
LOCAL THEORY EXTENSIONS VIA E-MATCHING

Kshitij Bansal, New York University

Andrew Reynolds, EPFL

Tim King, Verimag

Clark Barrett, New York University

Thomas Wies, New York University

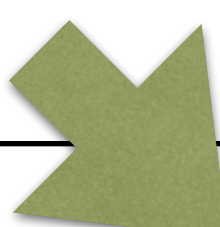
CAV, San Francisco, 23 Jul 2015

```
extern void __VERIFIER_error() __attribute__((__noreturn__));

int main() {
    unsigned int plus_one = 1;
    int minus_one = -1;

    if(plus_one < minus_one) {
        goto ERROR;
    }

    return (0);
    ERROR: __VERIFIER_error();
    return (-1);
}
```



```
(set-logic QF_BV)
(declare-const addr_of_plus_one (_ BitVec 32))
(declare-const plus_one (_ BitVec 32))
(declare-const addr_of_minus_one (_ BitVec 32))
(declare-const minus_one (_ BitVec 32))
(push)
(assert (and (bvult (_ bv1 32) (bvneg (_ bv1 32)))
true))
(check-sat)
```

```
extern void __VERIFIER_error() __attribute__((__noreturn__));

int main() {
    unsigned int plus_one = 1;
    int minus_one = -1;

    if(plus_one < minus_one) {
        goto ERROR;
    }

    return (0);
ERROR: __VERIFIER_error();
    return (-1);
}
```

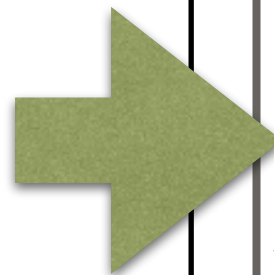


Quantifier-free Bitvector

```
(set-logic QF_BV)
(declare-const addr_of_plus_one (_ BitVec 32))
(declare-const plus_one (_ BitVec 32))
(declare-const addr_of_minus_one (_ BitVec 32))
(declare-const minus_one (_ BitVec 32))
(push)
(assert (and (bvult (_ bv1 32) (bvneg (_ bv1 32)))
true))
(check-sat)
```

BUT OFTEN...

```
/* Concatenate two lists 'a' and 'b'.
 * The result is a single list 'res'. */
procedure concat(a: Node, b: Node)
  returns (res: Node)
  requires lseg(a, null) &*& lseg(b, null)
  ensures lseg(res, null)
{
  if (a == null) {
    return b;
  } else {
    var curr: Node;
    curr := a;
    while (curr.next != null)
      invariant acc(curr) **~ lseg(a, null)
    {
      curr := curr.next;
    }
    curr.next := b;
    return a;
  }
}
```



```
(set-logic UF)
...
(declare-fun BtwN ((Map (Loc Node)
  (Loc Node)) (Loc Node) (Loc Node)
  (Loc Node)) Bool)
...
(assert (forall ((?f (Map (Loc
  Node) (Loc Node))) (?x (Loc Node))
  (?y (Loc Node))) (or (not (= (read
  ?f ?x) ?x)) (not (BtwN ?f ?x ?y ?
  y)) (= ?x ?y))))
...
(assert (or (and (= sk_?XNode_5
  (lseg_footprint next b null))
  (BtwN next b null null)) (not
  (lseg next b null sk_?XNode_5))))
...
(check-sat)
```

BUT OFTEN

Quantified ...

```
/* Concatenate two lists 'a' and 'b'.
 * The result is a single list 'res'. */
procedure concat(a: Node, b: Node)
  returns (res: Node)
  requires lseg(a, null) &
  ensures lseg(res, null)
{
  if (a == null) {
    return b;
  } else {
    var curr: Node;
    curr := a;
    while (curr.next != null)
      invariant acc(curr) ** lseg(a, null)
    {
      curr := curr.next;
    }
    curr.next := b;
    return a;
  }
}
```

$\forall x, y..$

(set-logic UF)

...
(declare-fun BtwN ((Map (Loc Node)
(Loc Node)) (Loc Node) (Loc Node)
(Loc Node)) Bool)

(assert (forall ((?f (Map (Loc
Node) (Loc Node))) (?x (Loc Node))
(?y (Loc Node))) (or (not (= (read
?f ?x) ?x)) (not (BtwN ?f ?x ?y ?
y)) (= ?x ?y))))))

