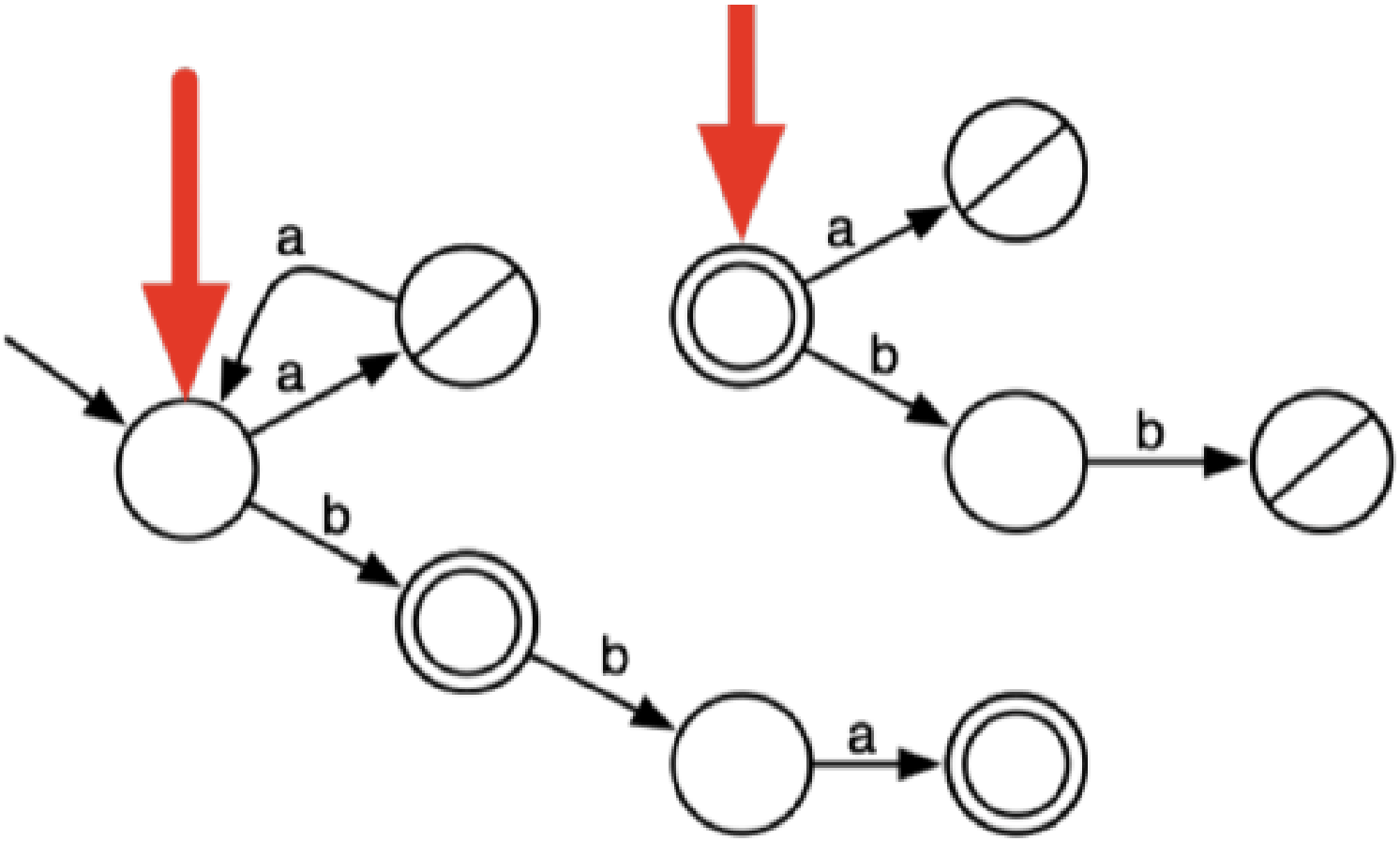
State-Merging for PDFA II

Reminder: Merging Conceptually



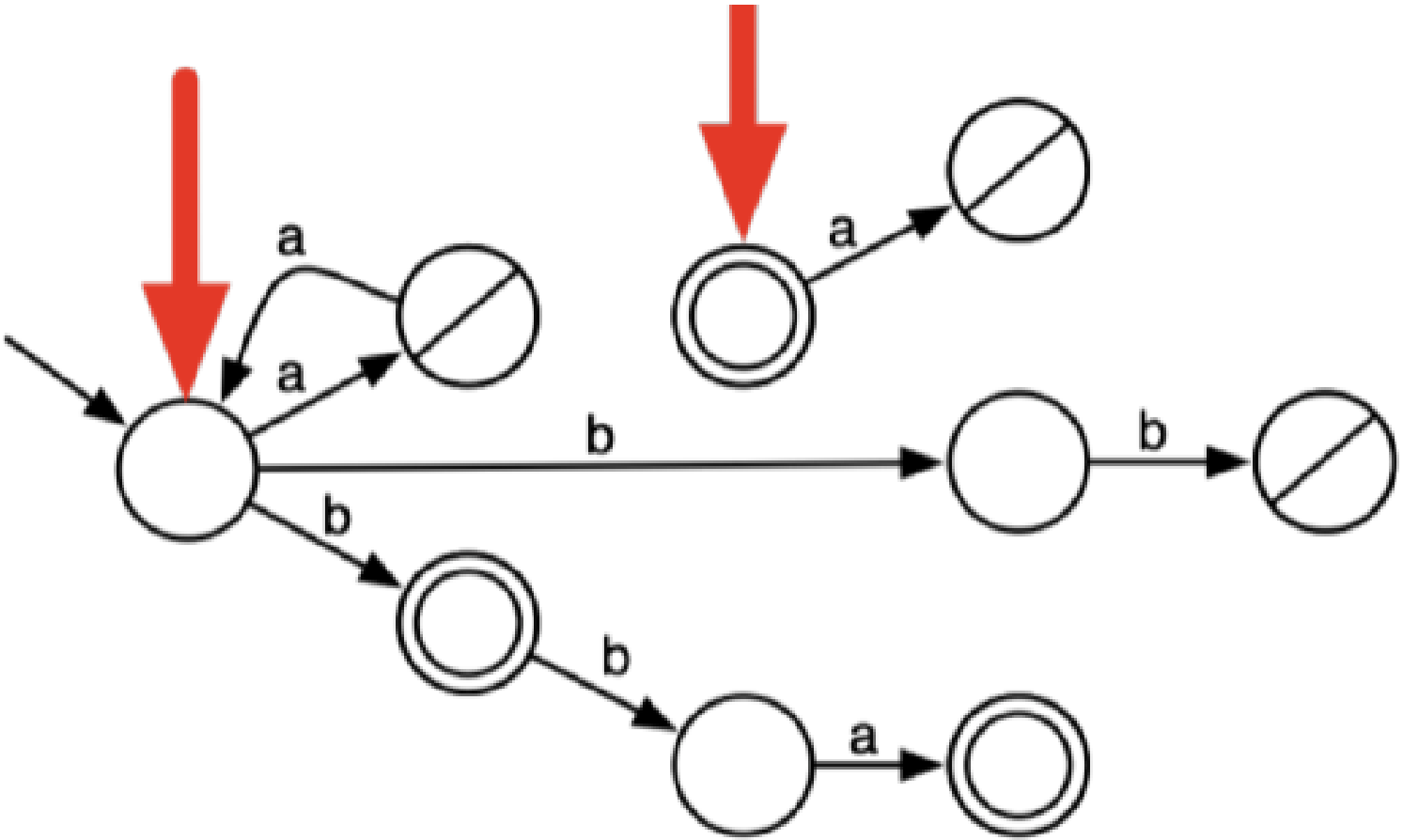
State-Merging for PDFA III

Reminder: Merging Conceptually



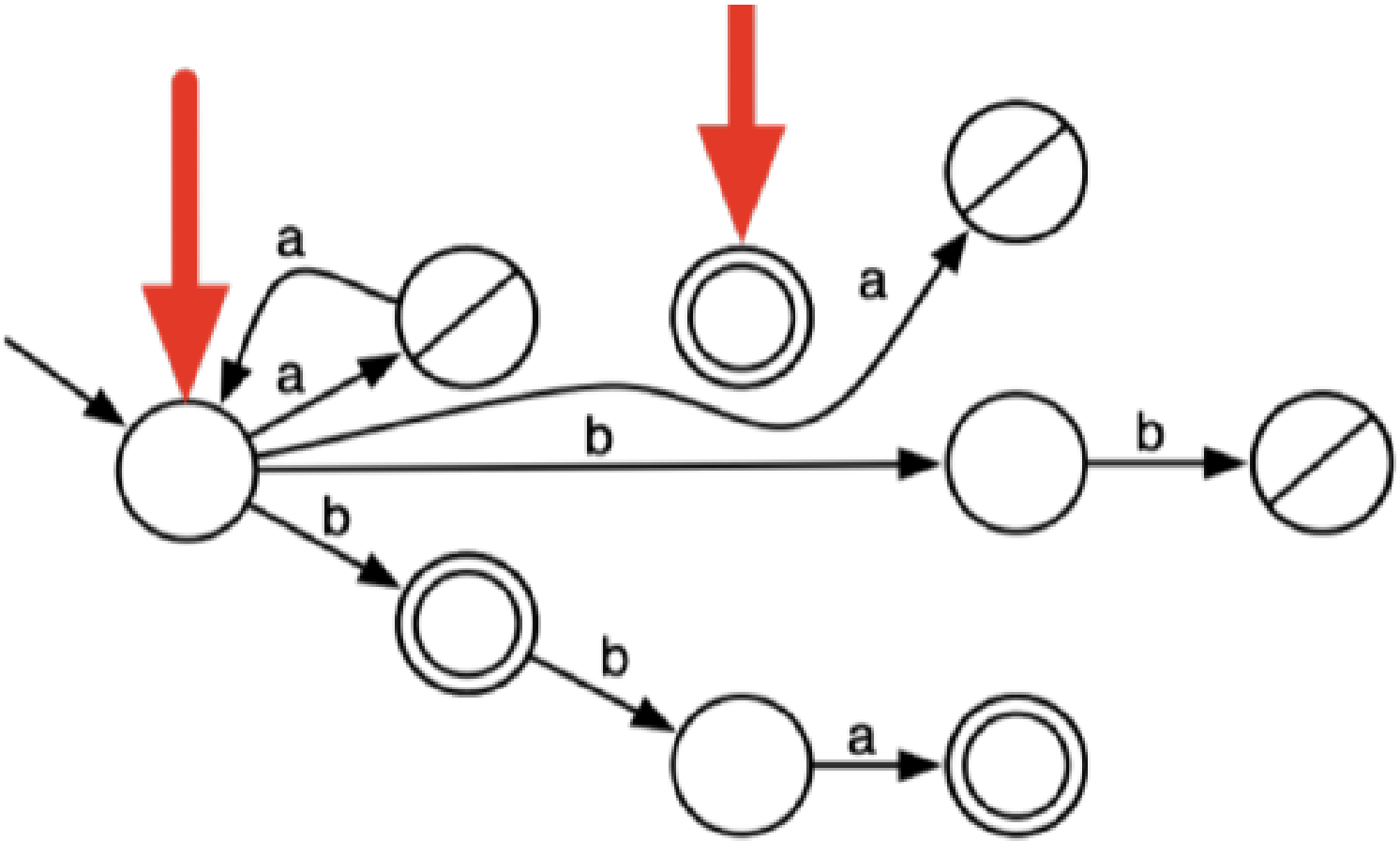
State-Merging for PDFA IV

Reminder: Merging Conceptually



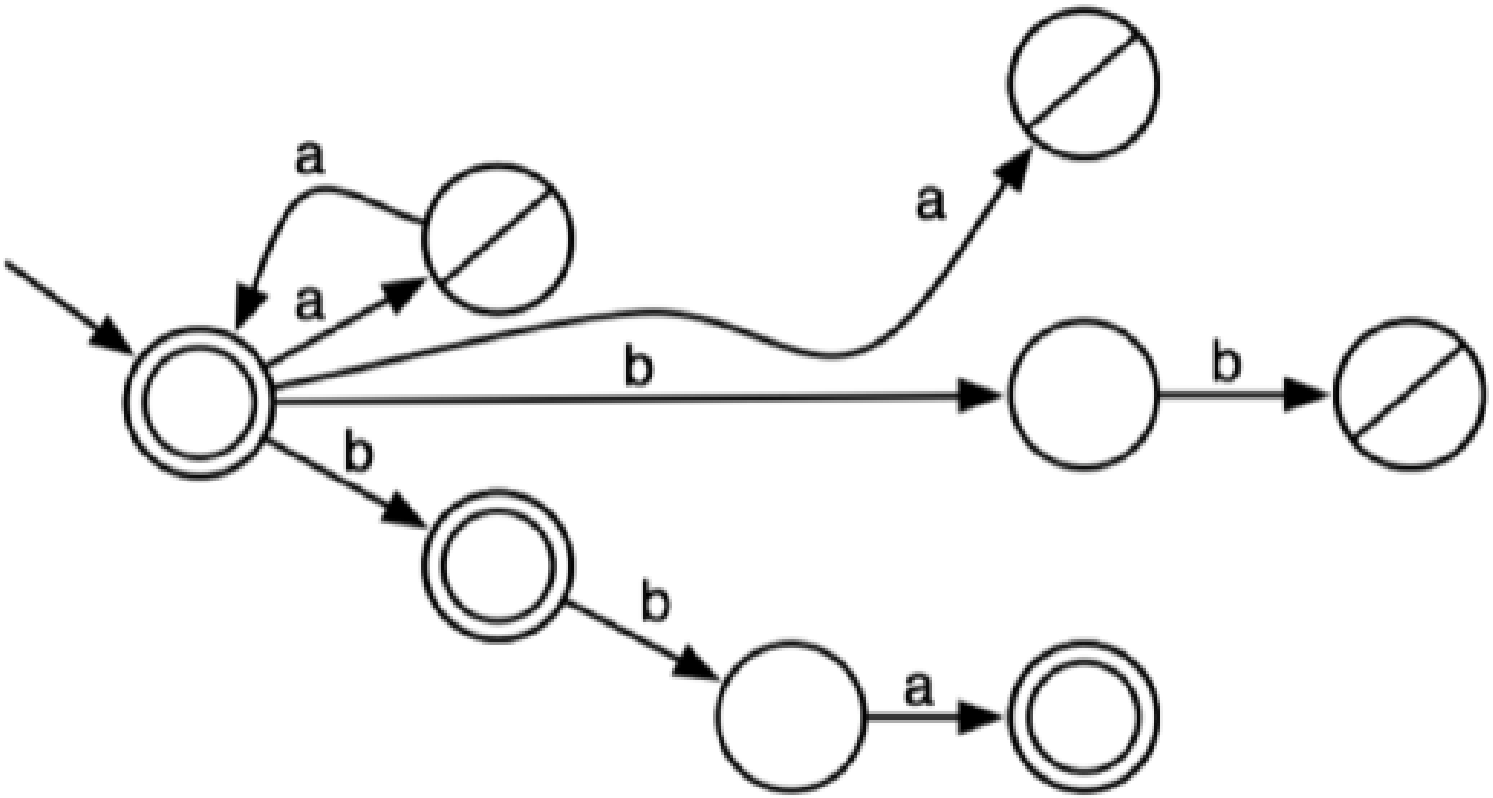
State-Merging for PDFA V

Reminder: Merging Conceptually



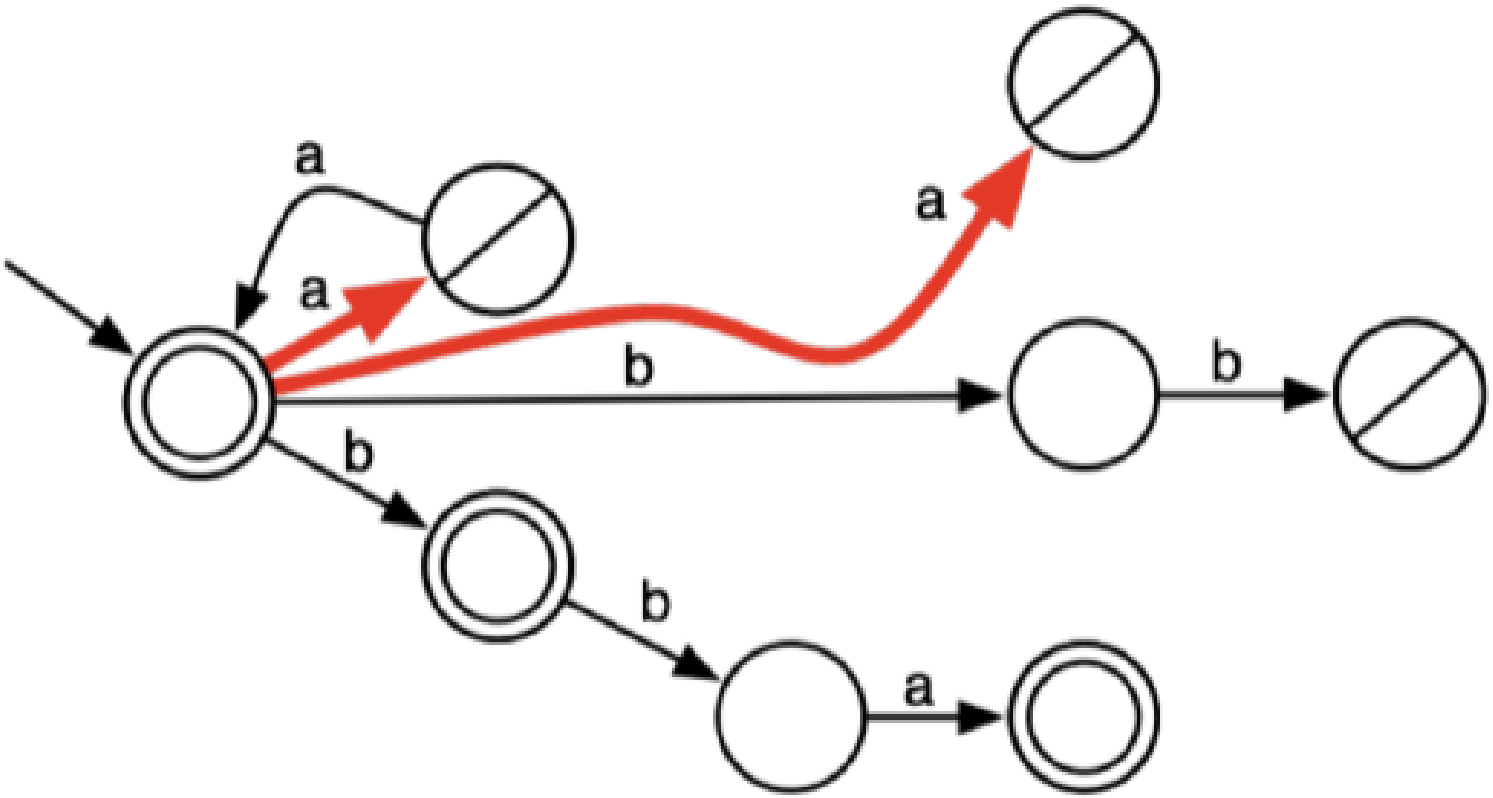
State-Merging for PDFA VI

Reminder: Merging Conceptually



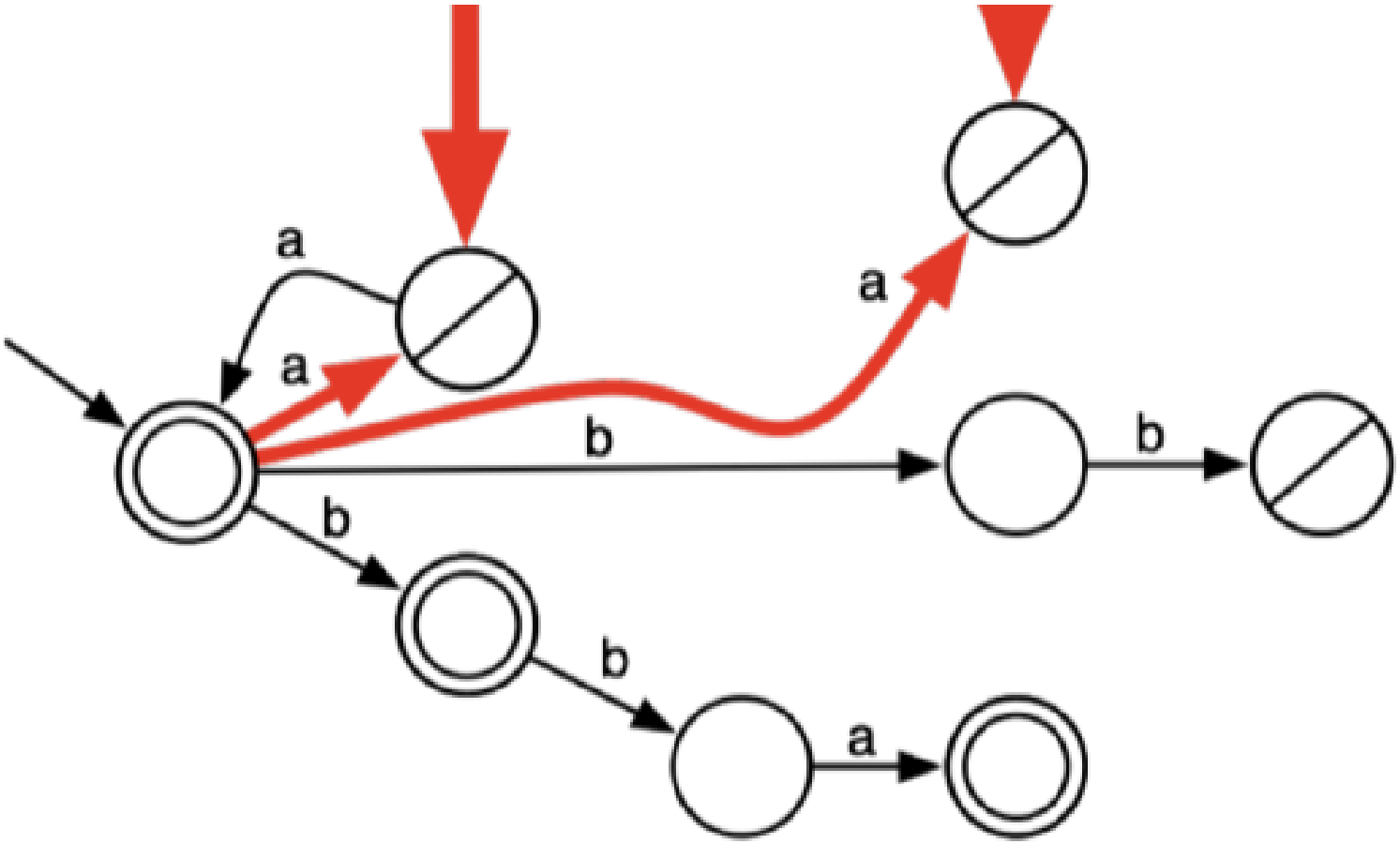
State-Merging for PDFA VII

Reminder: Merging Conceptually



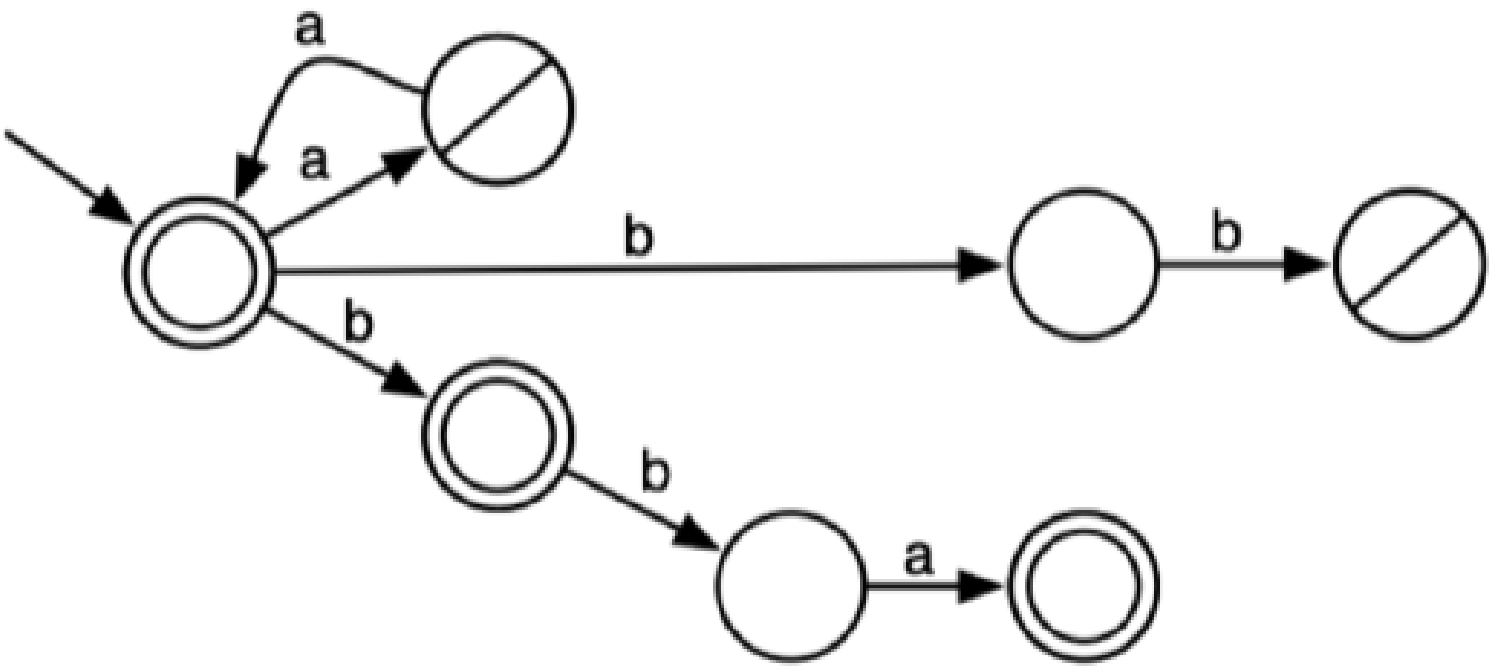
State-Merging for PDFA VIII

Reminder: Merging Conceptually



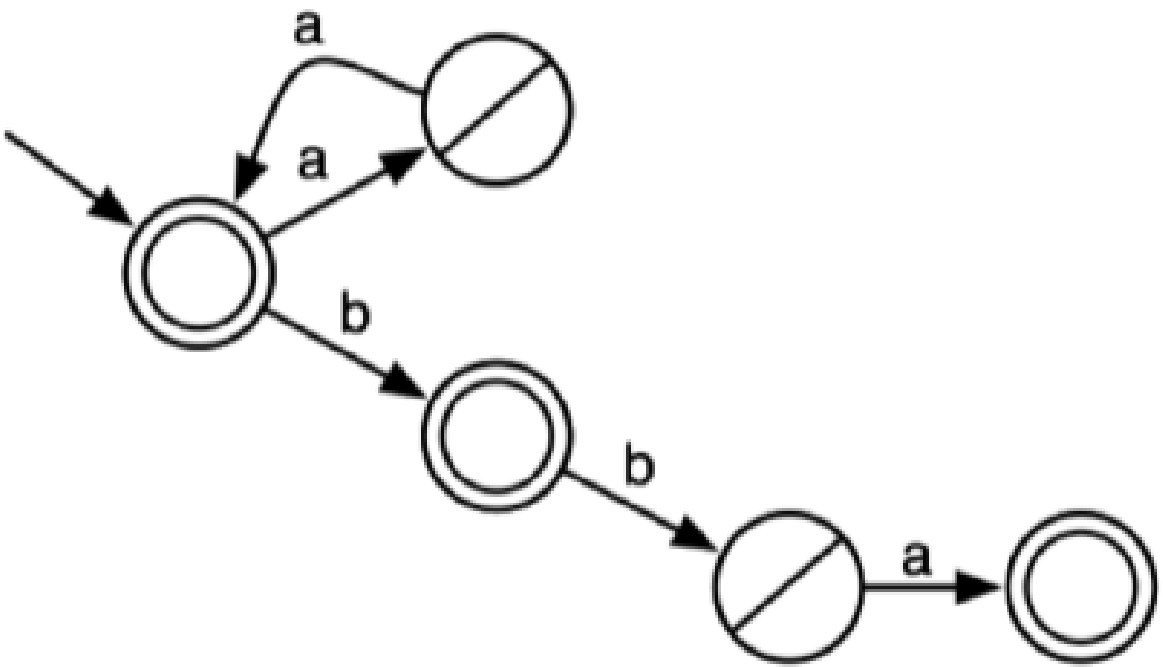
