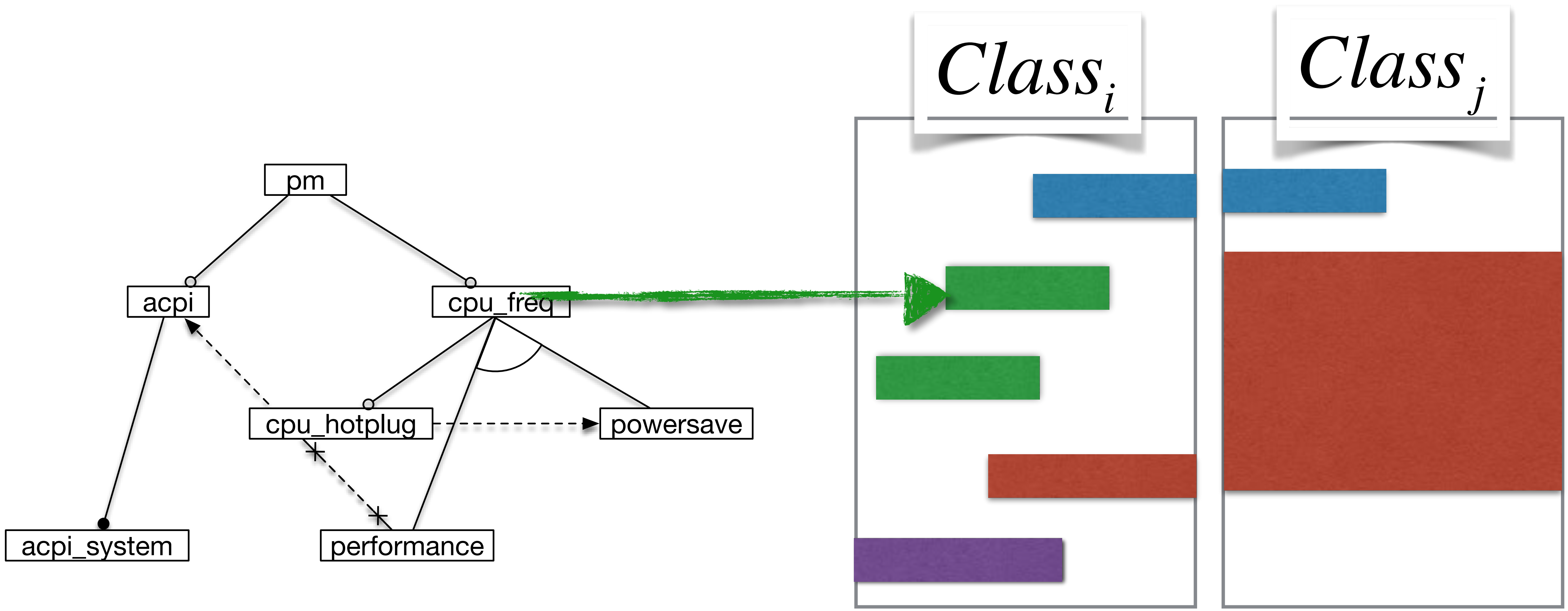
StiCProb: A Novel Feature Mining Approach using Conditional Probability

Yutian Tang, and Dr. Hareton Leung
The Hong Kong Polytechnic University, Hong Kong
@SANER 2017
more: www.chrisyttang.org/loong

Motivation

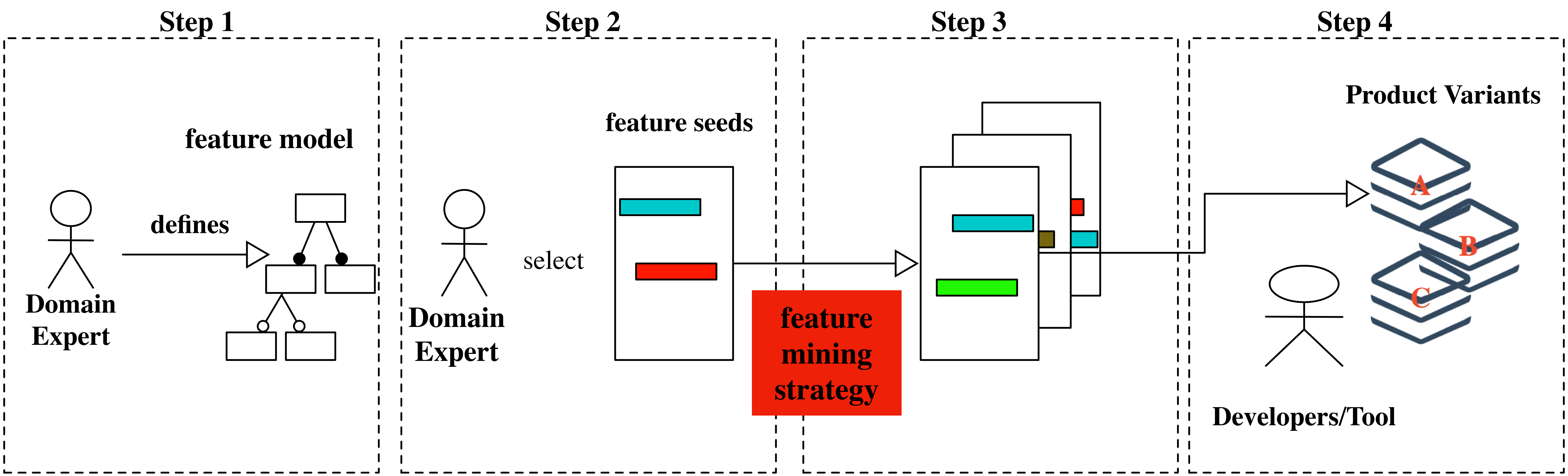
- legacy —-> product line



1. How to locate the feature ?
2. How to measure ?

feature $\longleftrightarrow$ *element*

How to locate?



1. Select seeds
2. Annotate features

Basic

Programming elements E

Relationship $R \subseteq E \times E$

Feature F

Annotation $A \subseteq E \times F$

Basic (cond')

mutual exclusion $M \subseteq F \times F$

implications $\Rightarrow \subseteq F \times F$

full annotation $A \subseteq E \times F$

$$A^* = \left\{ (e,f) \middle| (e,f) \in A, g \Rightarrow^* f \right\}$$

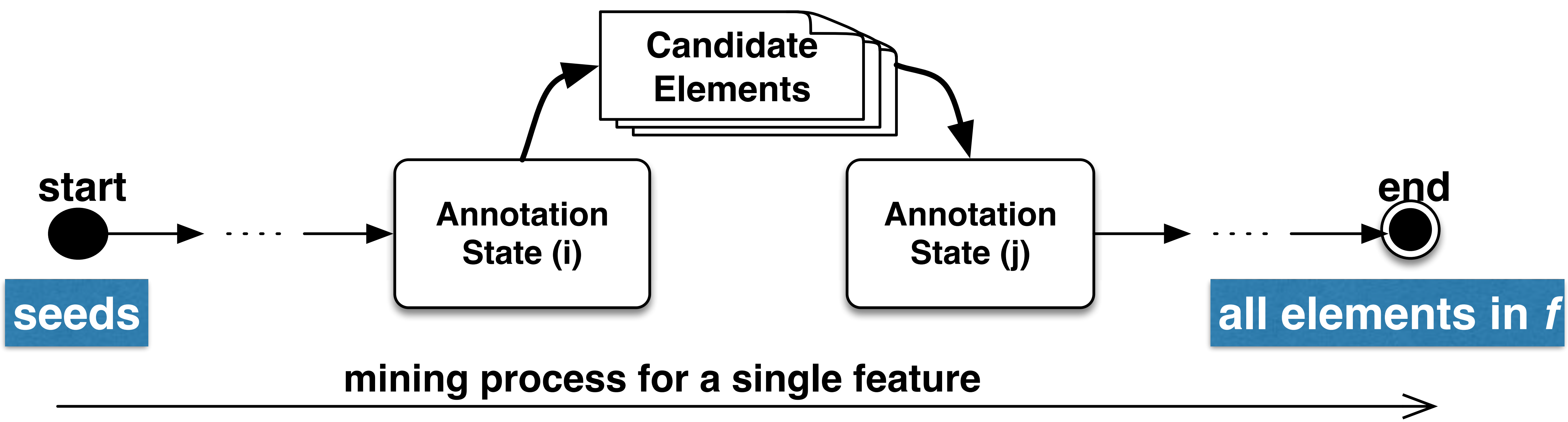
(e,f)