...
(assert (or (and (= sk_?XNode_5
(lseg_footprint next b null))
(BtwN next b null null)) (not
(lseg next b null sk_?XNode_5))))

...
(check-sat)

THIS WORK

- Local theory extensions [*Sofronie-Stokkermans, 2005*]
 - How to use existing SMT solvers for a complete decision procedure
 - Improvements in the solvers for better performance
-

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

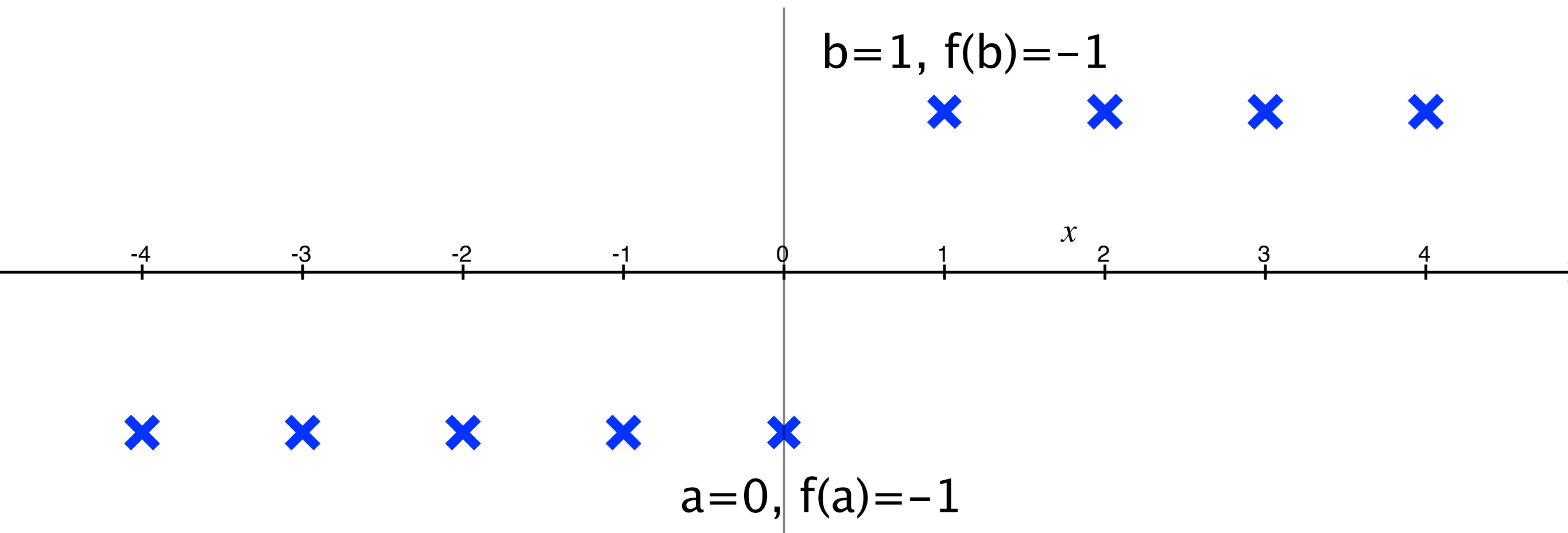
$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic. $f : \mathbb{Z} \rightarrow \mathbb{Z}$ monotonically increasing.

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic. $f : \mathbb{Z} \rightarrow \mathbb{Z}$ monotonically increasing.

SAT: $a = 0, b = 1, f(x) = \{-1 \text{ if } x \leq 0, 1 \text{ if } x > 0\}$



$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic. $f : \mathbb{Z} \rightarrow \mathbb{Z}$ monotonically increasing.

$$\text{SAT: } a = 0, b = 1, f(x) = \{-1 \text{ if } x \leq 0, 1 \text{ if } x > 0\}$$

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic. $f : \mathbb{Z} \rightarrow \mathbb{Z}$ monotonically increasing.

$$\text{SAT: } a = 0, b = 1, f(x) = \{-1 \text{ if } x \leq 0, 1 \text{ if } x > 0\}$$

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

Local if sufficient to instantiate such that all terms already exist in G or K.

