Program Analysis with Dynamic Change of Precision

Dirk Beyer

Joint work with
Tom Henzinger and Grégory Théoduloz

SFU Simon Fraser University, BC, Canada

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE EPFL, Switzerland

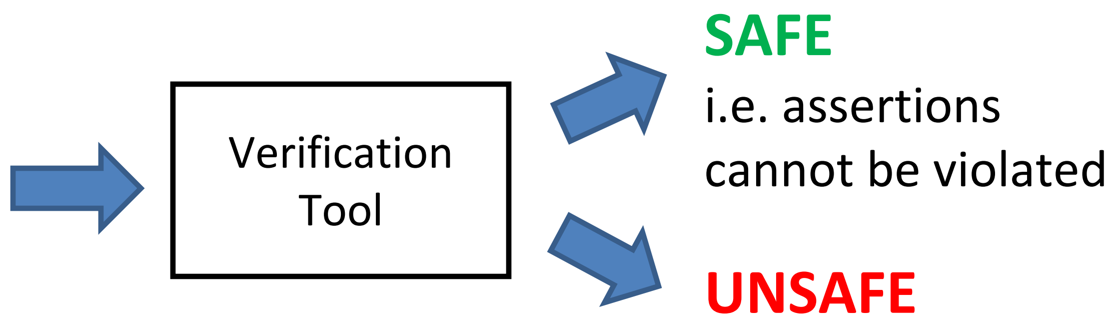
ASE 2008 – L'Aquila – 17 September 2008

Automatic Software Verification

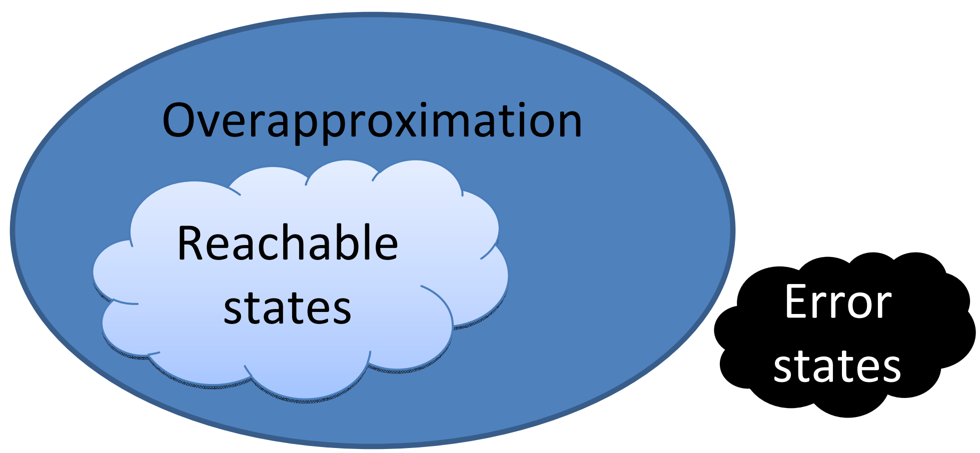
C program

```c
int main() {
  int a = foo();
  int b = bar(a);

  assert(a == b);
}
```



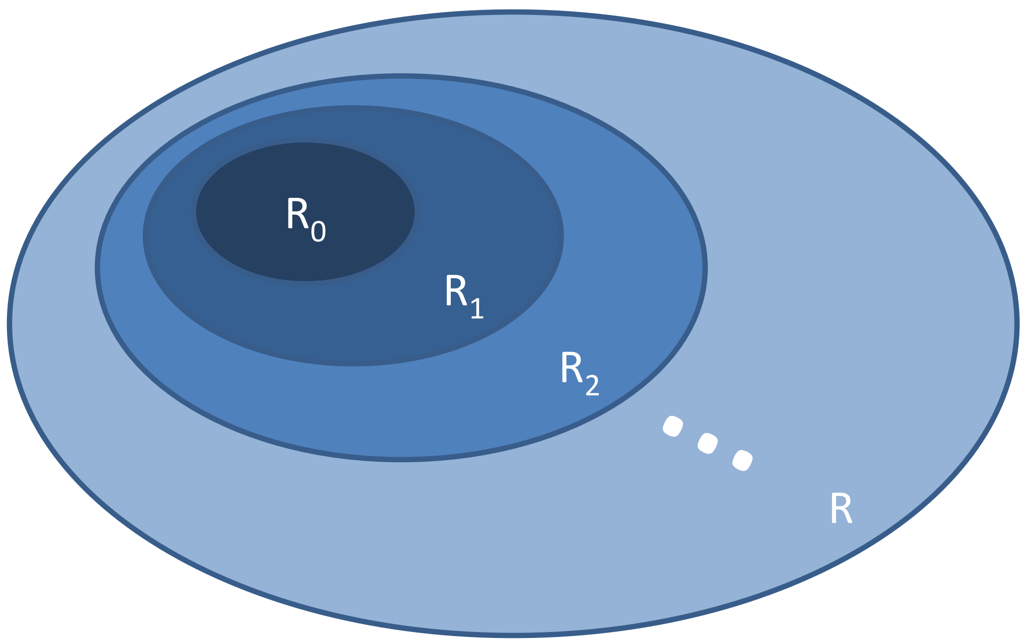
General method:
Create an overapproximation of the program states