State-Merging for PDFA IX

Reminder: Merging Conceptually



State-Merging for PDFA X

Reminder: Merging Conceptually



Animation graphics made by Sicco Verwer.

Our Results

Submitting baselines and our algorithms

securityandtrust.lu

Set	ALERGIA	3-gram	spectral	likelihood	overlap	Submitted	Score	Rank	Best
1	0.841	0.843	**0.874**	0.841	0.841	spectral	0.879	4	0.918
2	0.823	0.818	**0.872**	0.799	0.768	spectral	0.874	5	0.920
3	0.778	0.776	**0.828**	0.790	0.722	spectral	0.825	7	0.886
4	0.370	**0.538**	0.470	0.400	0.439	3-gram	0.528	5	0.608
5	**0.551**	0.527	0.361	/	/	$ALERGIA$	0.555	8	0.810
6	0.650	**0.675**	0.631	0.531	0.597	$ALERGIA_{50}$	0.724	7	0.860
7	0.337	**0.442**	0.367	/	/	3-gram	0.440	7	0.785
8	0.512	**0.591**	0.535	0.494	0.507	3-gram	0.597	4	0.657
9	**0.929**	0.859	0.818	0.867	0.847	$ALERGIA_{50}$	0.948	2	0.963
10	0.284	**0.413**	0.302	/	/	3-gram	0.396	7	0.552
11	0.301	**0.389**	0.379	/	/	3-gram	0.372	9	0.544
12	0.654	**0.700**	0.575	/	/	3-gram	0.699	7	0.811
13	0.299	**0.436**	0.372	/	/	4-gram	0.455	4	0.588
14	0.306	0.339	**0.359**	0.399	0.326	$ALERGIA_{100}$	0.379	7	0.465
15	**0.265**	0.251	0.221	/	/	spectral	0.279	5	0.298

Implementation and Algorithms

A Python library for state-merging

securityandtrust.lu

	GI-learning	LearnLib	libalf	GI toolbox
Algorithm strategies	Offline/Online	Online	Offline/Online	Offline
Noise tolerance	Yes	No	No	No
Parallel impl.	Yes	No	No	No
Efficiency	Very High	High	Medium	Low
Progr. language	C++	Java	C++, Java (JNI)	Matlab
Hierarchical structure	Yes	Yes	Yes	No