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic. $f : \mathbb{Z} \rightarrow \mathbb{Z}$ monotonically increasing.

$$\text{SAT: } a = 0, b = 1, f(x) = \{-1 \text{ if } x \leq 0, 1 \text{ if } x > 0\}$$

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$K\sigma_1 = a \leq b \implies f(a) \leq f(b) \text{ where } \sigma_1 = \{x \mapsto a, y \mapsto b\}$$

$$K\sigma_2 = b \leq a \implies f(b) \leq f(a) \text{ where } \sigma_2 = \{x \mapsto b, y \mapsto a\}$$

$$K\sigma_3 = a \leq a \implies f(a) \leq f(a) \text{ where } \sigma_3 = \{x \mapsto a, y \mapsto a\}$$

$$K\sigma_4 = b \leq b \implies f(b) \leq f(b) \text{ where } \sigma_4 = \{x \mapsto b, y \mapsto b\}.$$

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic.

$$\begin{aligned} K\sigma_1 &= a \leq b \implies f(a) \leq f(b) \text{ where } \sigma_1 = \{x \mapsto a, y \mapsto b\} \\ K\sigma_2 &= b \leq a \implies f(b) \leq f(a) \text{ where } \sigma_2 = \{x \mapsto b, y \mapsto a\} \\ K\sigma_3 &= a \leq a \implies f(a) \leq f(a) \text{ where } \sigma_3 = \{x \mapsto a, y \mapsto a\} \\ K\sigma_4 &= b \leq b \implies f(b) \leq f(b) \text{ where } \sigma_4 = \{x \mapsto b, y \mapsto b\}. \end{aligned}$$

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

Theory of linear arithmetic.

$G \cup K[G]$ is satisfiable in LIA
if and only if
 G is satisfiable in LIA+ K

$$K[G] \left\{ \begin{array}{l} K\sigma_1 = a \leq b \implies f(a) \leq f(b) \text{ where } \sigma_1 = \{x \mapsto a, y \mapsto b\} \\ K\sigma_2 = b \leq a \implies f(b) \leq f(a) \text{ where } \sigma_2 = \{x \mapsto b, y \mapsto a\} \\ K\sigma_3 = a \leq a \implies f(a) \leq f(a) \text{ where } \sigma_3 = \{x \mapsto a, y \mapsto a\} \\ K\sigma_4 = b \leq b \implies f(b) \leq f(b) \text{ where } \sigma_4 = \{x \mapsto b, y \mapsto b\}. \end{array} \right.$$

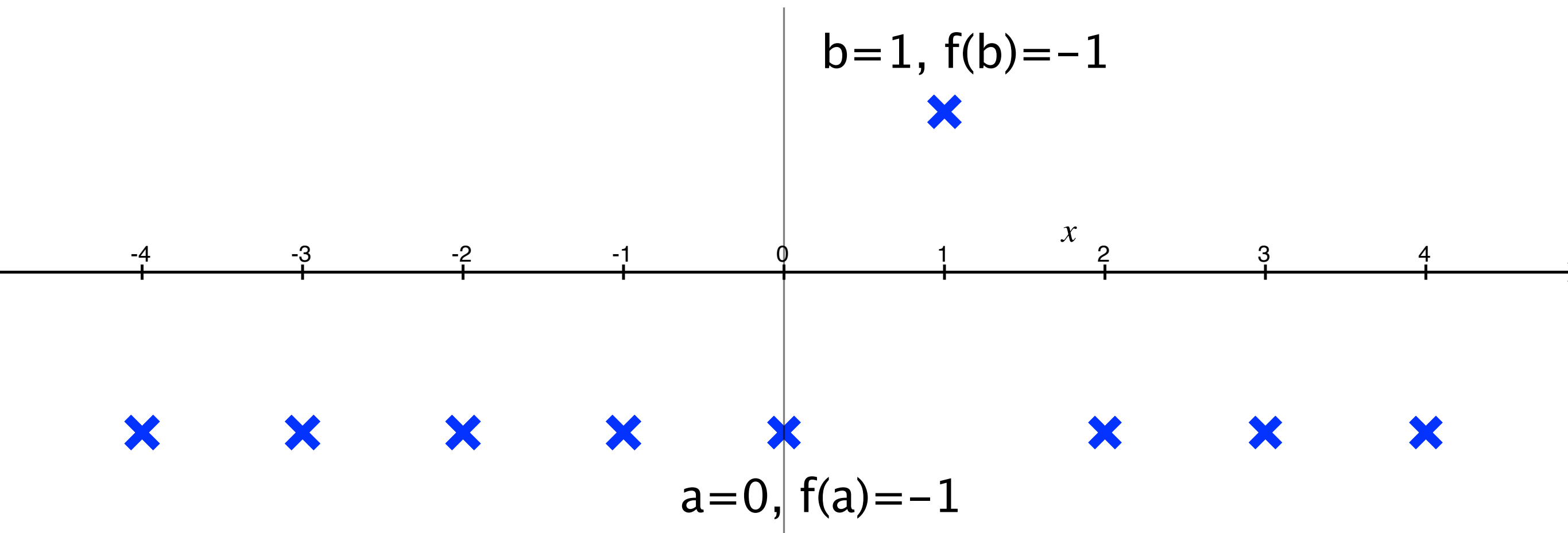
$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

