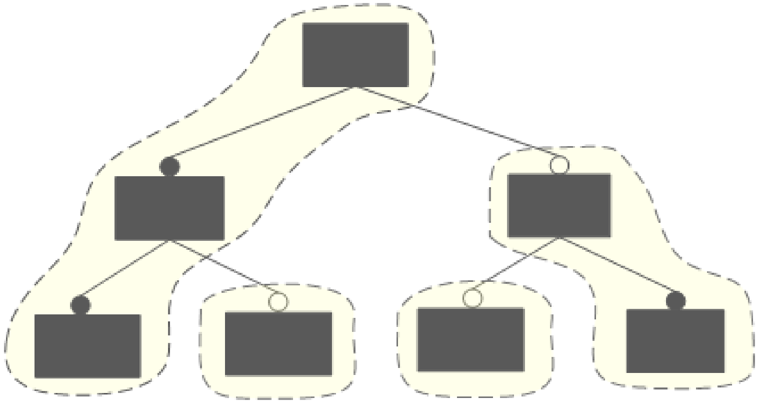
Automated Analysis of Feature Models using Atomic Sets



Sergio Segura

First Workshop on Analyses of Software Product Lines
(ASPL'08)

Motivation

Problem

Contribution

Conclusions

Motivation

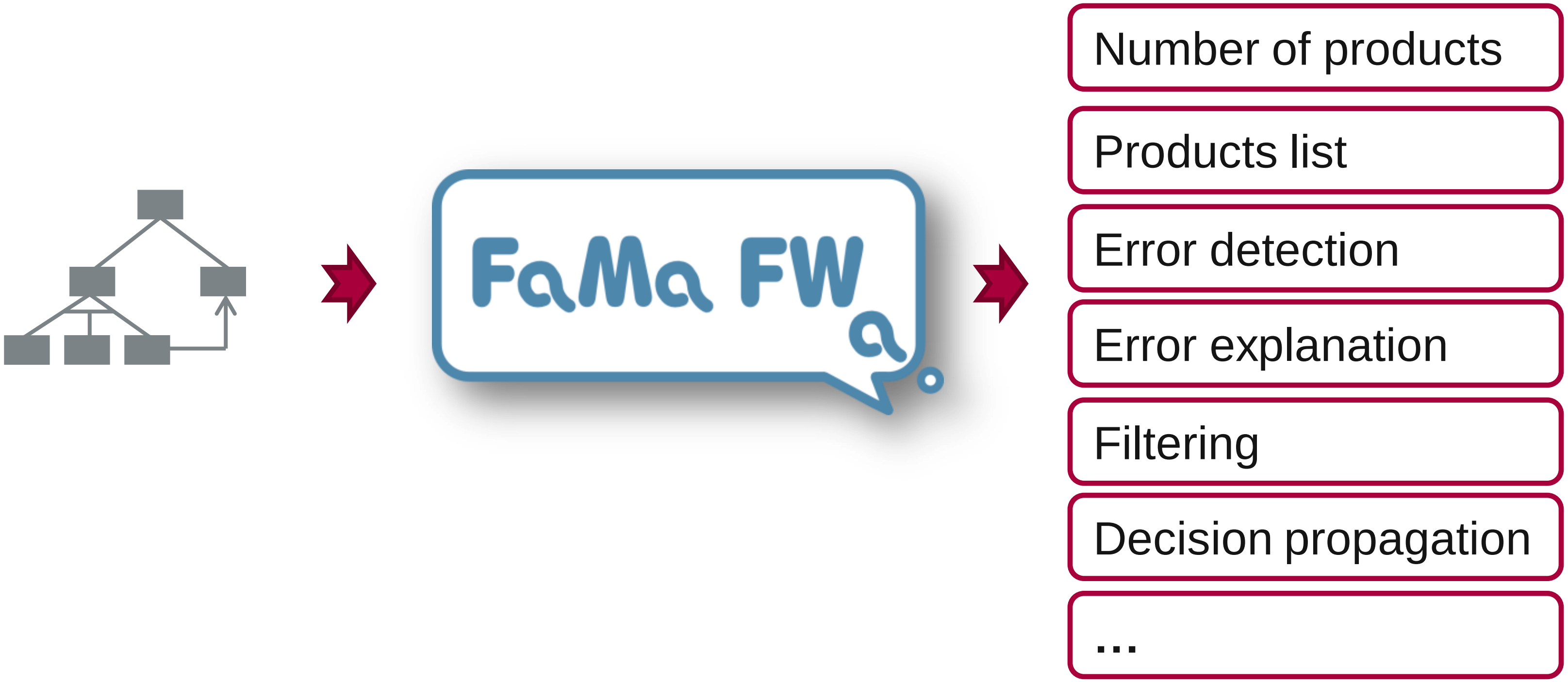
Problem

Contribution

Conclusions

Motivation

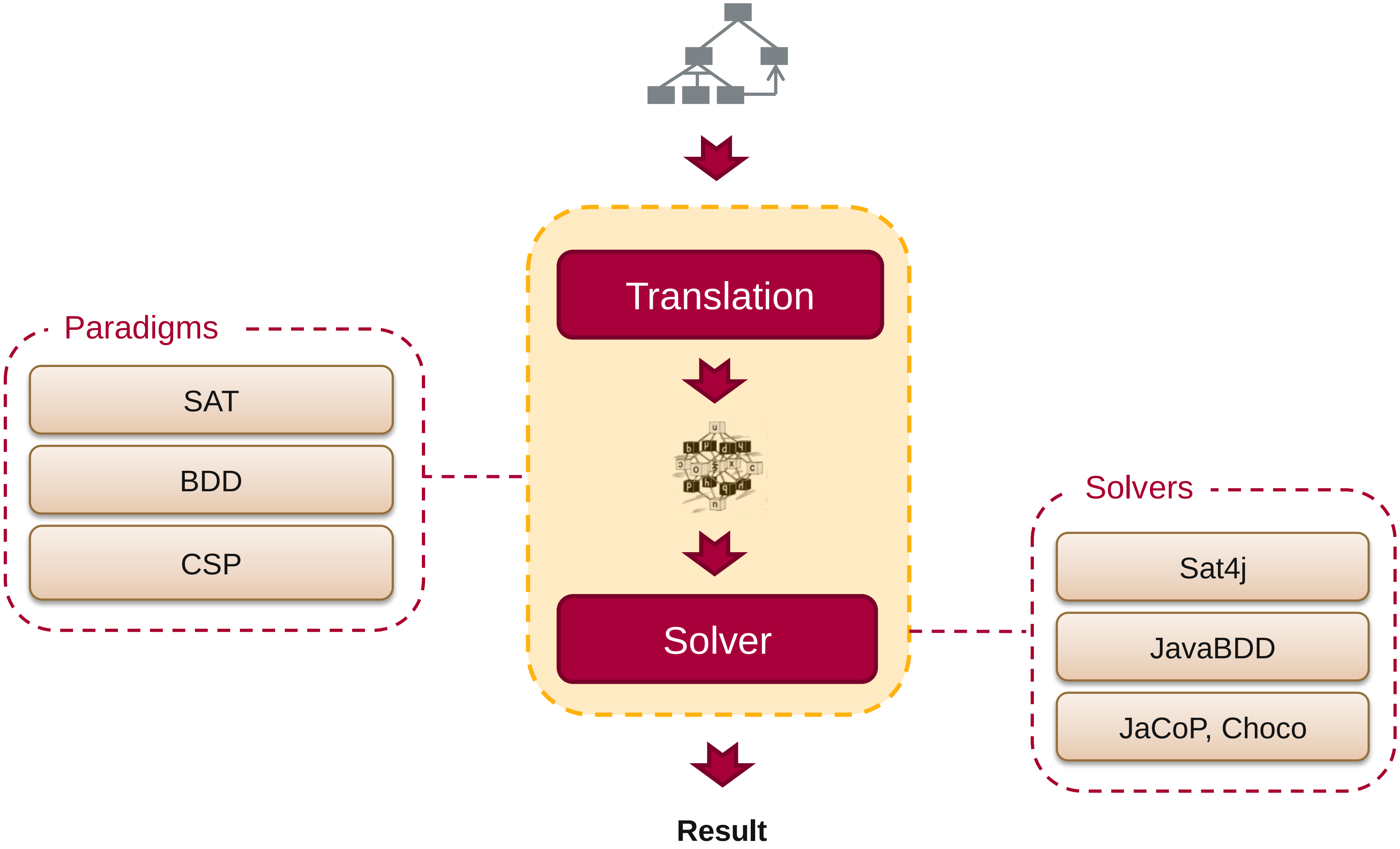
Automated analysis of feature models:
Computer-aided extraction of information from feature models