Software Verification by Model Checking

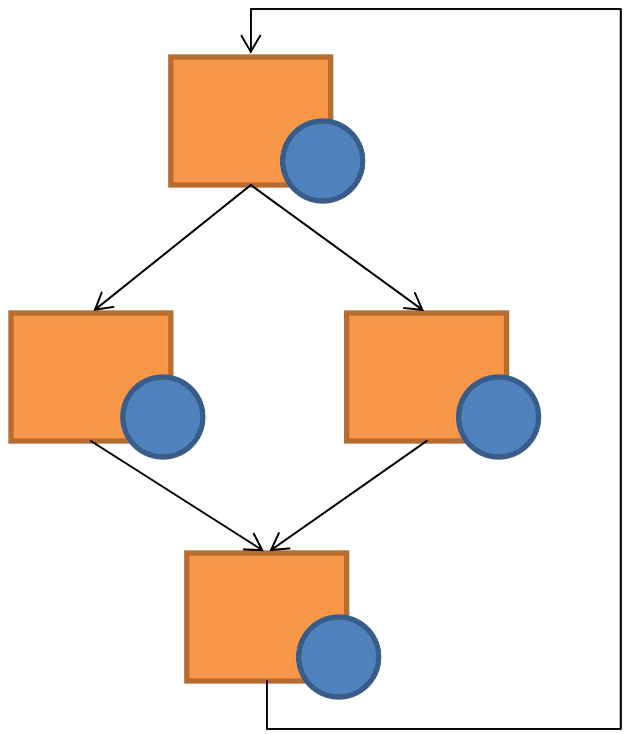
[Clarke/Emerson, Sifakis 1981]