$$K[G] = \{K\sigma_1, K\sigma_2, K\sigma_3, K\sigma_4\}$$

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

$$K[G] = \{K\sigma_1, K\sigma_2, K\sigma_3, K\sigma_4\}$$

$$a = 0, b = 1, f(x) = \{-1 \text{ if } x = 0, 1 \text{ if } x = 1, -1 \text{ otherwise}\}$$



$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

$$K[G] = \{K\sigma_1, K\sigma_2, K\sigma_3, K\sigma_4\}$$

$$a = 0, b = 1, f(x) = \{-1 \text{ if } x = 0, 1 \text{ if } x = 1, -1 \text{ otherwise}\}$$

$$a = 0, b = 1, f(x) = \{-1 \text{ if } x = 0, 1 \text{ if } x = 1, \text{undefined otherwise}\}$$

Restrict

$$b=1, f(b)=-1$$



x



$$a=0, f(a)=-1$$

$$G = \{a + b = 1, f(a) + f(b) = 0\}$$

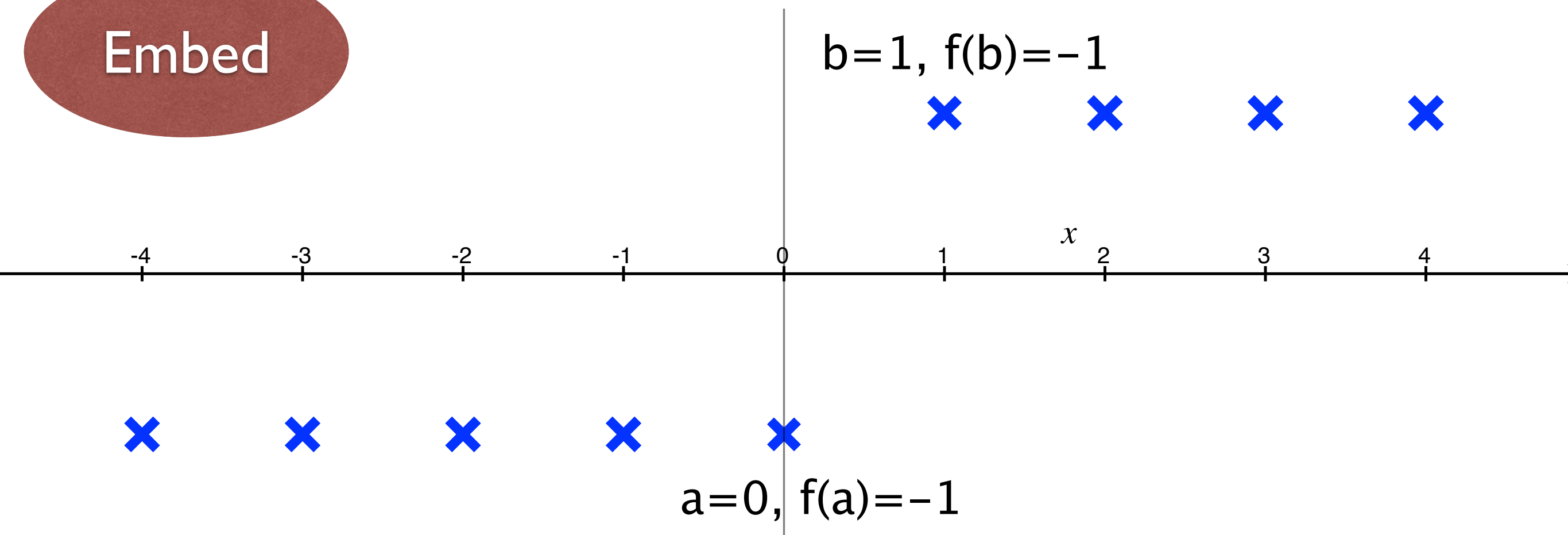
$$K[G] = \{K\sigma_1, K\sigma_2, K\sigma_3, K\sigma_4\}$$