$(e,f) | (e,f) \in A, g \Rightarrow f$

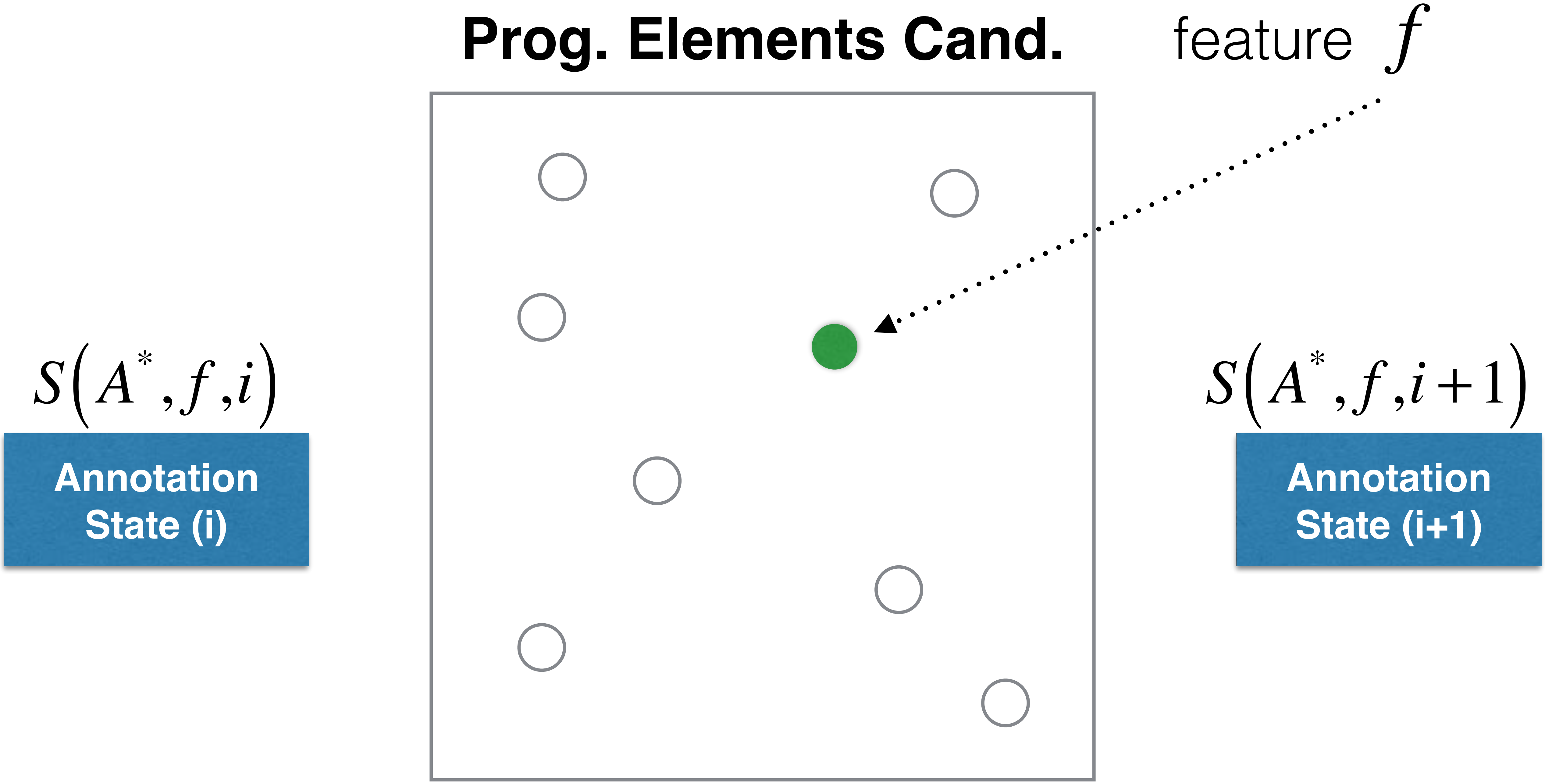
Annotate Features

Annotation State

$$S(A^*, f, i) = \left\{ e \middle| (e, f) \in A^* \right\}$$



Feature-Element Correlation Coefficient



Feature-Element Correlation Coefficient(cond')

Prog. Elements Cand.