Iterative fixpoint (forward) post computation

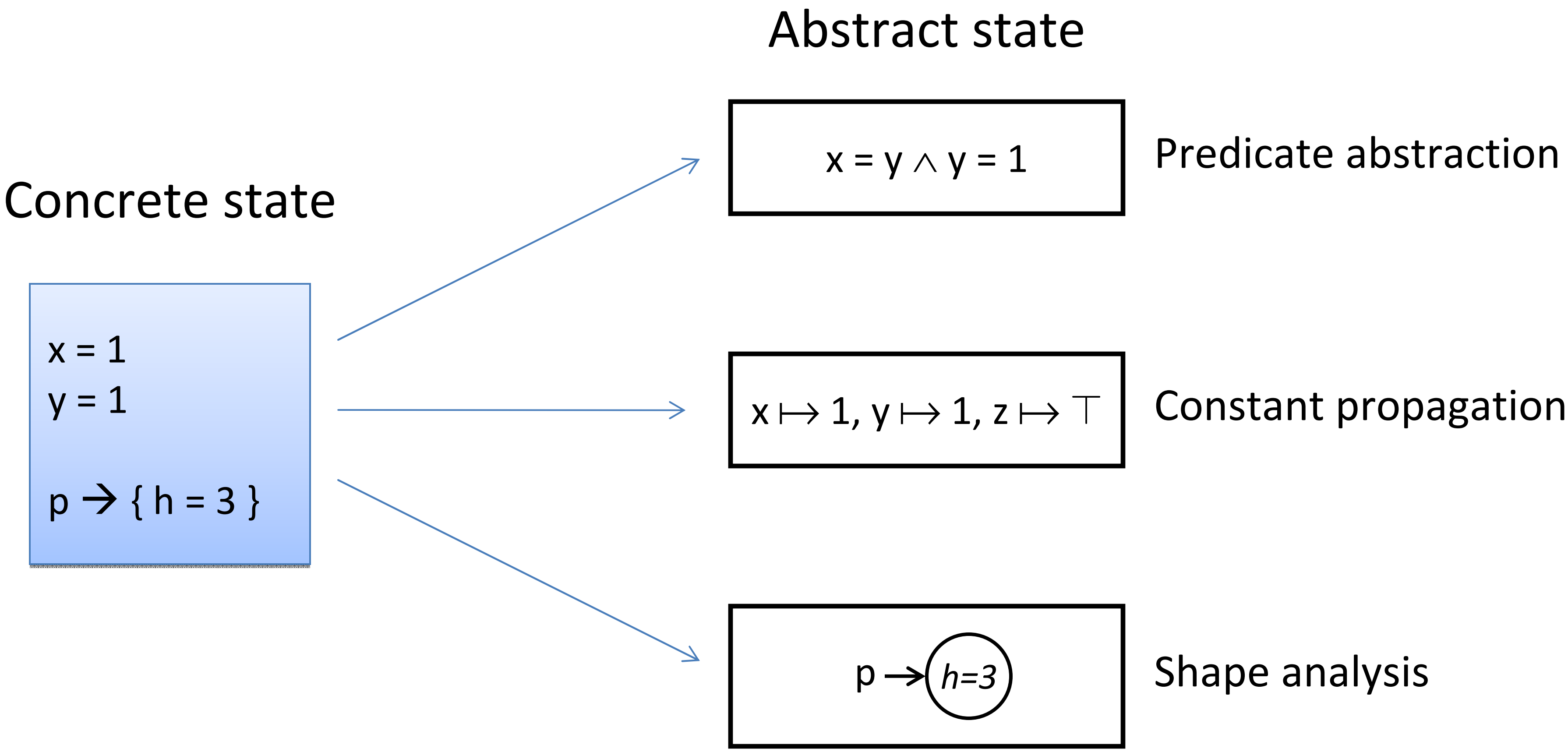


Software Verification by Data-flow Analysis

Fixpoint computation on the CFG



Abstraction



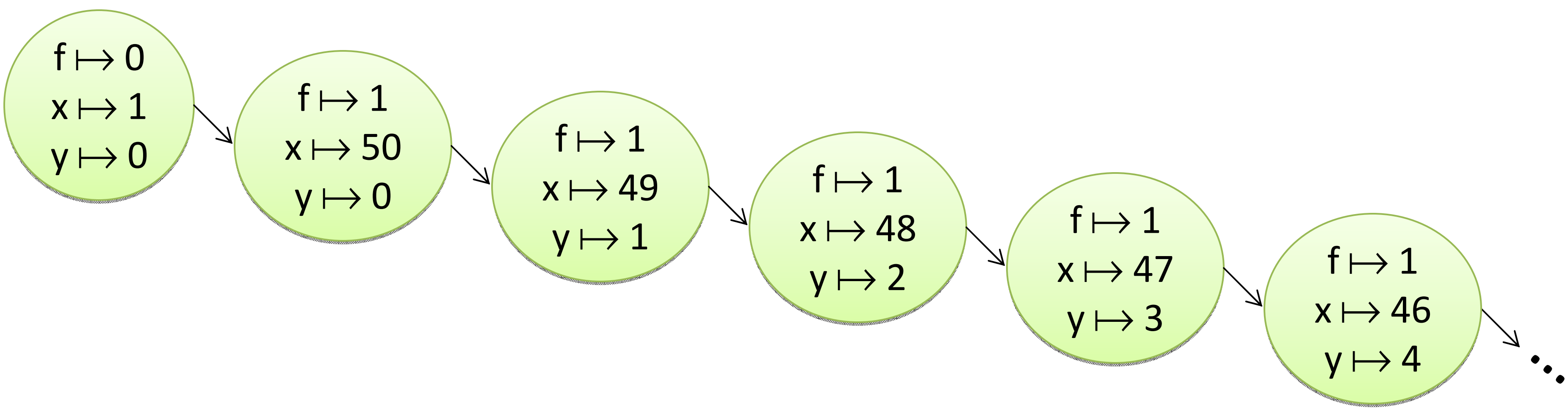
Few explicit values
→ Explicit domain

Many values
→ Predicate abstraction

```
int f = 0, x = 1, y = 0;
while (x > 0) {
  if (f == 0) { x = 50; f = 1; }
  else        { x--; y++; }
}
assert(y == 50);
```

Fully Explicit Analysis

```
int f = 0, x = 1, y = 0;
while (x > 0) {
   if (f == 0) { x = 50; f = 1; }
   else        { x--; y++; }
}
assert(y == 50);
```



\+ cheap to compute

\- many states

```
int f = 0, x = 1, y = 0;
while (x > 0) {
  if (f == 0) { x = 50; f = 1; }
  else        { x--; y++; }
}
assert(y == 50);
```

Combined Analysis

- Start with explicit
- Precision of explicit: threshold on number of diff. values

  E.g. π(x) = 3

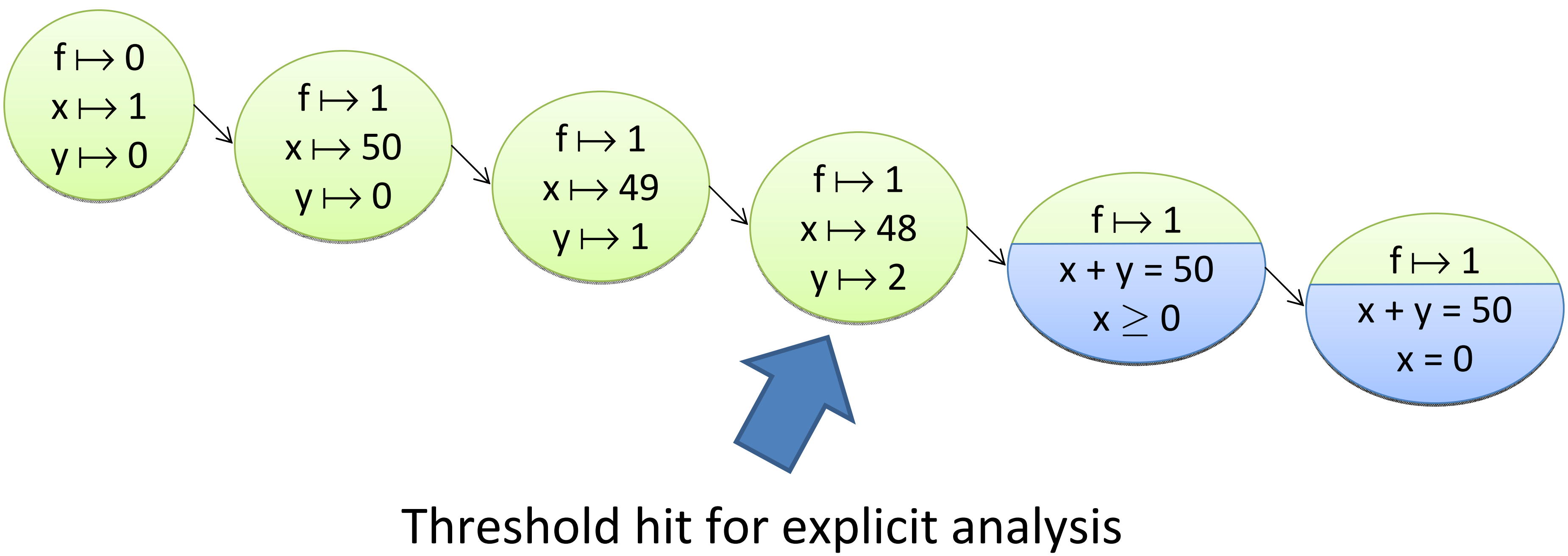
  - Precision of predicate: set of tracked predicates

    E.g. $\pi = \{ x + y = 50,\ x \geq 0, x = 0\}$

  - Switch to predicates when the explicit threshold is hit

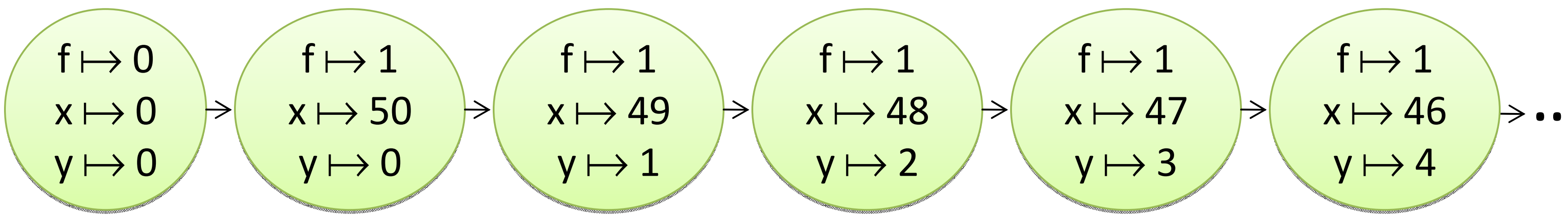
```
int f = 0, x = 1, y = 0;
while (x > 0) {
   if (f == 0) { x = 50; f = 1; }
   else        { x--; y++; }
}
assert(y == 50);
```