- Number of products
- Products list
- Error detection
- Error explanation
- Filtering
- Decision propagation
- …

Motivation

FAMA Framework



Translation

Solver

Paradigms

- SAT
- BDD
- CSP

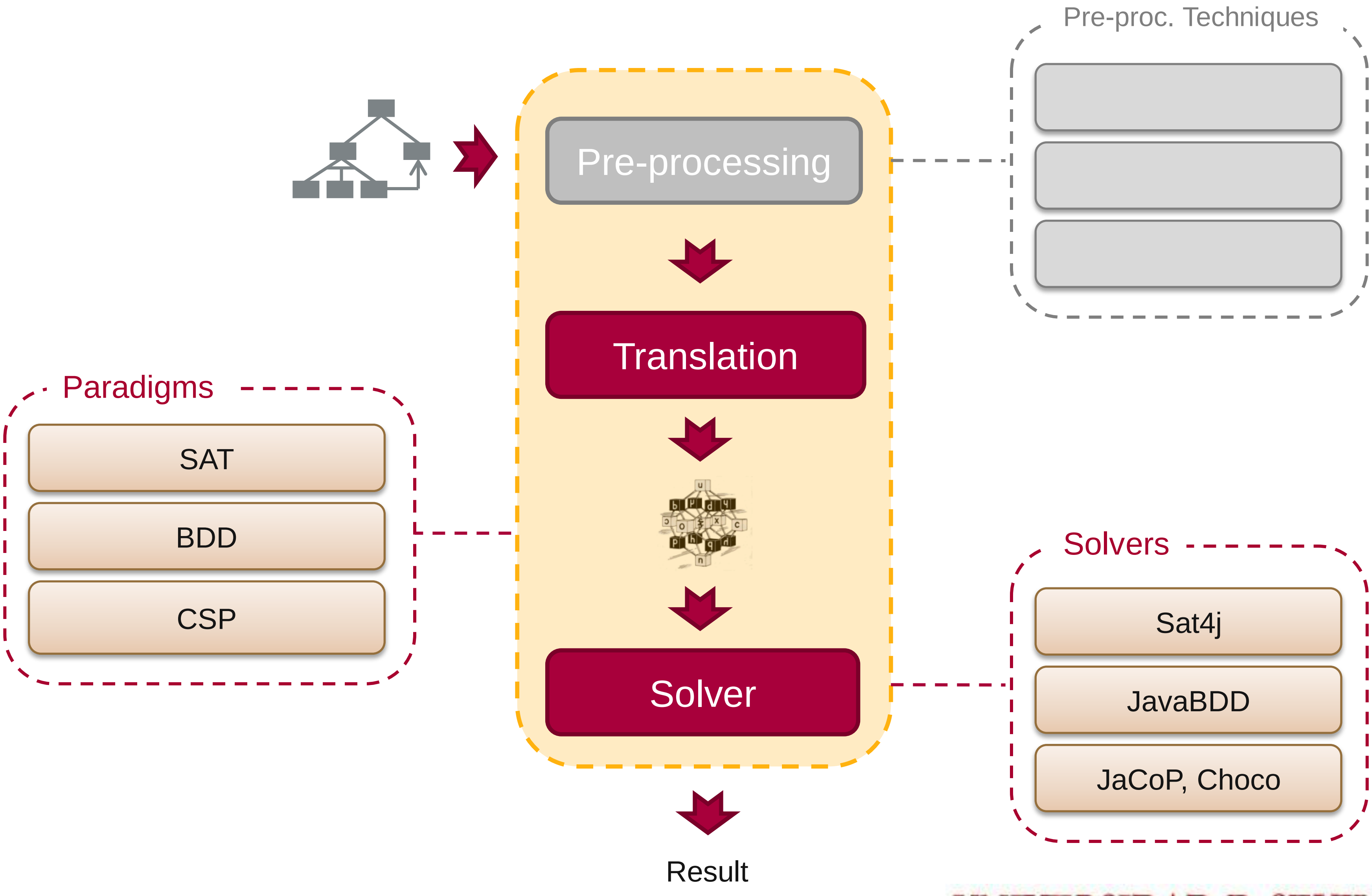
Solvers

- Sat4j
- JavaBDD
- JaCoP, Choco

Result

Motivation

FAMA Framework



Pre-proc. Techniques

Pre-processing

Translation