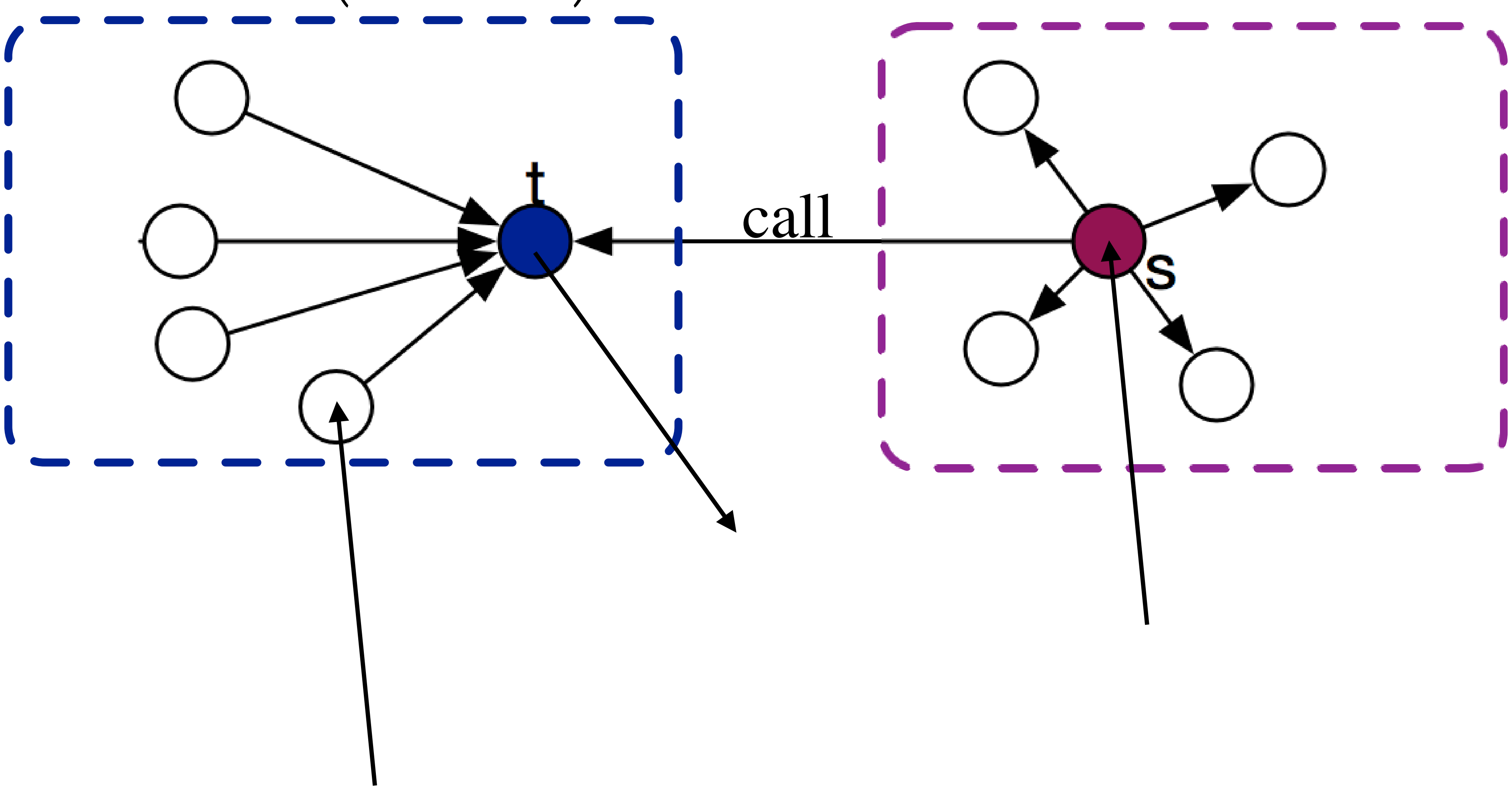
feature f

$$S\left(A^{*},f,i\right) \dashrightarrow \bullet \dashrightarrow S\left(A^{*},f,i+1\right)$$

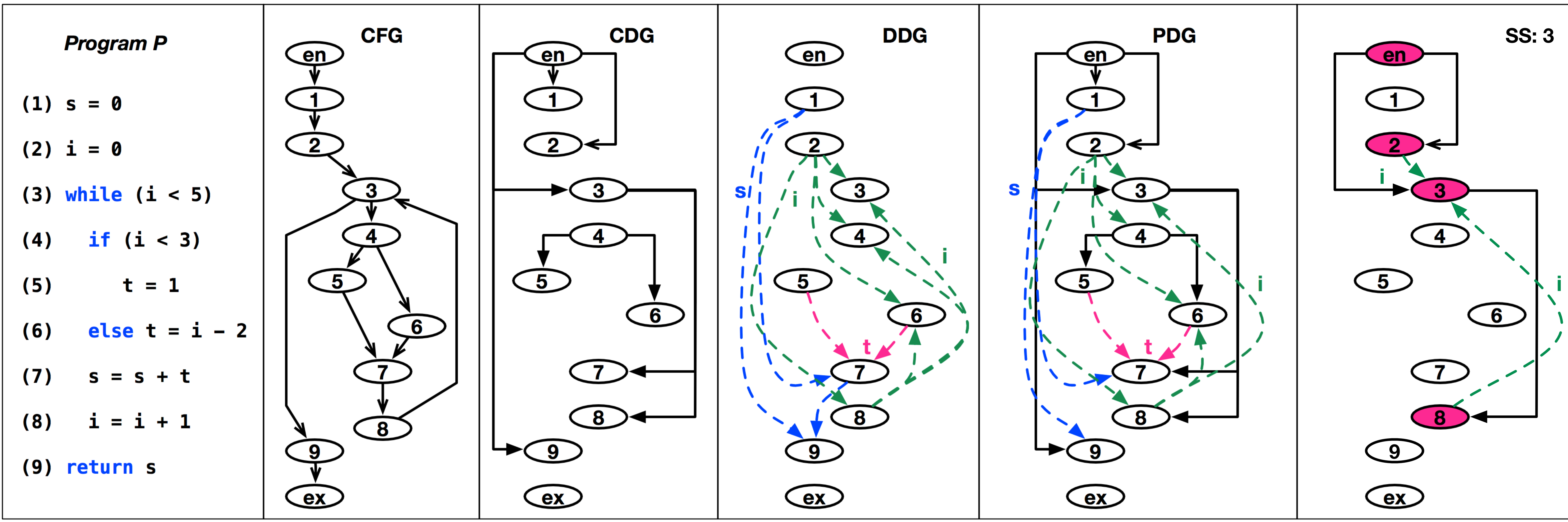
$$p\left(e \middle| S\left(A^{*},f,i\right)\right)$$

Feature-Element Correlation Coefficient(cond')

$t \in S\left(A^*, f, i\right)$



Feature-Element Correlation Coefficient(cond')



Slicing Scope

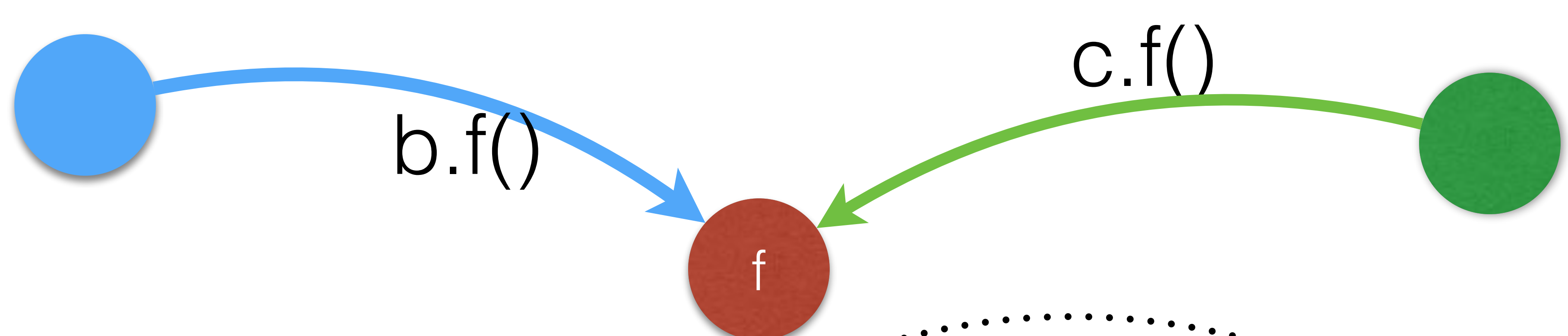
$$sscope(e) = e \cup \left\{ s \,\middle|\, s \xrightarrow[cf]{df} e, s \in E \right\}$$

Feature-Element Correlation Coefficient(cond’)

Binding

$$bind(e) = def(e) \cup use(e)^{*} \quad \text{in} \quad sscope(e)$$

Context Binding



$$contbind(e,[c])$$

input context

* All def and use within e

Context Binding

Context Binding @ Method Invocation

callsite $l = r_0.m(r_1, r_2, \ldots, r_n)$

$$contbind(m, [r_1, \ldots, r_n]) = dispatch(p_i = r_i) \rightarrow bind(m)$$

[13] L. O. Anderson, “Program analysis and specification for c programming language”