Combined Analysis

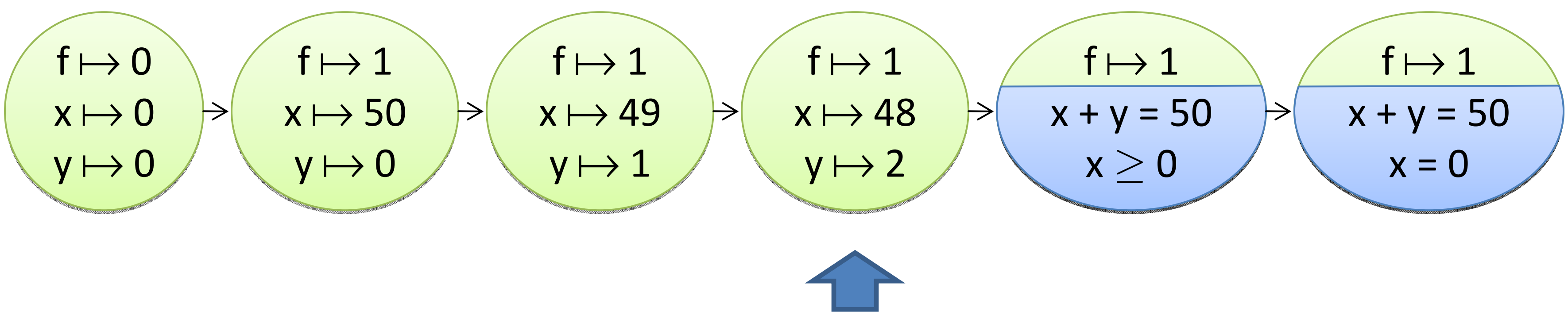


```
int f = 0, x = 0, y = 0;
while (x > 0) {
   if (f == 0) { x = 50; f = 1; }
   else        { x--; y++; }
}
assert(y == 50);
```

Fully Explicit Analysis



Combined Analysis



Motivation

- Flexible combination of abstract domains
- Dynamically update their respective *precisions*
  - precision: set of predicates, variables, etc. to track
  - e.g. switch on/off analyses
  - e.g. use different analyses for different variables
  - …

Software Model Checking

Reached, *Frontier* := { e_0 }

while *Frontier* ≠ ∅ do

  remove *e* from *Frontier*

  for each *e'* ∈ **post**(*e*) do

    if ¬ **stop**(*e'*, *Reached*) add *e'* to *Reached*, *Frontier*

return *Reached*

Configurable Program Analysis
[Beyer/Henzinger/T 2007]

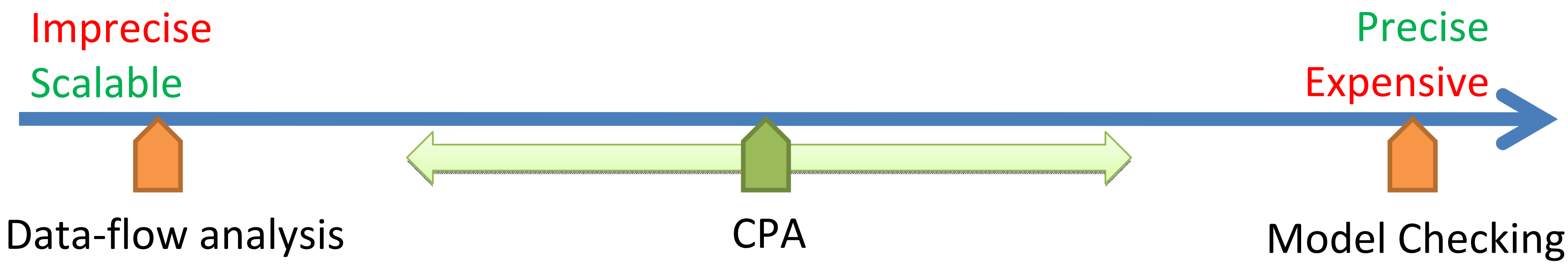
Reached, *Frontier* := { e_0 }
while *Frontier* $\neq \emptyset$ do
  remove *e* from *Frontier*

  for each $e' \in$ **post**(*e*) do
    for each $e'' \in$ *Reached* do
      e''_{new} := **merge**(*e'*, *e''*)
      if $e''_{new} \neq e''$ then
        replace *e''* in *Reached*, *Frontier* by e''_{new}
    if $\neg$ **stop**(*e'*, *Reached*) add *e'* to *Reached*, *Frontier*

return *Reached*

Configurable Program Analysis

- Better combination of abstractions
  ➔ Configurable Program Analysis [CAV07]



Unified framework that enables intermediate algorithms

Configurable Program Analysis
[CAV 2007]

Reached, *Frontier* := { e_0 }
while *Frontier* $\neq \emptyset$ do
  remove *e* from *Frontier*

  for each $e' \in$ **post**(*e*) do
    for each $e'' \in$ *Reached* do
      e''_{new} := **merge**(*e'*, *e''*)
      if $e''_{new} \neq e''$ then
        replace *e''* in *Reached*, *Frontier* by e''_{new}
    if $\neg$ **stop**(*e'*, *Reached*) add *e'* to *Reached*, *Frontier*

return *Reached*

Configurable Program Analysis (CPA+)
with Dynamic Precision Adjustment

[ASE 2008]

Reached, *Frontier* := { (e_0 , π_0) }

while *Frontier* $\neq \emptyset$ do

  remove (e , π) from *Frontier*

  ($\hat{e}$, π_{new}) = **prec**(e, π, *Reached*)

  for each $e' \in$ **post**($\hat{e}$, π_{new}) do

    for each (e'', π'') $\in$ *Reached* do

      e''_{new} := **merge**(e', e'', π_{new})

      if $e''_{new} \neq e''$ then

        replace (e'' , π'') in *Reached*, *Frontier* by (e''_{new} , π_{new})