Solver

Result

Paradigms

- SAT
- BDD
- CSP

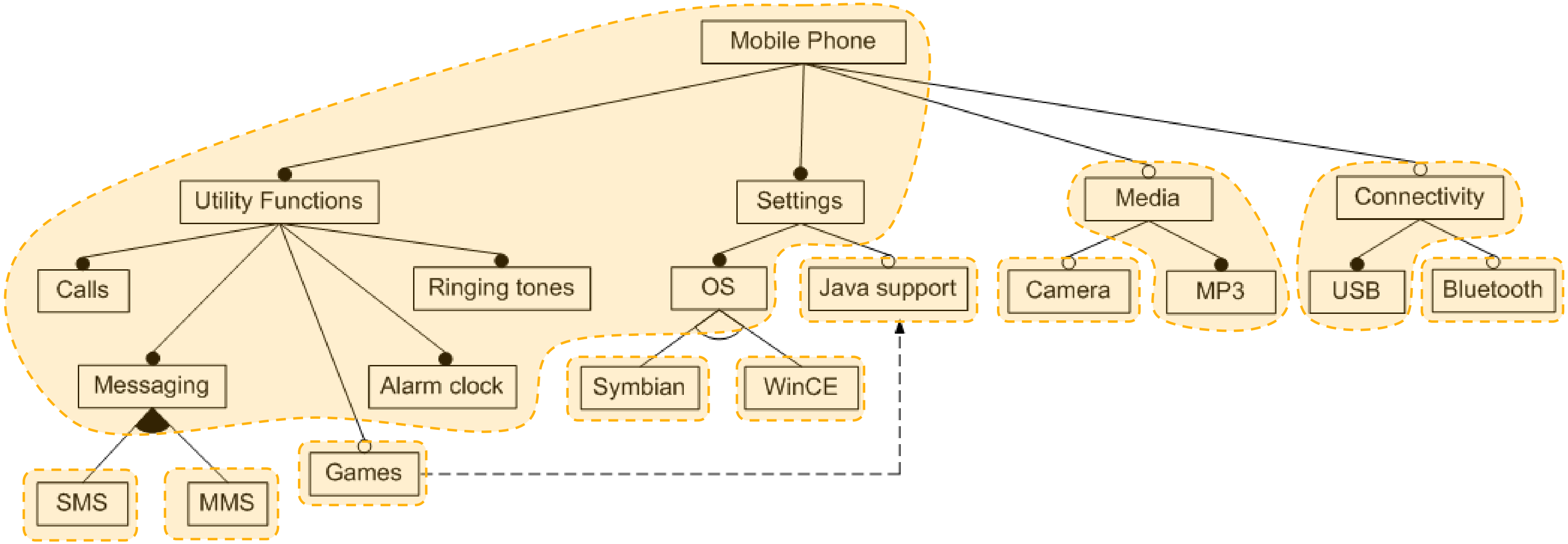
Solvers

- Sat4j
- JavaBDD
- JaCoP, Choco

Motivation

Atomic Sets

W. Zhang et al. (2004)



20 Features

20 Variables

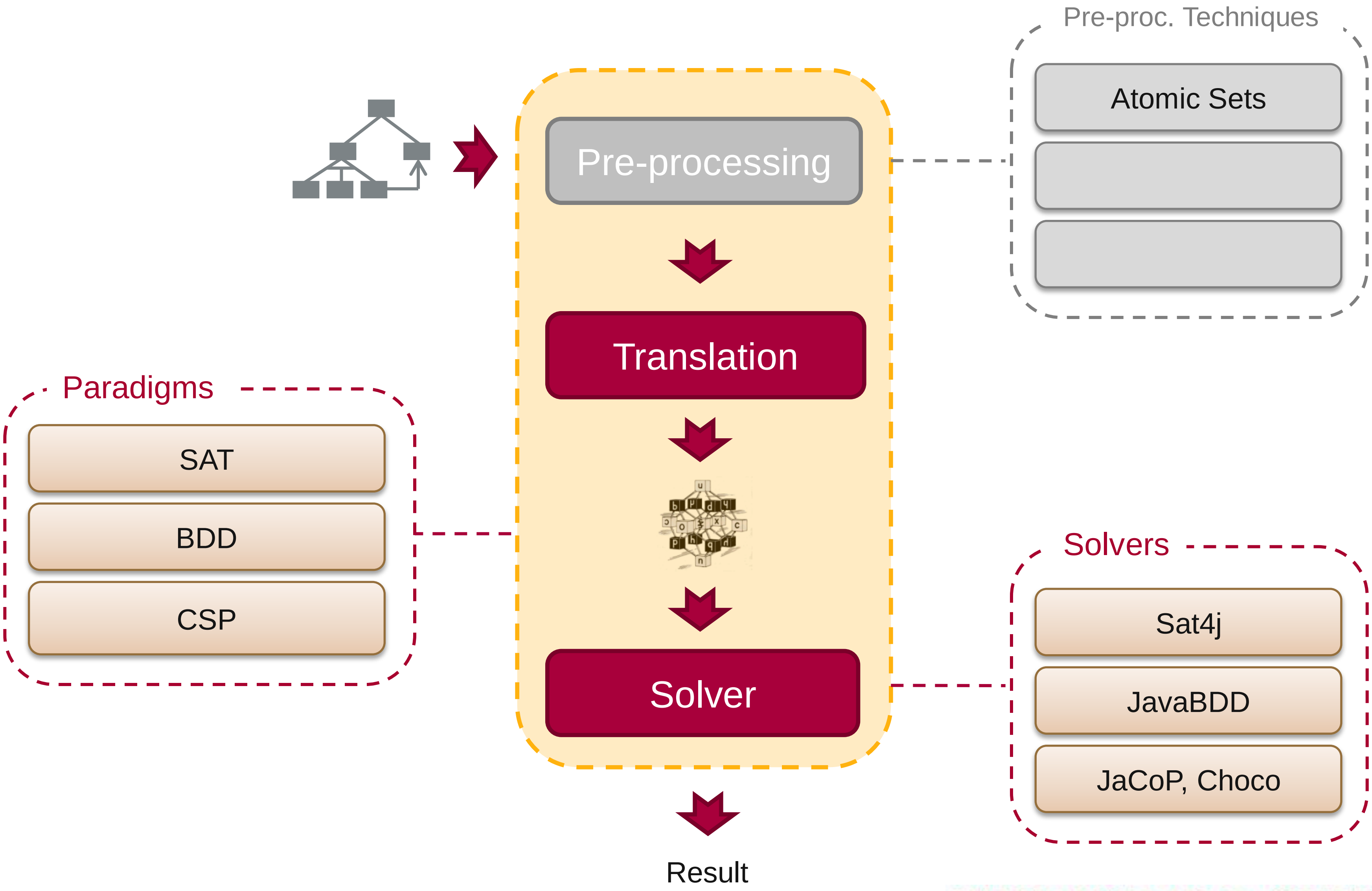
11 Atomic sets

11 Variables

UNIVERSIDAD Ð SEVILLA

Motivation

FAMA Framework



Pre-proc. Techniques

Atomic Sets

Pre-processing

Translation

Paradigms

SAT

BDD

CSP

Solver

Solvers

Sat4j

JavaBDD

JaCoP, Choco

Result

Motivation

Problem

Contribution

Conclusions

Problem

How can atomic sets be constructed?

What is the performance improvement when using ASs?

Motivation

Problem

Contribution

Conclusions

Contribution

How can atomic sets be constructed?