Context Binding

Context Binding @ Method Invocation


```
main(){
    A a = new A();
csite 1: z = wrapper(a);
}
```

callsite 1

```
wrapper (A b){
    y = bar(b); return y;
}
```

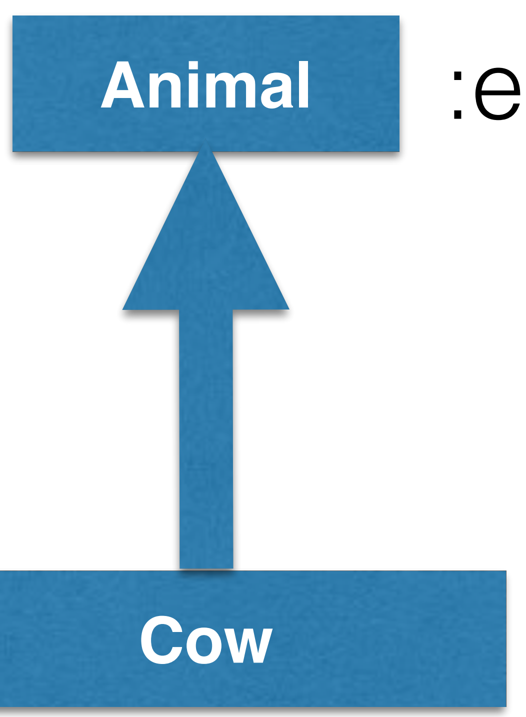
$$contbind\left(wrapper(..),[a]\right)$$

```
bar (A c){
    x = c.f; return x;
}
```

[13] L. O. Anderson, “Program analysis and specification for c programming language”

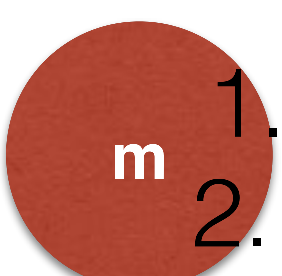
Context Binding

Context Binding @ Overriding



:eat(food: Food)

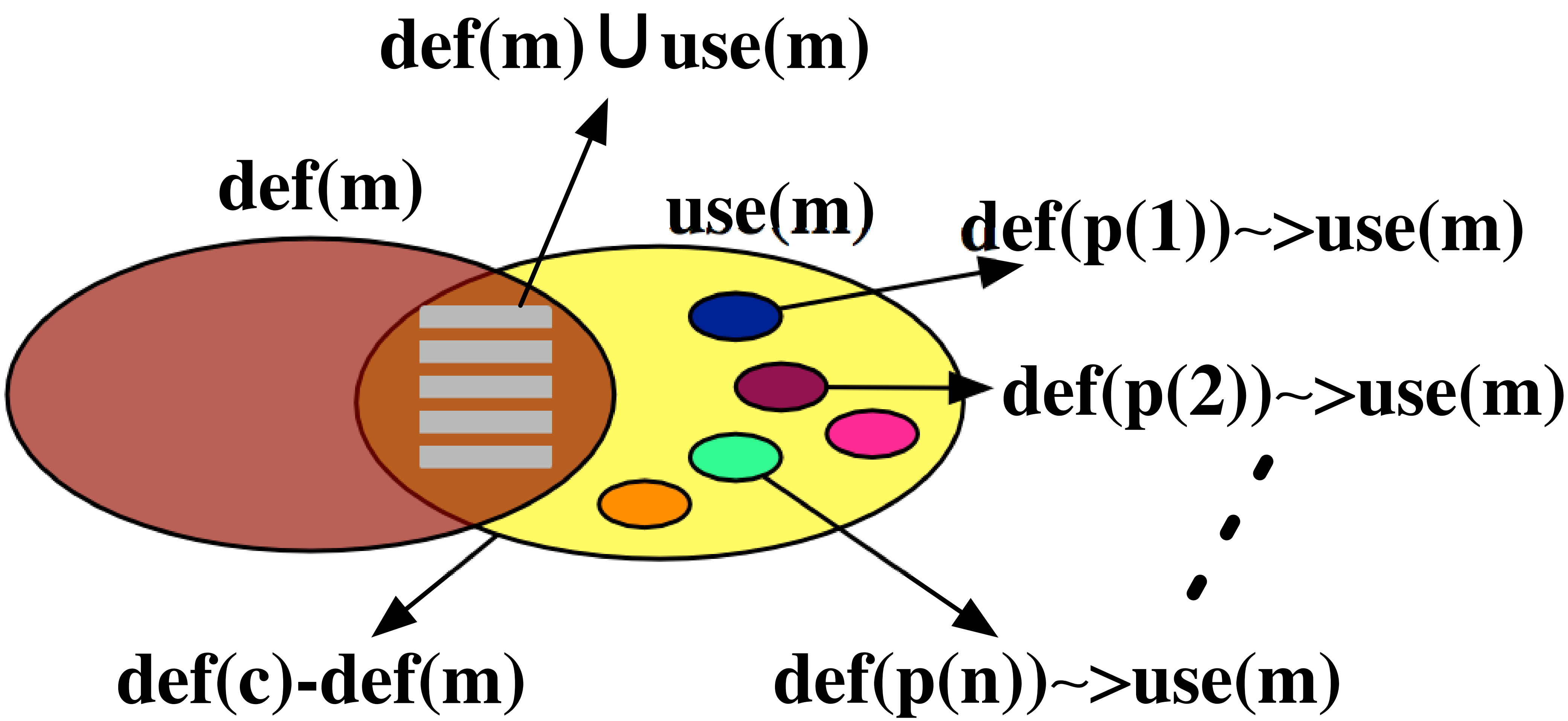
:eat(food: Grass)



1. def and use in m
2. def in parent class/interface, used in m

Context Binding

Context Binding @ Overriding



Context Binding

Context Binding @ Overriding

defined in m: $def(m)$

used in m: $use(m)$

1. used in m and defined in m
2. used in m and not defined in m

$$contbind\left(m,\left[p_1,\ldots,p_n\right]\right) = def(m) \cup \bigcup_{i=1}^{n}\left(use(m) \sim> def(p_i)\right)$$

~> used to specify the source of the context

Context Binding

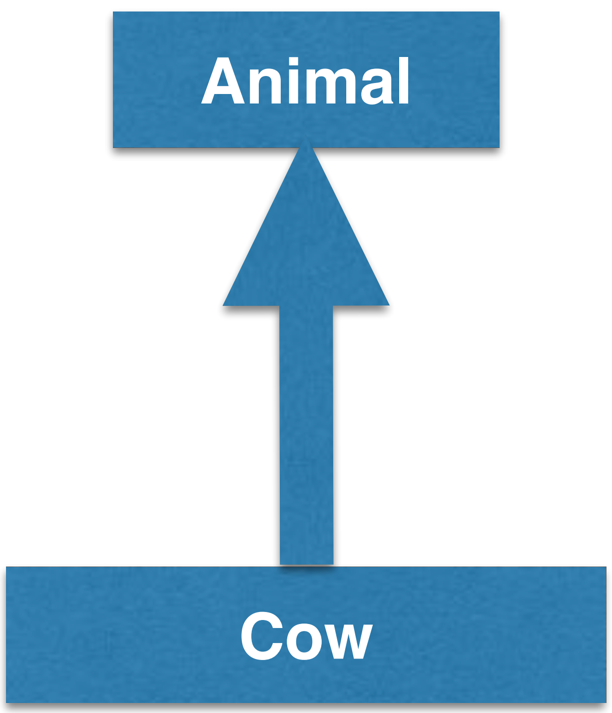
Context Binding @ Overriding : **Example**

```
public class FlyingCar implements OperateCar {
   public int startEngine(int encryptedValue) {
       OperateCar.super.startEngine(OperateCar.encryptedValue);
   }
}
---------------------------------------------------------
public interface OperateCar {
   int encryptedValue = 1;
   default public int startEngine(int value) {...}
}
```