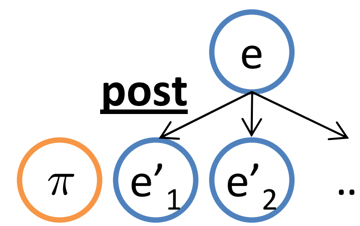
    if $\neg$ **stop**(e', *Reached*, π_{new}) add (e', π_{new}) to *Reached*, *Frontier*

return *Reached*

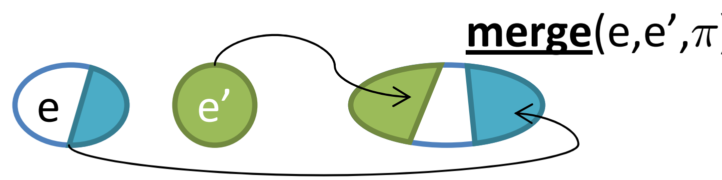
CPA+

Configurable program analysis with dynamic precision adjustment:

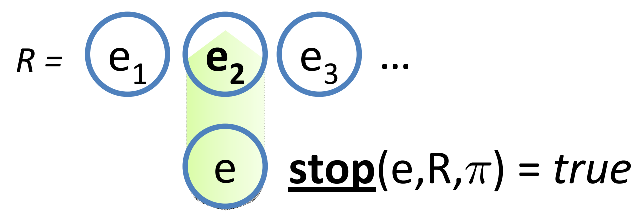
- concrete system $(C, c_0, \rightarrow)$
- abstract domain $(E, \top, \bot, \sqsubseteq, \sqcup)$
- a set of precisions Π
- concretization function $\gamma: E \rightarrow 2^C$
- transfer function **post** $\subseteq E \times \Pi \rightarrow 2^E$



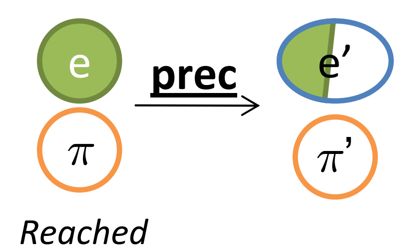
- merge operator **merge**: $E \times E \times \Pi \rightarrow E$



- termination check **stop**: $E \times 2^E \times \Pi \rightarrow \mathbb{B}$



- precision adjustment: **prec**: $E \times \Pi \times 2^{E \times \Pi} \rightarrow E \times \Pi$



Note: Operators are required to be soundly overapproximating

CPA+

Configurable program analysis with dynamic precision adjustment:

- concrete system $(C, c_0, \rightarrow)$
- abstract domain $(E, \top, \bot, \sqsubseteq, \sqcup)$
- a set of precisions Π
- concretization function $\gamma: E \rightarrow 2^C$

- transfer function **post** $\subseteq E \times \Pi \rightarrow 2^E$

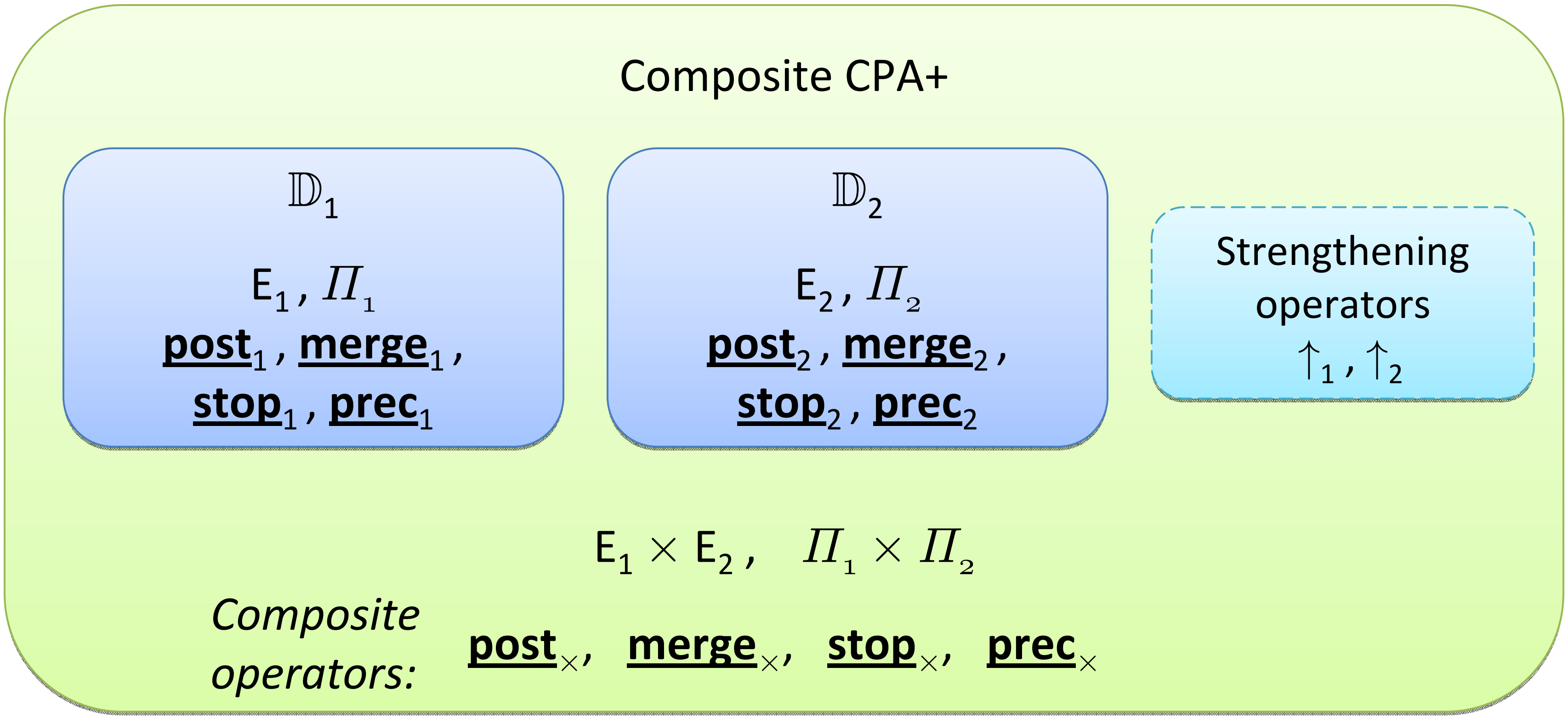
- merge operator **merge**: $E \times E \times \Pi \rightarrow E$
  - $merge^{sep}(e,e',\pi) = e'$
  - $merge^{join}(e,e',\pi) = e \sqcup e'$