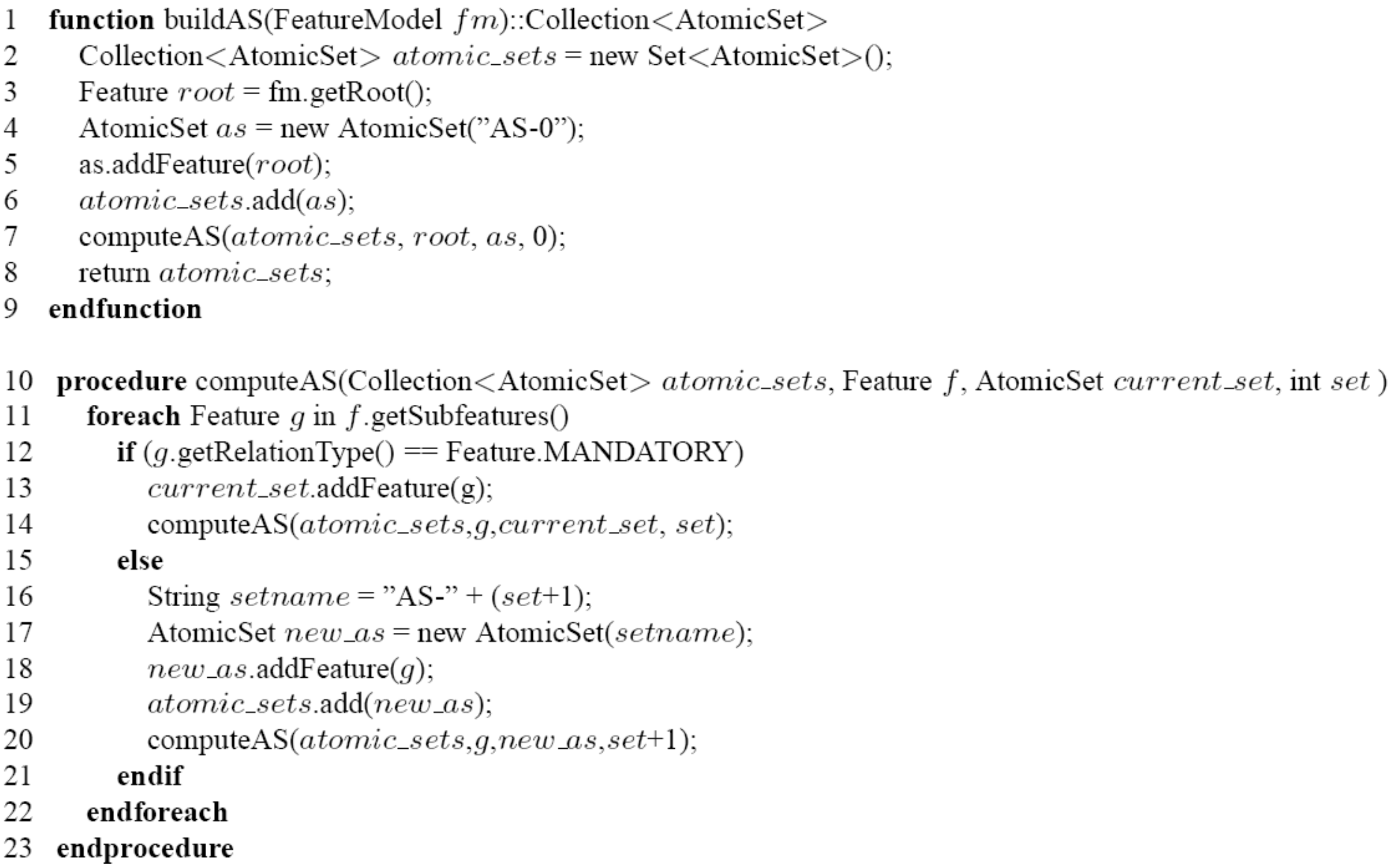
We propose an algorithm

What is the performance improvement when using ASs?

We provide an empirical performance comparison

Contribution

Algorithm for AS computation



```
function buildAS(FeatureModel fm)::Collection<AtomicSet>
   Collection<AtomicSet> atomic_sets = new Set<AtomicSet>();
   Feature root = fm.getRoot();
   AtomicSet as = new AtomicSet(”AS-0”);
   as.addFeature(root);
   atomic_sets.add(as);
   computeAS(atomic_sets, root, as, 0);
   return atomic_sets;
endfunction

procedure computeAS(Collection<AtomicSet> atomic_sets, Feature f, AtomicSet current_set, int set )
   foreach Feature g in f.getSubfeatures()
      if (g.getRelationType() == Feature.MANDATORY)
         current_set.addFeature(g);
         computeAS(atomic_sets,g,current_set, set);
      else
         String setname = ”AS-” + (set+1);
         AtomicSet new_as = new AtomicSet(setname);
         new_as.addFeature(g);
         atomic_sets.add(new_as);
         computeAS(atomic_sets,g,new_as,set+1);
      endif
   endforeach
endprocedure
```