$$contextbind(startEngine) = \{encryptedValue, OperateCar.encryptedValue\}$$

Context Binding

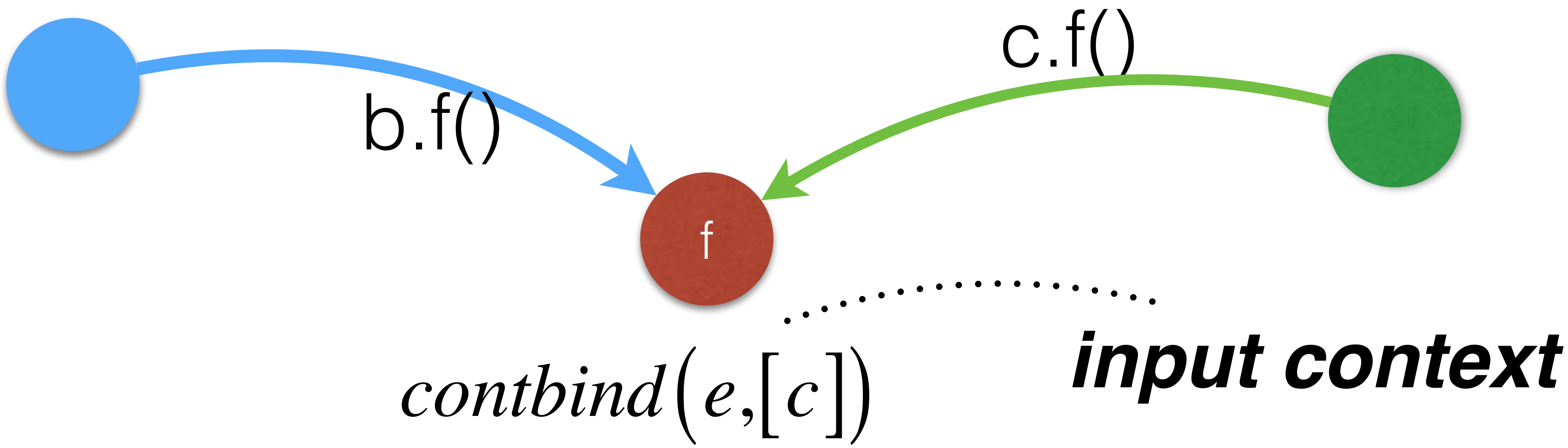
Context Binding @ Inheritance



$$contbind\left(c,\left[p_1,\ldots,p_n\right]\right)=bind(c)\cup\bigcup_{i=1}^{n}def(p_i)$$

Feature-Element Correlation Coefficient(cond')

Context Binding



by default: ***context-aware points-to analysis***

Feature-Element Correlation Coefficient(cond')

$$S(A^{*},f,i)=\left\{e\,\middle|\,\left(e,f\right)\in A^{*}\right\}$$

$$S(A^{*},f,i)=\bigcup_{a}^{a\in S\left(A^{*},f,i\right)} contextbind\left(a\right)$$

StiCProb

1. Build Program DB

2. Build uniqueness table

3. Annotate features

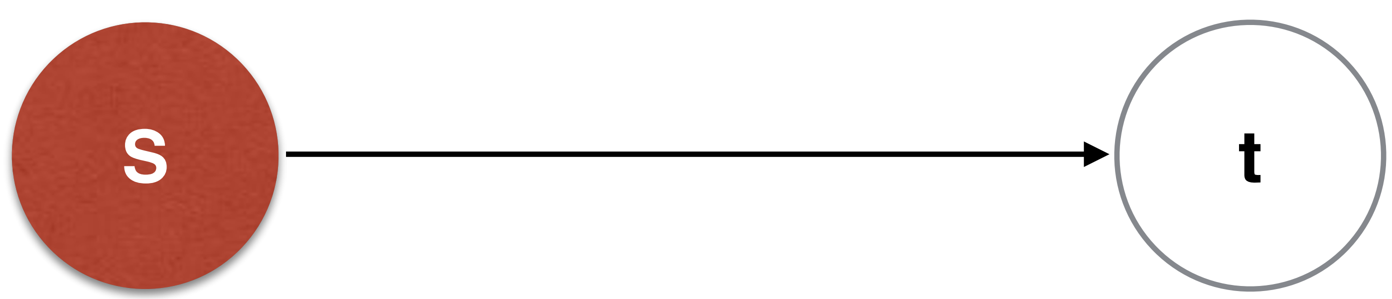
StiCProb: Uniquess Table

element *s* and *t* with a relation r $\quad s \xrightarrow{r} t$

$$U\left(E,T,R,P_{forward},P_{backward}\right)$$

$$P_{forward}$$

the uniqueness of *t to s* if *s* has been annotated to a feature ***f***

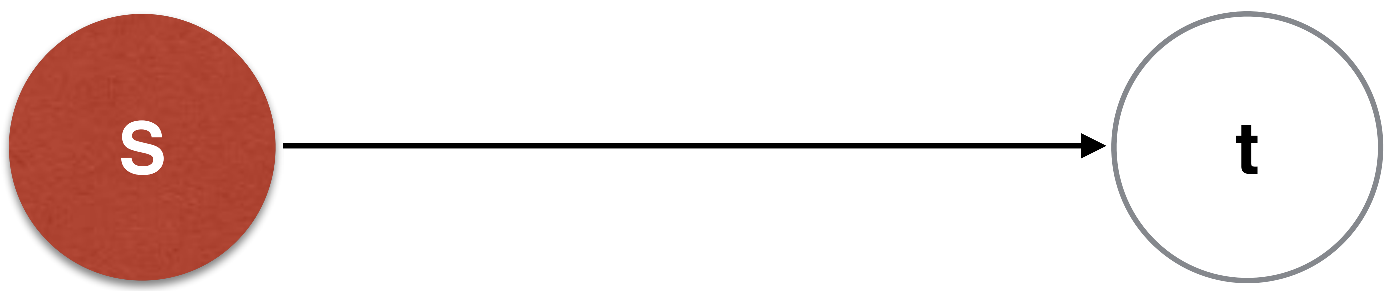


$s \in S\left(A^*,f,i\right) \qquad S(A^*,f,i)=\left\{e \middle| (e,f) \in A^*\right\}$

StiCProb: Uniquess Table(cond')

$$P_{forward}$$

the uniqueness of *t to s* if *s* has been annotated to a feature ***f***



$s \in S\left(A^*, f, i\right)$ $\qquad$ $S(A^*, f, i) = \left\{ e \middle| \left(e, f\right) \in A^* \right\}$

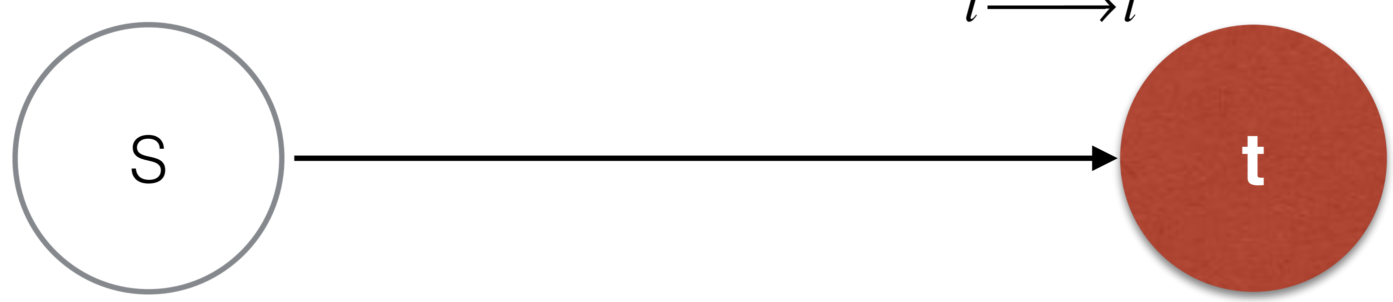
$$p_{forward}\left(s \xrightarrow{r} t \middle| \left(s, f\right) \in A^*\right) = \frac{contbind\left(t, [s]\right)}{contbind\left(s\right)}$$