$$a = 0, b = 1, f(x) = \{-1 \text{ if } x = 0, 1 \text{ if } x = 1, -1 \text{ otherwise}\}$$

$$a = 0, b = 1, f(x) = \{-1 \text{ if } x = 0, 1 \text{ if } x = 1, \text{undefined otherwise}\}$$

Can be embedded in full model of LIA+K

Embed



EXAMPLES

- Local theory extensions — more general than EPR
 - Array property fragment [*Bradley, Manna, Sipma, 2006*]
 - Theory of reachability in linked lists
[*Lahiri, Qadeer, 2006; Rakamafić, Bingham, Hu, 2007*]
 - Theory of finite sets and multisets [*Zarba, 2004; Zarba 2002*]
-

E-MATCHING

Nelson, 1980; Detlefs, Nelson, Saxe, 2005; deMoura, Bjørner, 2007

- **input:**
 - a set of terms G
 - a set of ground equalities E ($t_1 \approx t_2$).
 - patterns P (e.g. $f(x)$)

 - **output:**
 - The set of substitutions σ over the variables in p , modulo E , such that:
for all $p \in P$ there exists a $t \in G$ with $E \models t \approx p\sigma$.
-

E-M

Nelson, 19

$$G = \{a, b, c, f(a), f(b), f(c)\}$$
$$E = \{a \approx b\}$$
$$P = \{f(x), f(y)\}$$

■ **input:**

- a set of terms G
- a set of ground equalities E ($t_1 \approx t_2$).
- patterns P (e.g. $f(x)$)

■ **output:**

- The set of substitutions σ over the variables in p , modulo E , such that:
for all $p \in P$ there exists a $t \in G$ with $E \models t \approx p\sigma$.

E-MA

Nelson, 1980

$$G = \{a, b, c, f(a), f(b), f(c)\}$$

$$E = \{a \approx b\}$$

$$P = \{f(x), f(y)\}$$

■ input:

- a set of terms G
- a set of ground equations E
- patterns P (e.g.

$$\{x \rightarrow a, y \rightarrow a\},$$

$$\{x \rightarrow a, y \rightarrow c\},$$

$$\{x \rightarrow c, y \rightarrow a\},$$

$$\{x \rightarrow c, y \rightarrow c\}.$$

■ output:

- The set of substitutions σ over the variables in p , modulo E , such that:
for all $p \in P$ there exists a $t \in G$ with $E \models t \approx p\sigma$.

EXAMPLE

φ :

$$a + b \approx 1$$

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

φ :

$$a + b \approx 1$$

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

Terms: $a, b, c, d, f(a), f(b), f(c), 0, 1$

φ :

$$a + b \approx 1$$

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

Terms: $a, b, c, d, f(a), f(b), f(c), 0, 1$

φ :

$$a + b \approx 1$$

Externally solve: Instantiate such that all terms already exist in G or K.

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

Terms: $a, b, c, d, f(a), f(b), f(c), 0, 1$

φ :

$$a + b \approx 1$$

Externally solve: Instantiate such that all terms already exist in G or K.

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

$$\{x \rightarrow a, b, c\} \times \{y \rightarrow a, b, c\}$$

Not $d, 0, 1$ as $f(.)$ not in G or K.