- termination check **stop**: $E \times 2^E \times \Pi \rightarrow \mathbb{B}$
  - $stop^{sep}(e,R,\pi) = \exists e' \in R, e \sqsubseteq e'$
  - $stop^{join}(e,R,\pi) = e \sqsubseteq \sqcup R$

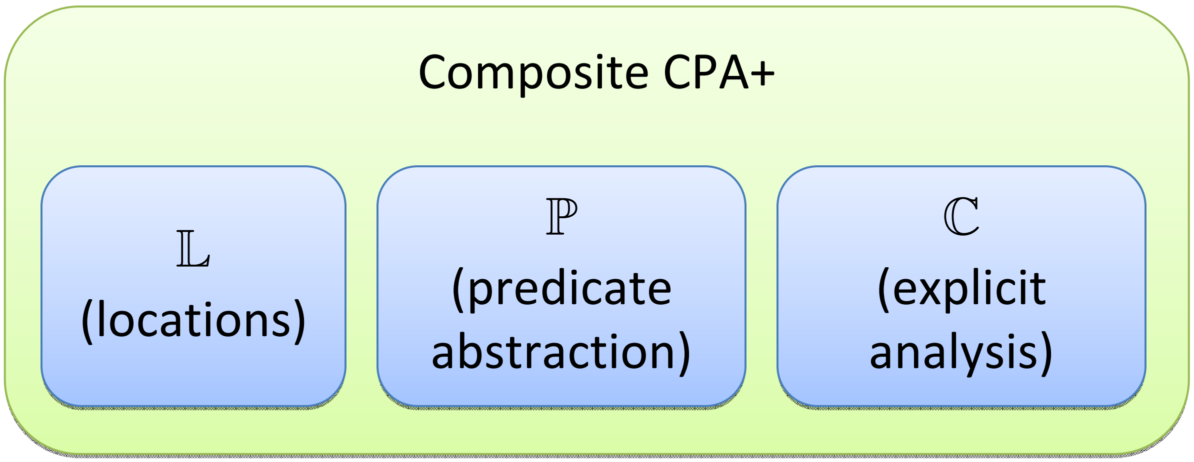
- precision adjustment: **prec**: $E \times \Pi \times 2^{E \times \Pi} \rightarrow E \times \Pi$

Note: Operators are required to be soundly overapproximating

Composite CPA+



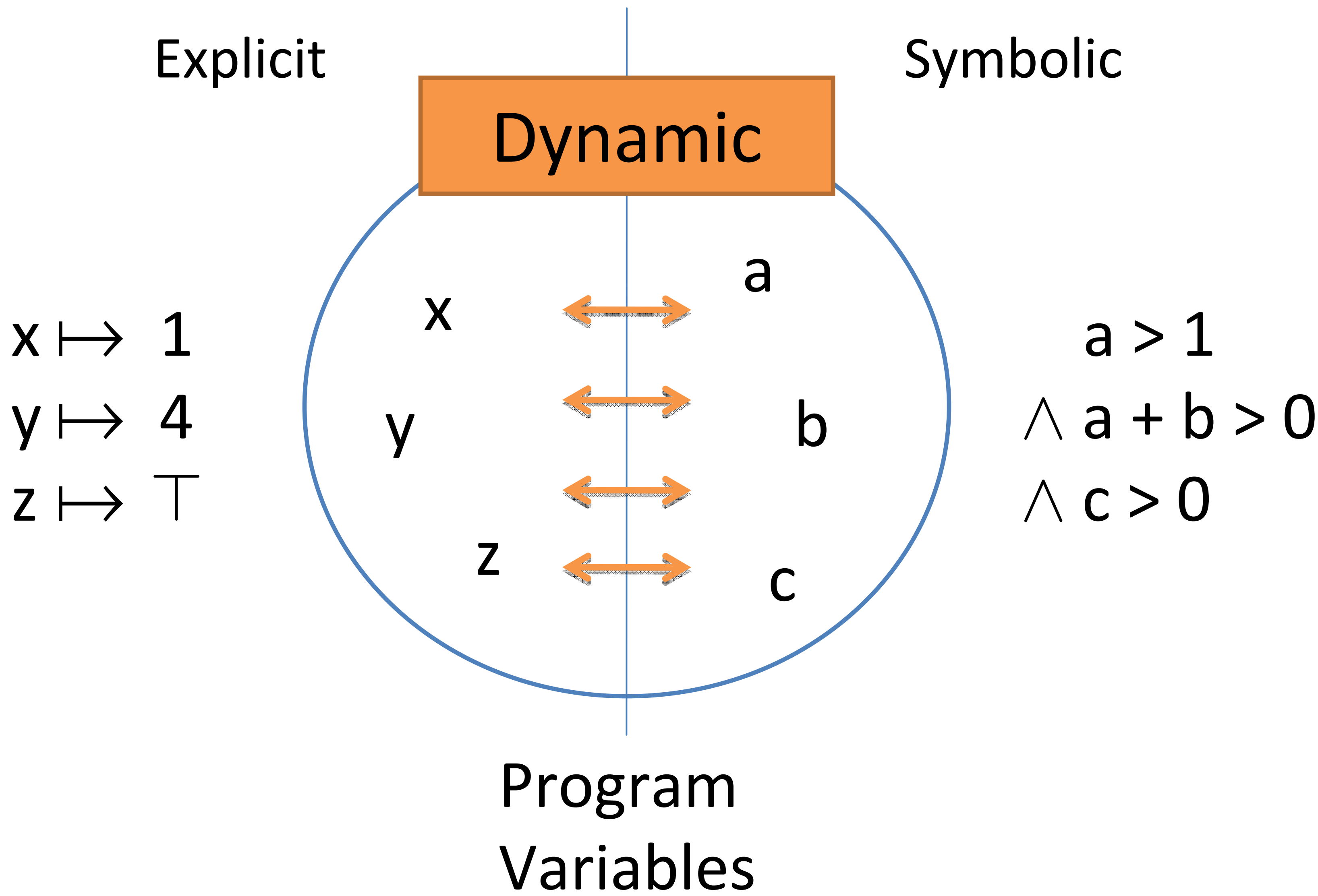
Example: Predicate Abstraction + Explicit