StiCProb: Uniquess Table(cond')

$P_{backward}$

the uniqueness of <u>*s* to *t*</u> if *t* has been annotated to a feature ***f***

$$p_{backward}\left(s \xrightarrow{r} t \,\middle|\, (t, f) \in A^{*}\right) = \frac{contbind\left(t, [s]\right)}{\bigcup_{i \xrightarrow{r} t} contbind\left(t, [i]\right)}$$

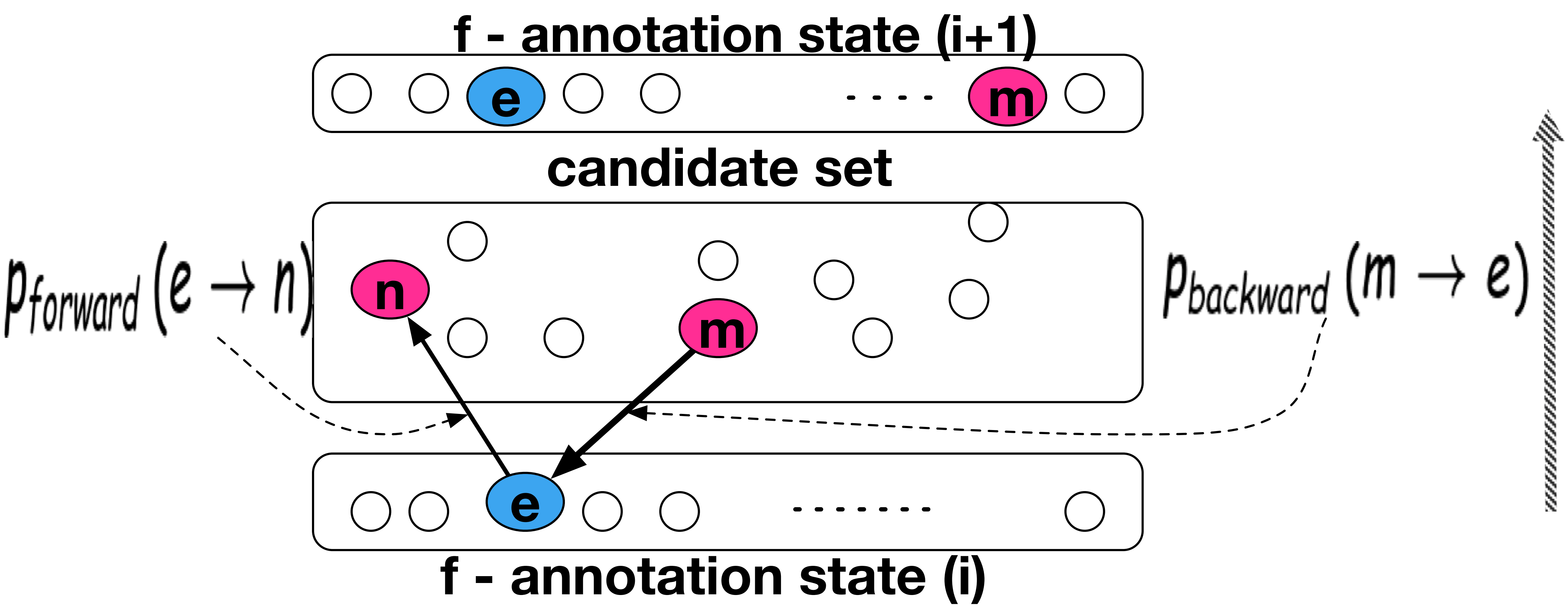


$s \in S\left(A^{*}, f, i\right)$

$S(A^{*}, f, i) = \left\{ e \,\middle|\, (e, f) \in A^{*} \right\}$

$\bigcup_{i \xrightarrow{r} t} contbind\left(t, [i]\right)$ a collection of context binding from all prog. elements, which have relation *r* with *t*.

StiCProb



f - annotation state (i+1)
e
. . . .
m
candidate set
$p_{forward}(e \to n)$
n
m
$p_{backward}(m \to e)$
e
. . . .
f - annotation state (i)

StiCProb

Algorithm 1: StiCProb feature mining approach

Input: $seeds$, fm, $threshold$, U

Output: all annotation states for features $Sset$ in fm

Create a set of annotation states as $Sset = \bigcup_{f}^{f \in features} S(A^*, f)$;

Assign seeds to each feature as $S(A^*, f) = seeds(f)$;

Create feature set $features$ with all features in fm;

while *$features$ not NULL* **do**

  for *feature f in $features$* **do**

    Create set $waitList = \varnothing$;

    Create candidate set $C(S, f) = \varnothing$ for f;

    Add all elements have relations with elements in $S(A^*, f)$ to $C(S, f)$; // initialize $C(S, f)$

    for *element m in $C(S, f)$* **do**

      if *there is a relation r from m to the element e in $S(A^*, f)$* **then**

        Let $value = p_{backward}\left(m \xrightarrow{r} e | (e, f) \in A^*\right)$;

      else

        Let $value = p_{forward}\left(e \xrightarrow{r} m | (e, f) \in A^*\right)$;

      if *$value > threshold$* **then**

        Add m to $waitList$;

    Update $S(A^*, f) \leftarrow S(A^*, f) \cup waitList$;

    if *StopCheck(f) is TRUE* **then**

      Remove f from $features$;

return $Sset$;

Feature model(fm)

Seeds(seeds)

threshold

Uniqueness Table(U)

Case Study

Projects	LOC	#features	domain
Prevalyer	8,009	5	object persistence library
MobileMedia	4,653	6	mobile
Lampiro	44,584	*2	message client
ArgoUML	~120K	7	modeling tool

Case Study

Experimental Setting:

seeds: FLAT3 tool

tool: Loong Eclipse plugin

feature model: benchmark

benchmark

Related Approaches:

Type system, Topology analysis, Text comparison

Measurement:

precision recall f-score

Case Study

Other Settings:

seeds: 3

threshold: 0.6

Experimental Result

StiCProb with threshold *t = 0.6*

		Feature Size			Mining Results		
Project	Feature	LOC	FR	FI	IT	Recall	Prec.
Prevayler	Censor	105	10	5	3	17%	60%
	Gzip	165	4	4	3	16%	100%
	Monitor	240	19	8	2	17%	82%
	Replication	1487	37	28	26	79%	98%
	Snapshot	263	29	5	9	42%	99%
MobileM.	CopyMedia	79	18	6	4	43%	95%
	Sorting	85	20	6	4	32%	100%
	Favorites	63	18	6	12	20%	100%
	SMS Trans.	714	26	14	23	91%	49%
	Music	709	38	16	4	39%	90%
	Photo	493	35	13	5	63%	61%

LOC: line of code, **FR**: count of distinct code fragments, **IT**: number of iteration, **Prec.**: precision