Contribution

Experimental results

N. of features	N. of instances	CTC
[50 – 100)	50	[0 – 25%]
[100 – 150)	50	[0 – 25%]
[150 – 200)	50	[0 – 25%]
[200 – 300]	50	[0 – 25%]

Operations

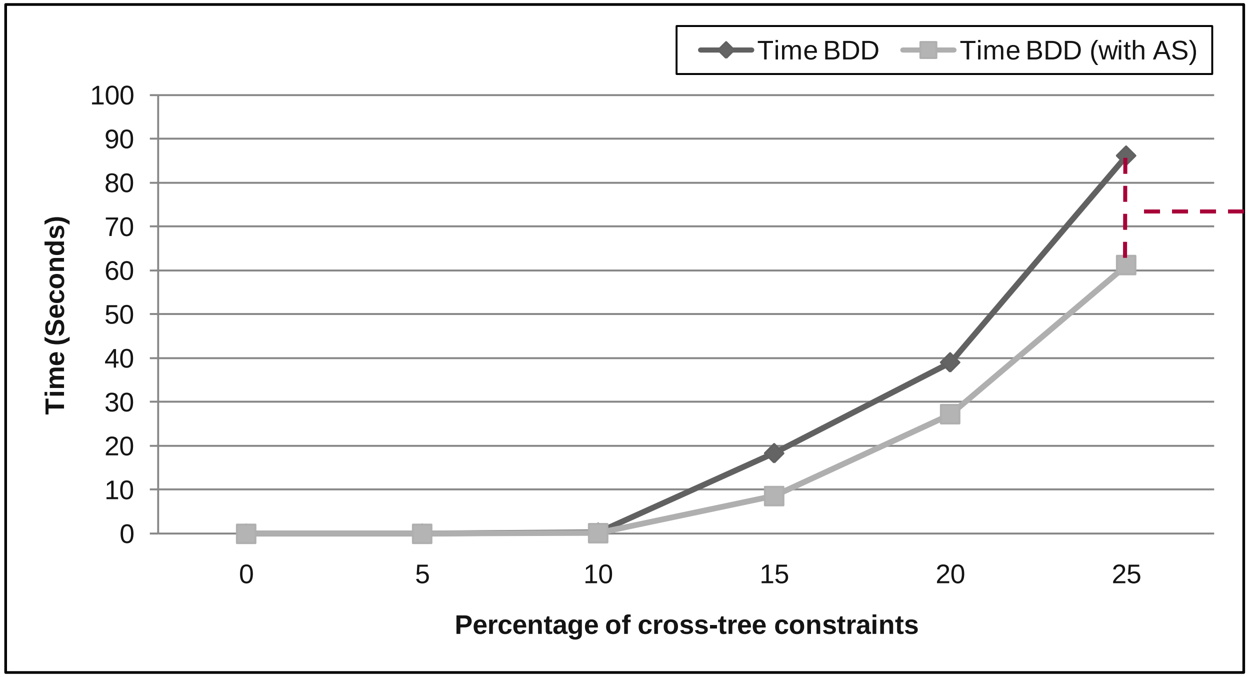
- Void FM
- Number of products

Solvers

- JaCoP (CSP)
- SAT4j (SAT)
- JavaBDD (BDD)