Code documentation	Yes	Yes	No	Yes

Taken from GI-learning: an optimized framework for grammatical inference by P. Cottone, M. Ortolani, G. Pergola in Proceedings of the 17th International Conference on Computer Systems and Technologies

Piggyback on `scikit-learn` with our existing Tool: dfasat

securityandtrust.lu

SVM Estimator in sklearn

```
from sklearn import svm
# get training samples and labels
X_samples, Y_labels = get_data()
# initialize classifier
clf = svm.SVC(gamma=0.001, C=100)

# learn and predict
clf.fit(X_samples, Y_labels)
clf.predict(sequence)
```

Piggyback on `scikit-learn` with our existing Tool: dfasat

securityandtrust.lu

SVM Estimator in sklearn

```
from sklearn import svm
# get training samples and labels
X_samples, Y_labels = get_data()
# initialize classifier
clf = svm.SVC(gamma=0.001, C=100)

# learn and predict
clf.fit(X_samples, Y_labels)
clf.predict(sequence)
```

DFASAT Estimator in sklearn

```
from dfasat import DFASATEstimator
# get training samples and labels
X_samples, Y_labels = get_data()
# initialize classifier
estimator = DFASATEstimator(hName="alergia",
                            hData="alergia_data",
                            tries=1, state_count=25)
# learn and predict
estimator.fit(X_samples, Y_labels)
estimator.predict(sequence)
```

Compatibility with the Rest of the World!

As well as piggybacking on `scikit-learn`s' features

securityandtrust.lu

Added bonus: scikit-learn infrastructure:

- cross-validation
- ensembles
- grid-search
- ...