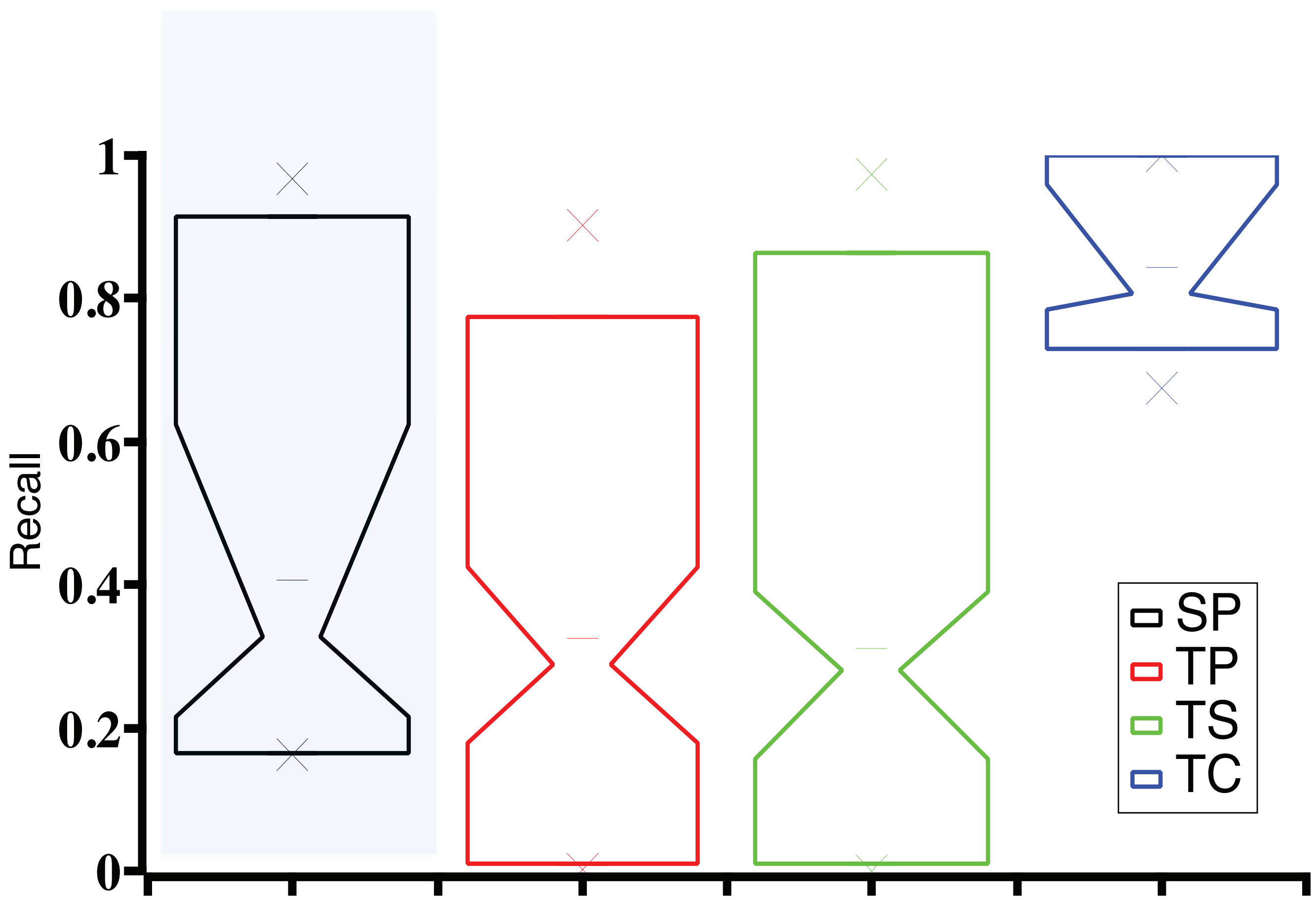
Experimental Result (cond')

StiCProb with threshold $t = 0.6$

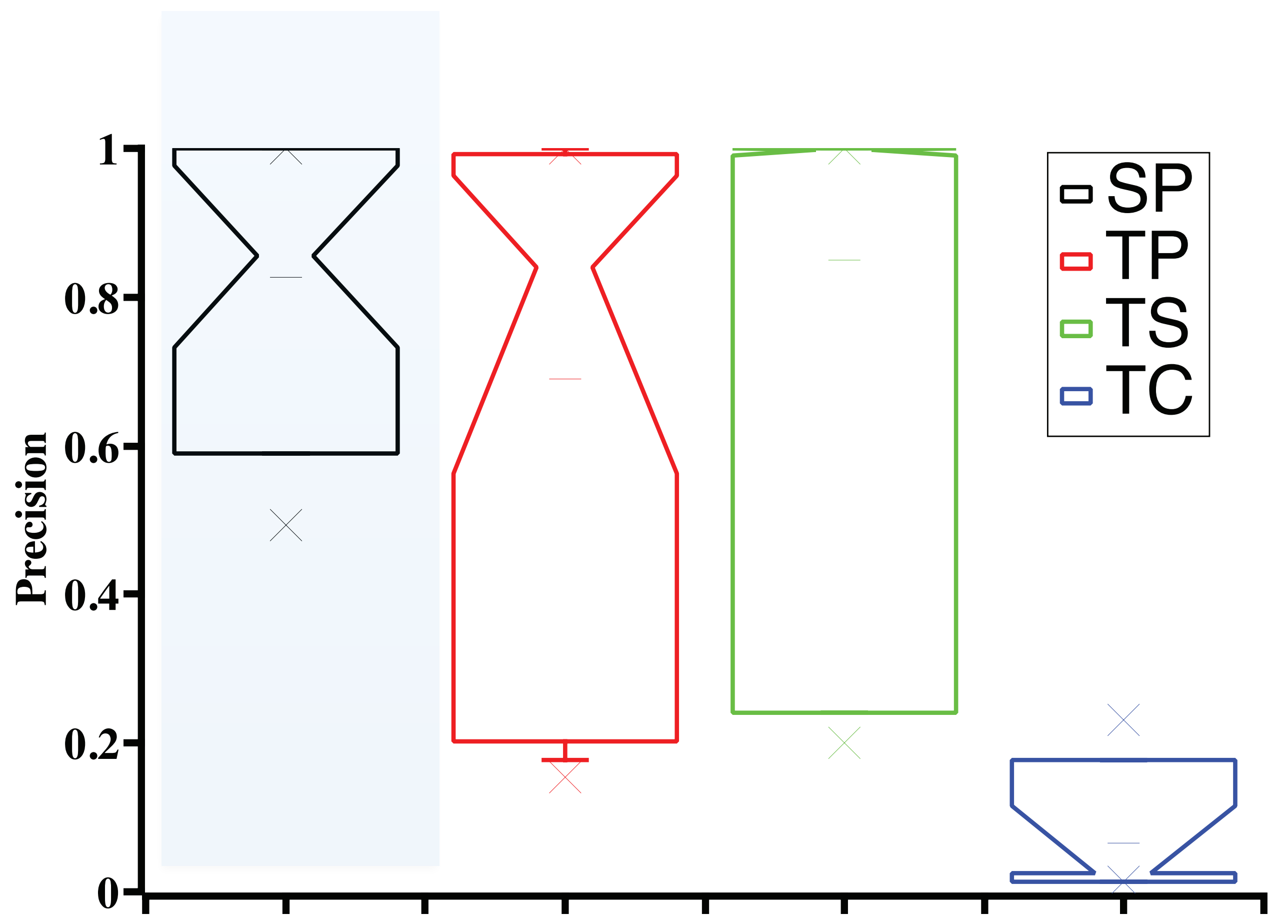
		Feature Size				Mining Results	
Project	Feature	LOC	FR	FI	IT	Recall	Prec.
MobileM.	M.Transfer	153	4	3	14	97%	94%
Lampiro	Compre.	5155	33	20	34	40%	82%
ArgoUML	Cognitive	16319	285	233	127	70%	92%
	Activity	2282	115	80	17	26%	74%
	State	3917	115	88	18	33%	82%
	Collab.	1579	53	40	40	17%	72%
	Sequence	5379	65	53	98	33%	89%
	Use-Case	2712	59	49	39	19%	70%
	Deployment	3147	57	47	36	22%	67%