Contribution

Experimental results

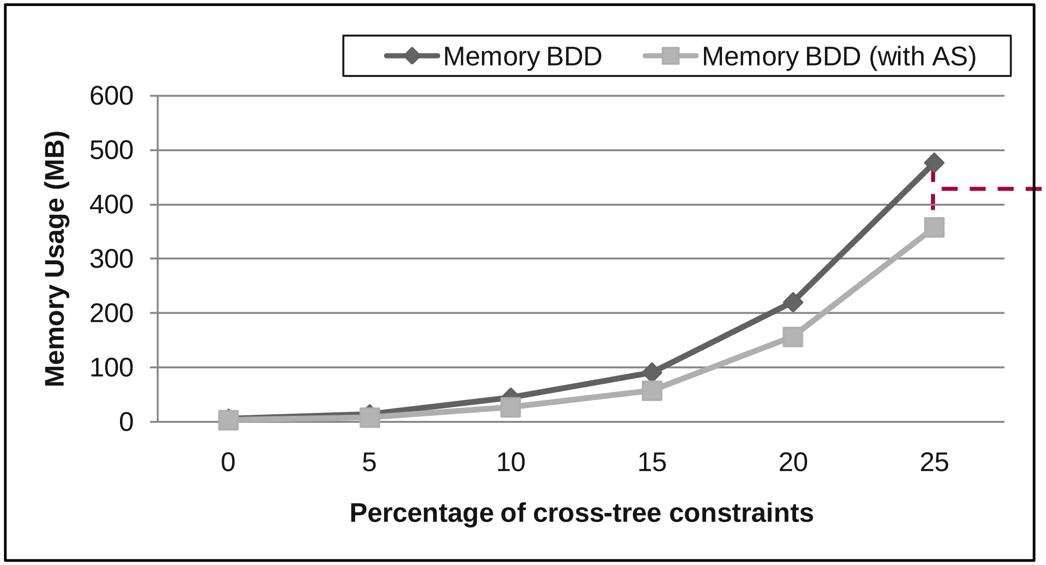


25 seconds
(28% improvement)

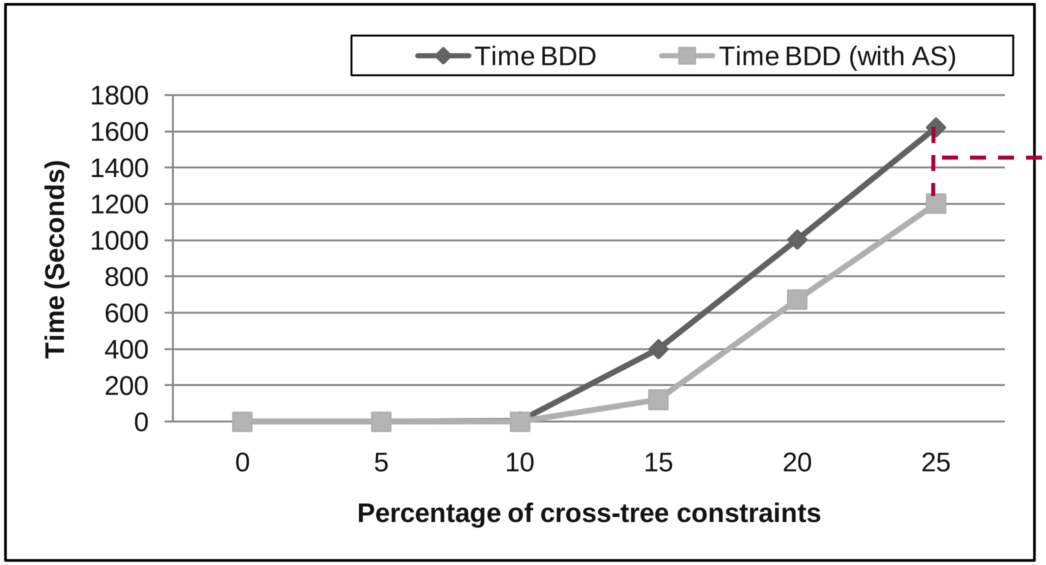
Average time to find the number of products (Range 200-300 features)

Contribution

Experimental results



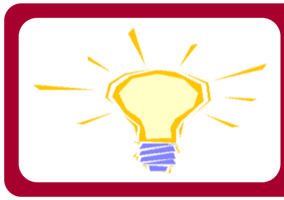
119 MB
(25% improvement)



7 minutes
(26% improvement)

Motivation

Problem



Contribution

Conclusions

Conclusions

- We set the basis for the usage of AS as a generic technique for the automated analysis of feature models.

- We propose an algorithm for the construction of AS + Empirical measurement of performance improvement.

- Improvement when using atomic sets can be notable in both time and memory.

- The cost of implementing this technique is minimum.

- These kind of techniques may be applicable to the analysis of other kind of variability models.

Thanks!

sergiosegura@us.es

isa

www.isa.us.es

FaMa Tool Suite

www.isa.us.es/fama