Flexible State-Merging

Why we care about our implementation.

securityandtrust.lu

- GI is often a heuristic process, intends to recover/converge to a target
- what to do if there is no clear target?
- what to do if we have extra information from an application field?

Our approach: Change the heuristic!

- Use case: Distilling/privileged data
- $wind^{speed^{prediction}}$, protocol reverse engineering

E.g. learn from a tuple $< a, b >, a \in A, b \in B$ and only classify/predict only from $a \in A$.

Flexible State-Merging

Why we care about our implementation.

securityandtrust.lu

- GI is often a heuristic process, intends to recover/converge to a target
- what to do if there is no clear target?
- what to do if we have extra information from an application field?

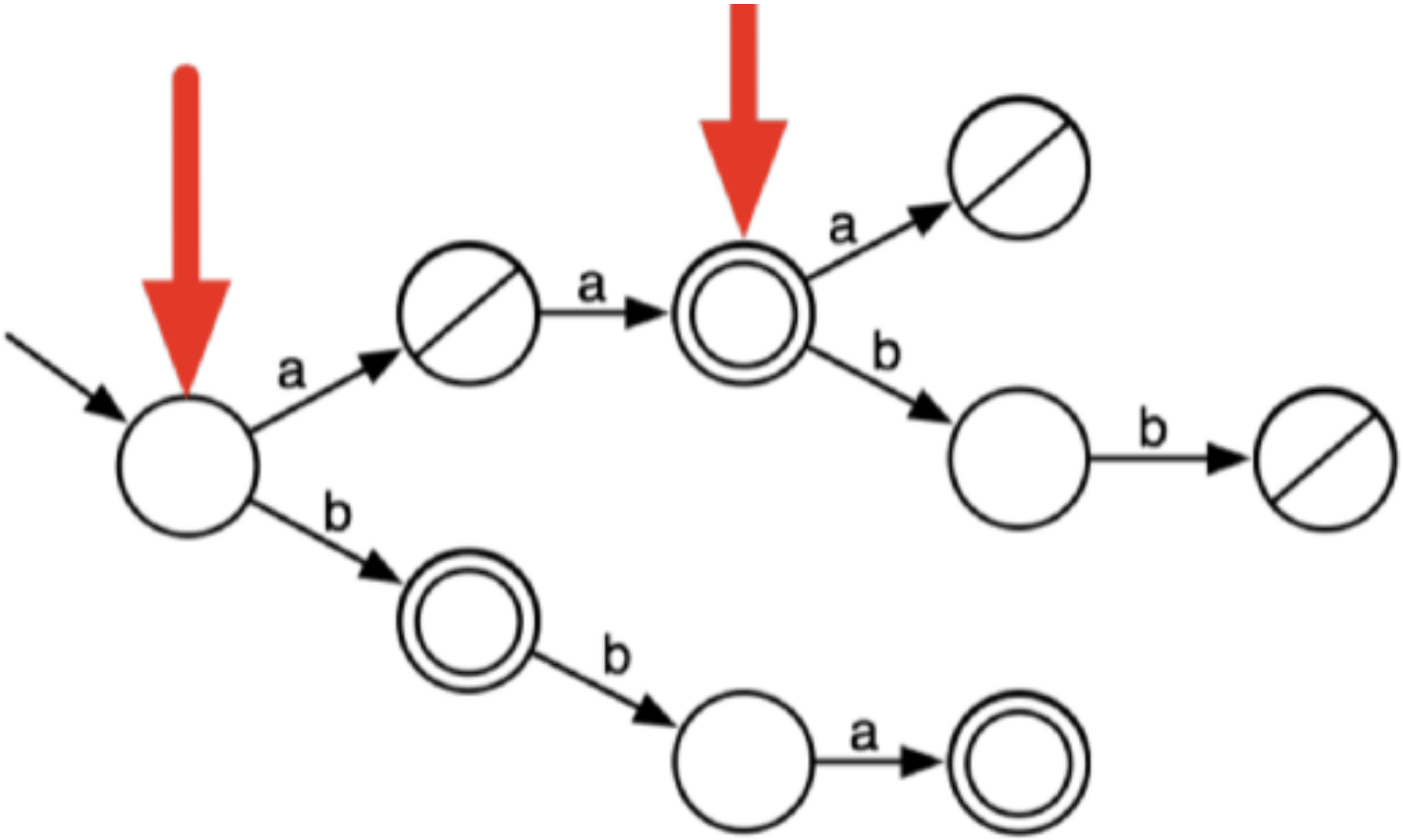
Our approach: Change the heuristic!

- Use case: Distilling/privileged data
- $wind^{speed^{prediction}}$, protocol reverse engineering

E.g. learn from a tuple $< a, b >, a \in A, b \in B$ and only classify/predict only from $a \in A$.

Flexible State-Merging

What do we mean by it?



Consistency check, score calculation, summary statistic collection on a a tree of node objects managed by a state-merger.

Flexible State-Merging

How does it work?

securityandtrust.lu

```
70  class evaluation_function  {
71
72  protected:
73    static DerivedRegister<evaluation_function> reg;
74
75  public:
76
77  /* Global data */
103  * huge influence on performance, needs to be simple */
104    virtual bool consistent(state_merger*, apta_node* left, apta_node* right);
105    virtual void update_score(state_merger*, apta_node* left, apta_node* right);
106    virtual void undo_update(state_merger*, apta_node* left, apta_node* right);
107
108  /* Called when testing a merge
109  * compute the score and consistency of a merge, and reset global counters/structures
110  *
111  * influence on performance, needs to be somewhat simple */
112    virtual bool compute_consistency(state_merger *, apta_node* left, apta_node* right);
113    virtual int  compute_score(state_merger *, apta_node* left, apta_node* right);
114    virtual void reset(state_merger *);
115
```

Plug and play evaluation functions.