LOC: line of code, **FR**: count of distinct code fragments, **IT**: number of iteration, **Prec.**: precision

Experimental Result (cond')



Recall Performance



Precision Performance

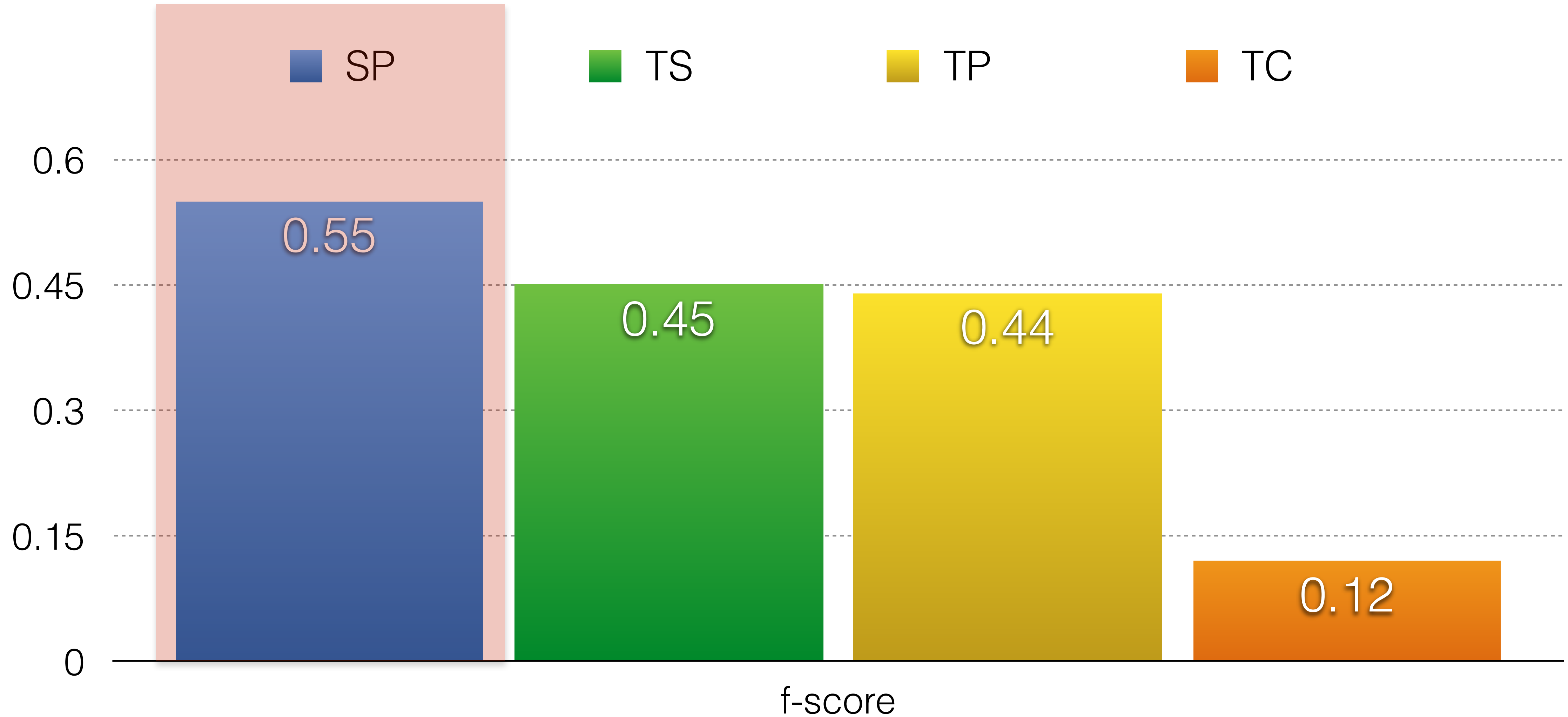
SP: StiCProb (t = 0.6)

TP: topology analysis

TS: type system

TC: text comparison

Experimental Result
f-score



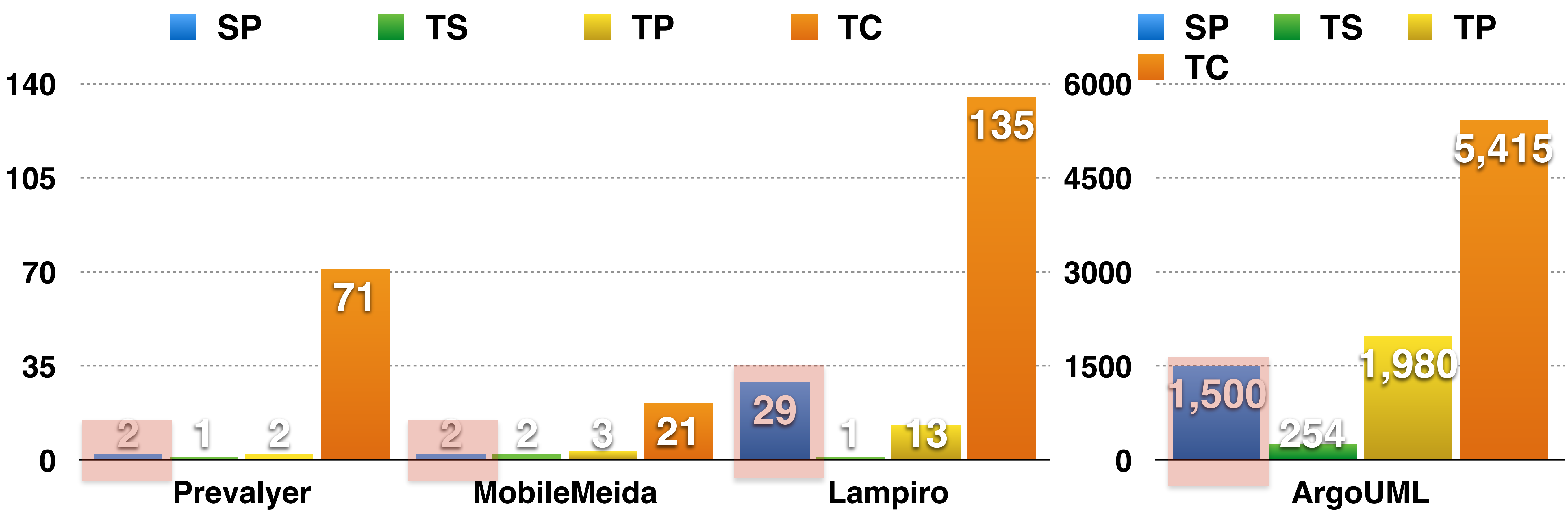
SP: StiCProb (t = 0.6)

TS: type system

TP: topology analysis

TC: text comparison

Experimental Result
Runtime



SP: StiCProb (t = 0.6)

TS: type system

TP: topology analysis

TC: text comparison

Discussion

Seeds:

1. seeds provided by FLAT3 might be not correct

2. number of seeds

3. granularity of seeds: coarse granularity could improve the recall performance, but sometimes at the cost of precision.

Discussion

Thresholds:

threshold: 0.6 —-> 0.8

precision: 83% —-> 85%

The *threshold* contributes less to the performance.

Structure of the program

Thanks

Loong Plugin

- Download: http://www.chrisyttang.org/loong/
- Source code: https://github.com/csytang/Loong
- Experimental results: https://drive.google.com/folderview?id=0B9l0qvk6pnW0ZDRYMmxIQVhRb0U&usp=sharing
- Online Tutorial: http://www.chrisyttang.org/loong/

Discussion

- Need of req. specification <—> seeds selection/ poor naming
- Variants of our approach? or better solutions ?
- - weighted graph —> graph clustering
- ?