Example of composite abstract element:

$(6 , \quad x > 0 \wedge x = y , \quad \{ i \mapsto 2, x \mapsto \top, y \mapsto \top \})$

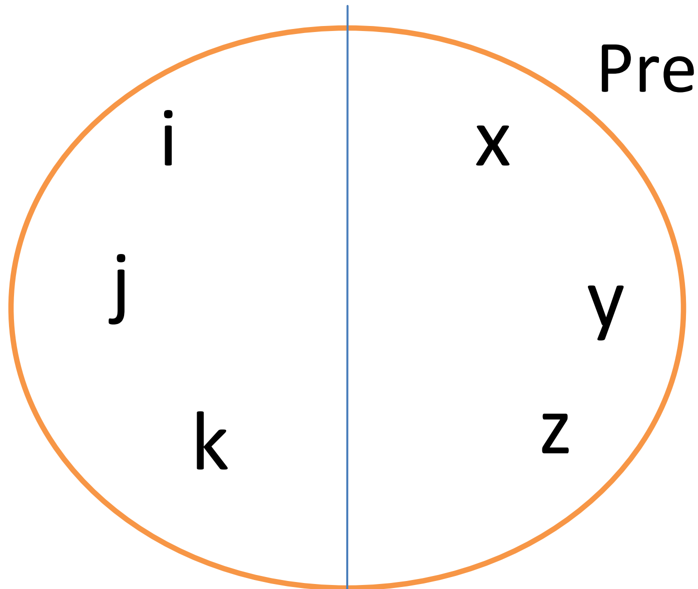
Combined Abstraction



Composite Precision

Explicit

$i \mapsto 1$
$j \mapsto 4$
$k \mapsto \top$



Predicate Abstraction

$x > 1$
$\wedge\ x + y > 0$
$\wedge\ z > 0$

Precision

$\pi = \{ (i,5), (j,3), (k,5), (x,0), (y,0), (z,0) \}$

$\pi = \{ x > 1, z < 7, x + y > 0, z > 0 \}$

Domain and Precisions

CPA+	**Abstract domain**	**Precisions / Precision Adjustment**
$\mathbb{P}$	Predicate Abstraction $E = 2^{\mathcal{P}}$ e.g. $e = \{ x < 3, y > 0 \}$	*Set of tracked predicates* $\Pi = 2^{\mathcal{P}}$ $prec_{\mathbb{P}}(e,\pi,R) = (e,\pi)$
$\mathbb{C}$	Explicit Analysis $E = [X \rightarrow \mathcal{Z}]$ $\mathcal{Z} = \mathbb{Z} \cup \{\perp,\top\}$ e.g. $e = \{ x \mapsto 2, y \mapsto \top, \ldots \}$	*Max. number of diff. values per variable* $\Pi = [X \rightarrow \mathbb{N}]$ $prec_{\mathbb{C}}(e,\pi,R) = (e',\pi)$ if for all $x \in X$: if $\lvert R(x) \rvert \geq \pi(x)$, then $e'(x) = \top$ otherwise, $e'(x) = e(x)$

Example: Predicate Abstraction + Explicit

Idea:

Dynamically choose between explicit & predicate abstraction

- Too many explicit values $\rightarrow$ predicate abstraction

- Use explicit values to infer predicates

Implementable in the composite <u>prec</u> operator

Note: explicit analysis $\approx$ testing on some variables

Few explicit values → Explicit domain

Too many values, or need of symbolic representation → Predicate abstraction

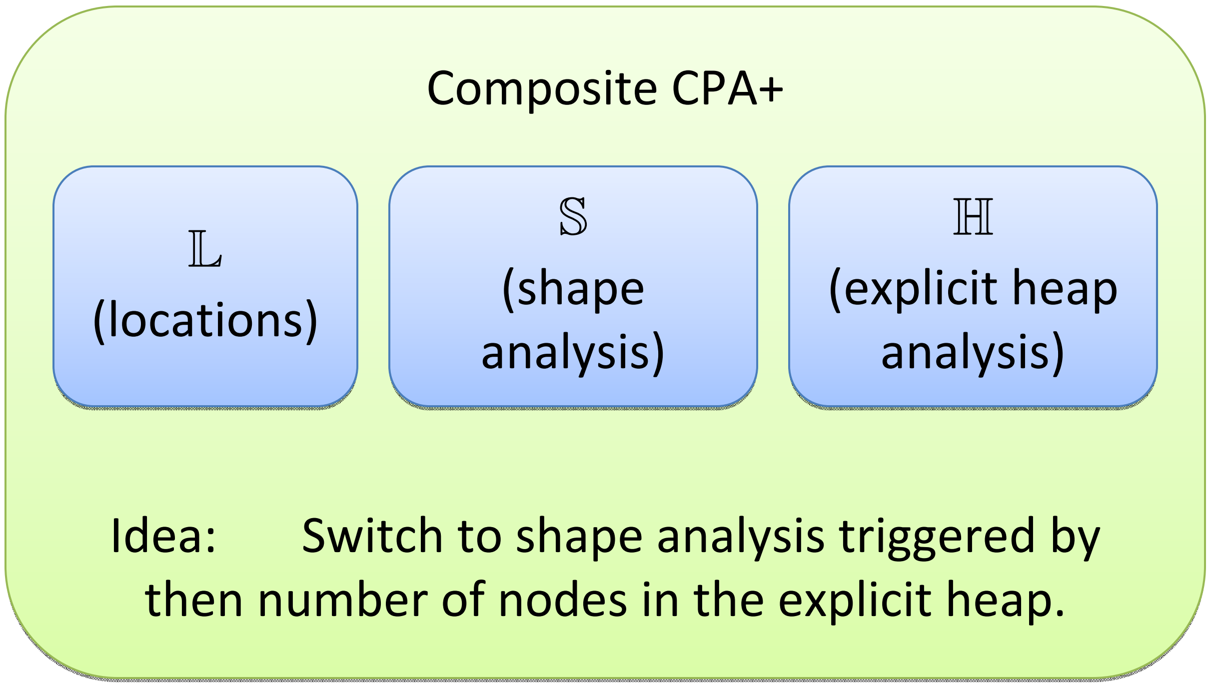
```
int main() {
  int st = 0, ok = 0, cmd = readcmd();
  int *a = getarray(); int n = length(a), p = 0;
  while (1) {
    switch (st) {
      case 0: if (cmd == 177) st = 1; else st = 3; break;
      case 1: st = 3; cmd = readcmd(); break;
      case 3: if (cmd == 78) { ok = 1; goto cont; }
         else { cmd = readcmd(); st = 0; }
      ...
    }
  }
cont:  assert(ok == 1);
  while (p < n) { a[p] = 0; p++; }
  return 0;
}
```