Flexible State-Merging

Example: Evidence Driven Heuristic

securityandtrust.lu

```
61  bool count_driven::consistent(state_merger *merger, apta_node* left, apta_node* right){
62      if(inconsistency_found) return false;
63
64      count_data* l = (count_data*)left->data;
65      count_data* r = (count_data*)right->data;
66
67      if(l->num_accepting != 0 && r->num_rejecting != 0){ inconsistency_found = true; retur
68      if(l->num_rejecting != 0 && r->num_accepting != 0){ inconsistency_found = true; retur
69
70      return true;
71  };
72
73  void count_driven::update_score(state_merger *merger, apta_node* left, apta_node* right){
74    num_merges += 1;
75  };
76
77  int count_driven::compute_score(state_merger *merger, apta_node* left, apta_node* right){
78    return num_merges;
79  };
```

Flexible State-Merging

Example: Evidence Driven Heuristic

securityandtrust.lu

```
17  /* Evidence driven state merging, count number of pos-pos and neg-neg merges */
18  void evidence_driven::update_score(state_merger *merger, apta_node* left, apta_node* rig
19      edsm_data* l = (edsm_data*) left->data;
20      edsm_data* r = (edsm_data*) right->data;
21
22      if(l->num_accepting > 0 && r->num_accepting > 0) num_pos += 1;
23      if(l->num_rejecting > 0 && r->num_rejecting > 0) num_neg += 1;
24  };
25
26  int evidence_driven::compute_score(state_merger *merger, apta_node* left, apta_node* rig
27    return num_pos + num_neg;
28  };
29
30  void evidence_driven::reset(state_merger *merger){
31    inconsistency_found = false;
32    num_pos = 0;
33    num_neg = 0;
34  };
```

Flexible State-Merging

Example: Mealy Machine Heuristic

```
63  bool mealy::consistent(state_merger *merger, apta_node* left, apta_node* right){
64      if(inconsistency_found) return false;
65
66      mealy_data* l = reinterpret_cast<mealy_data*>(left->data);
67      mealy_data* r = reinterpret_cast<mealy_data*>(right->data);
68
69      int matched = 0;
70
71      for(output_map::iterator it = r->outputs.begin(); it != r->outputs.end(); ++it){
72          int input  = (*it).first;
73          int output = (*it).second;
74
75          if(l->outputs.find(input) != l->outputs.end()){
76              if(l->outputs[input] != output){
77                  inconsistency_found = true;
78                  return false;
79              }
80              matched = matched + 1;
81          }
82      }
83
84      num_unmatched = num_unmatched + (l->outputs.size() - matched);
85      num_unmatched = num_unmatched + (r->outputs.size() - matched);
86      num_matched   = num_matched   + matched;
87
88      return true;
89  };
```

Encapsulating C++ Code in Python

Lessons learned

securityandtrust.lu

It's very easy ...

... to shoot yourself in the foot.

Encapsulating C++ Code in Python

Lessons learned

securityandtrust.lu

It's very easy ...

... to shoot yourself in the foot.

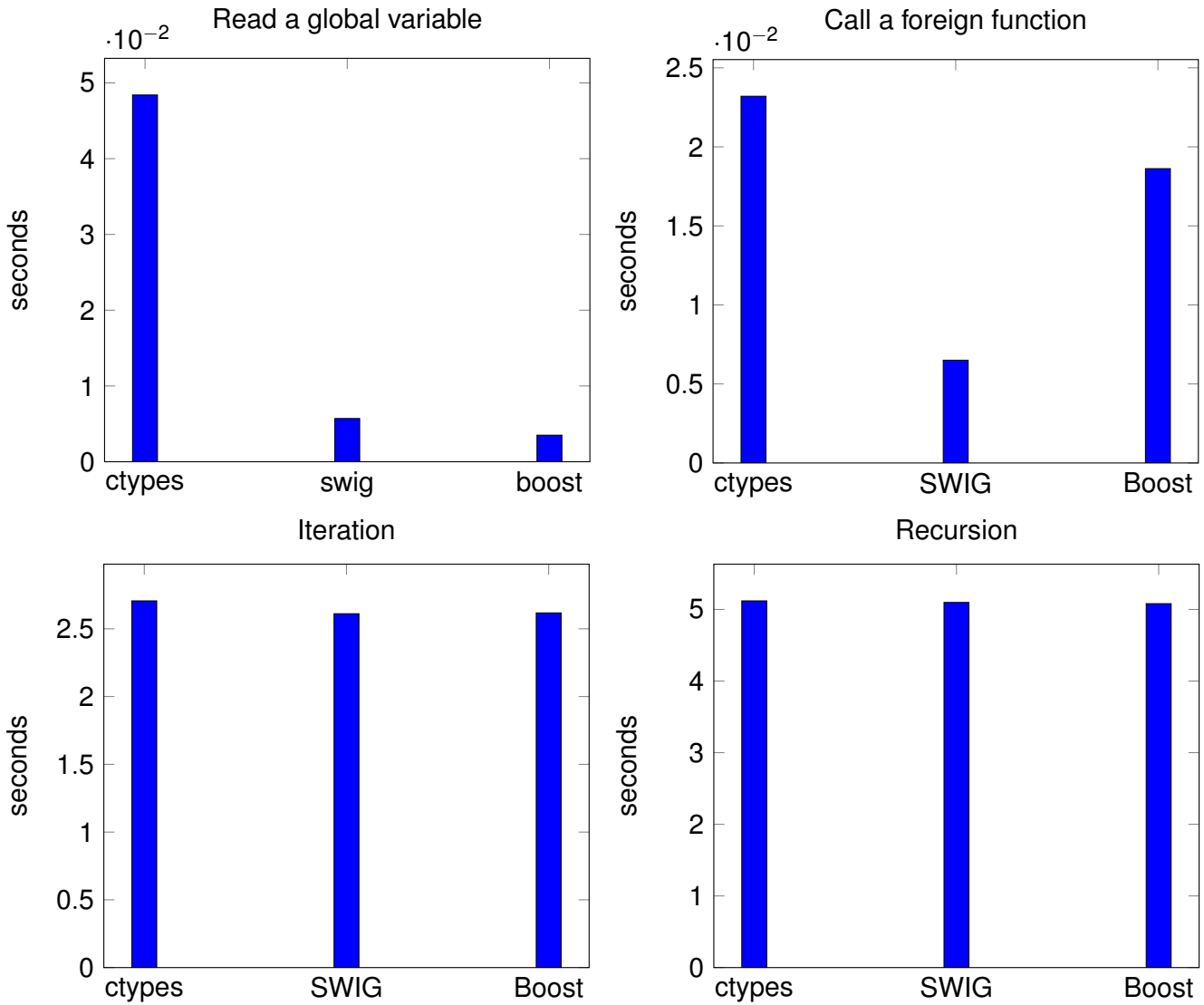
Evaluation of Wrappers

securityandtrust.lu

- Around a dozen wrappers using different methods
- How did we decide?
- Performance: our own benchmark suite
- Ease of use

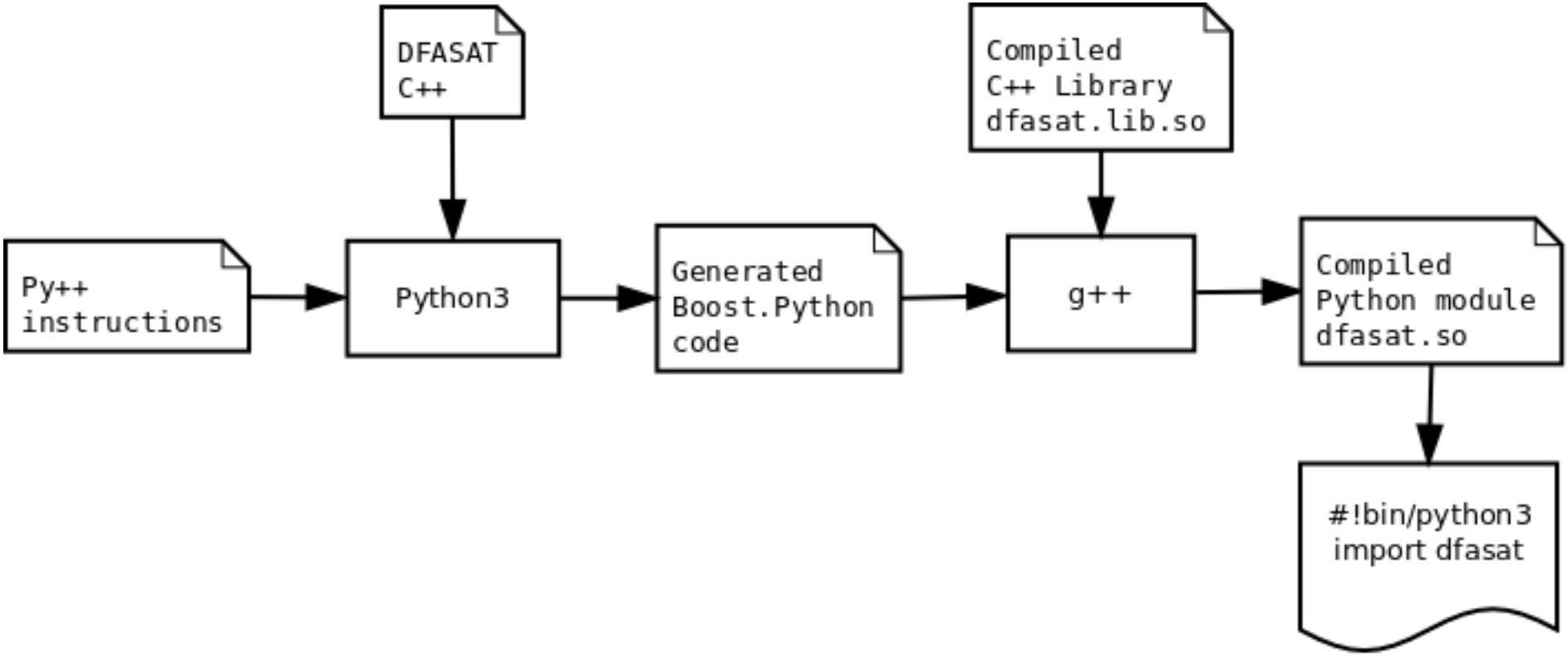
ctypes
Direct access to shared compiled libraries (gcc -shared -fPIC)
SWIG
Code generator, automatically creates bindings from C++ headers
Boost.Python
Interface library, explicit mappings, **works both ways**

Evaluation of Wrappers



Boost.Python I

Py++



Boost.Python I

The ultimate goal: rapid prototyping heuristics.

securityandtrust.lu

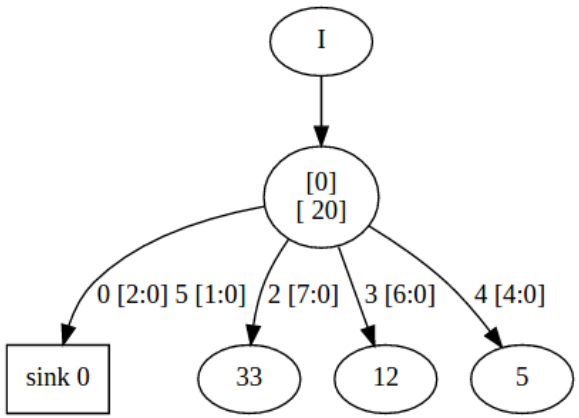
```
40  class evaluation(main.evaluation_function):
41    def __init__(self):
42      print('from python: evaluation function init')
43    def consistent(self, merger, left, right):
44      return True
```

Implementation

Jupyter notebooks

securityandtrust.lu

```
In [1]: from dfasat import DFASATEstimator

In [2]: est = DFASATEstimator(
            hName = 'alergia',
            hData = 'alergia_data',
            state_count  = 5,
            symbol_count = 5,
            parameter    = 0.5,
            tries = 1
        )

In [3]: est.fit_(dfa_file="train_toy.c.txt")

        dfa file set to train_toy.c.txt
        starting dfasat
        dfasat done

Out[3]: DFASATEstimator()

In [4]: est.plot()

Out[4]:
```




```
In [ ]:
```

Package Criticism

Shortcomings

securityandtrust.lu

- np.array input format is weird, as columns are not features
- we also provide ensemble methods as we don't have serialization yet and the student graduated

```
bag = BaggingClassifier(estimator=DFASATEstimator, n=50, output_file=file_prefix,
                        random_seed=True, random_counts=[5, 15, 25],
                        hData='alergia_data', hName='alergia',
                        symbol_count=5, # -y
                        state_count=5, # -t
                        parameter=0.5, # -p
                        method=1     # -m
                       )

fit = bag.fit(train_data_x, train_data_y, subset=True)
```

We should team up to create a sequence-prediction counterpart to scikit-learn.

Conclusion

securityandtrust.lu

- It's possible to benefit from the existing ecosystem, but it doesn't come for free.
- Flexible state-merging has a lot of unexplored use-cases.

Returning to the topic: SPiCE

- we were got at software traces, and really bad at NLP problems–no surprise
- we did ok on most synthetic datasets, but they were generated by a slightly more expressive model
- I think we can leverage grammatical information in our framework, but it requires some additional feature engineering

Conclusion

securityandtrust.lu

- It's possible to benefit from the existing ecosystem, but it doesn't come for free.
- Flexible state-merging has a lot of unexplored use-cases.

Returning to the topic: SPiCE

- we were got at software traces, and really bad at NLP problems–no surprise
- we did ok on most synthetic datasets, but they were generated by a slightly more expressive model
- I think we can leverage grammatical information in our framework, but it requires some additional feature engineering

Thank You!

Also to Sicco Verwer, Benjamin Loos[1], and the team.

Time for questions.

[1] and supervisors Thomas Engel and Radu State

Remarks I

securityandtrust.lu

`dfasat` is currently on the python package index, and can be installed via

```
pip install dfasat
```

although it's *not quite production-ready*.

Talk to me about your needs in a tool.