Program Threshold	**Pred. only** k = 0	**Predicate + Explicit** k = 1	k = 5	**Explicit only** k = ∞
ex1	0.46 s	0.17 s	0.21 s	—
ex2	0.43 s	0.16 s	0.21 s	1.00 s
loop1	25.20 s	26.01 s	22.78 s	0.16 s
loop2	279.84 s	277.07 s	258.79 s	0.44 s
square	—	—	—	0.08 s

Program Threshold	**Pred. only** k = 0	**Predicate + Explicit** k = 1	k = 2
s3_clnt (total)	75.11 s	10.40 s	18.42 s
s3_srvr (total)	536.79 s	24.23 s	34.40 s

Precision for explicit analysis: $\pi_{\mathbb{C}}(x) = k$

Another combination: Shapes + Explicit heap



Conclusion

- Unifying Data-Flow Analysis and Model Checking

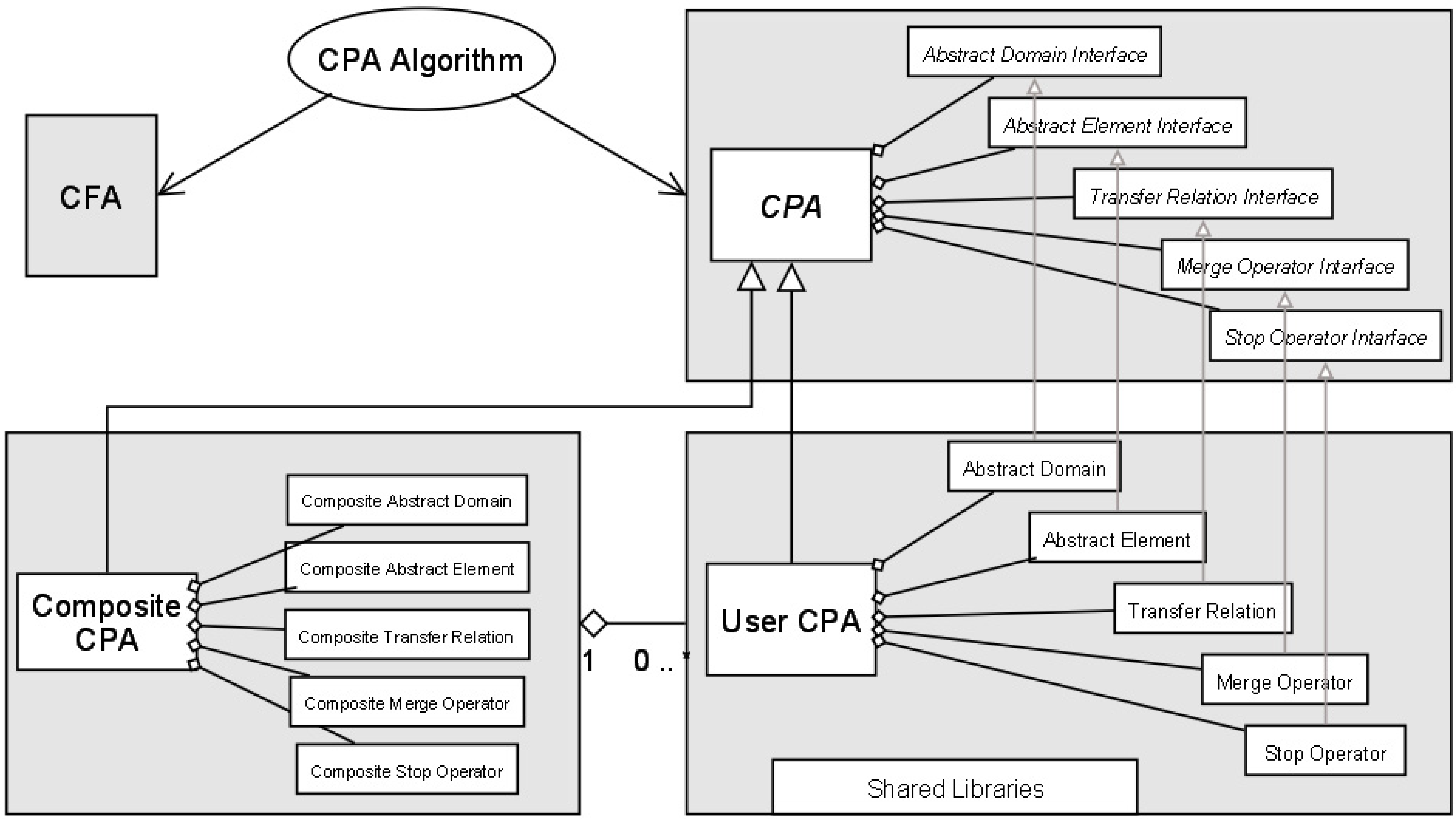
- Framework to express change of precision during the analysis ($\neq$ refinement)
  - → Useful when composing *existing* analyses
  - → Make combination more effective/precise

- References
  - → CAV'07: Configurable Software Verification
  - → ASE'08: PA with Dynamic Precision Adjustment

Current Development: **CPAchecker**

- Tool that implements these concepts

CPAchecker -- Design



CPAchecker -- Architecture and Flow