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

φ :

$$a + b \approx 1$$

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

SAT Solver

Core

Base theory
Solvers



EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

φ :

$$a + b \approx 1$$

$$\wedge (f(a) + f(b) \approx 0 \vee f(b) + f(c) \approx 0)$$

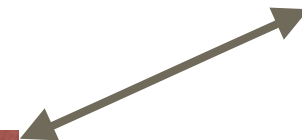
$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$

SAT Solver

Core

Base theory
Solvers



EXAMPLE

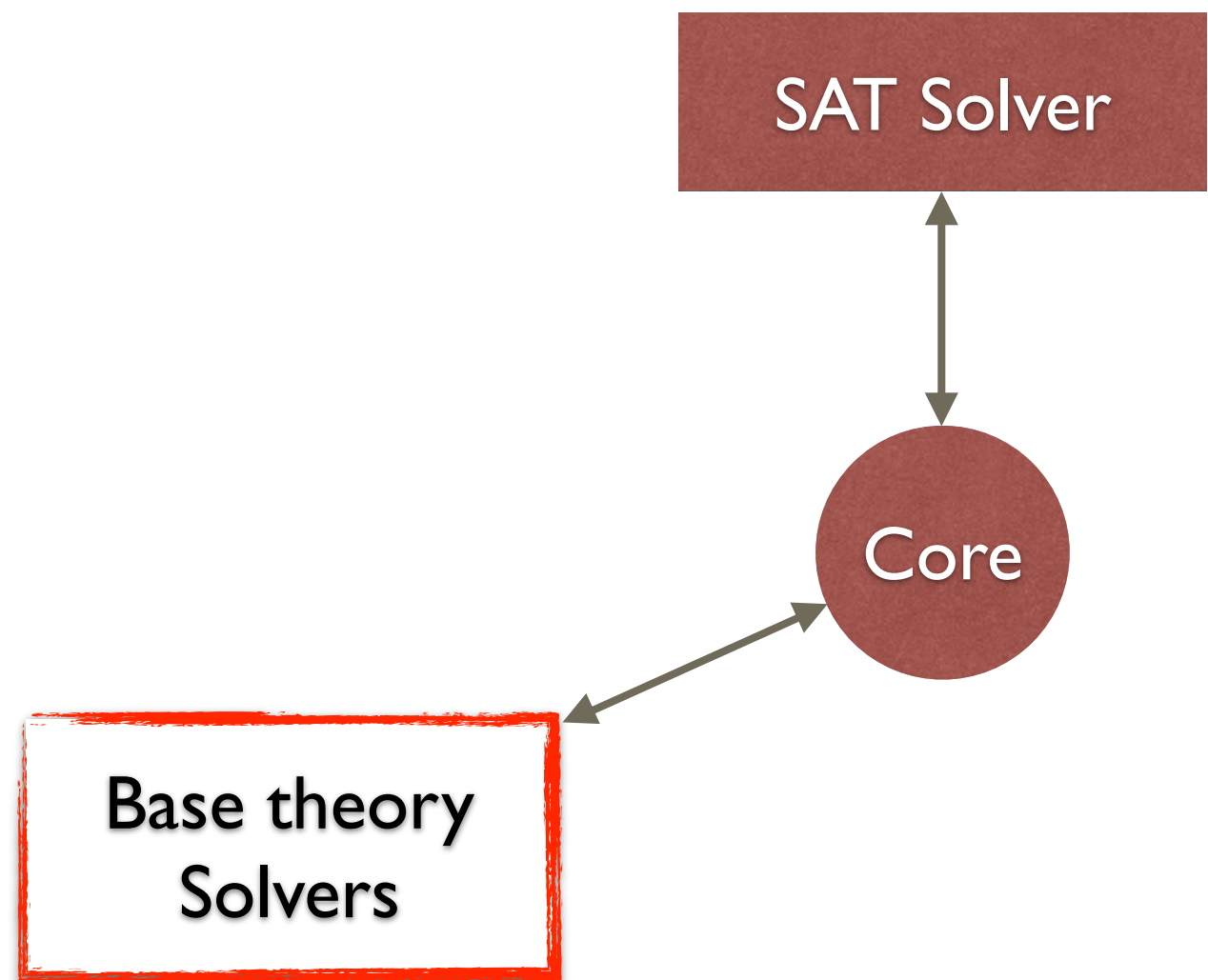
$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$a + b \approx 1$$

$$\wedge f(a) + f(b) \approx 0$$

$$\wedge a + c \approx b + d$$

$$\wedge c \approx d.$$



EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$a + b \approx 1$$

$$\wedge f(a) + f(b) \approx 0$$

$$\left. \begin{array}{l} \wedge a + c \approx b + d \\ \wedge c \approx d. \end{array} \right\} a \approx b$$

Base theory
Solvers

SAT Solver

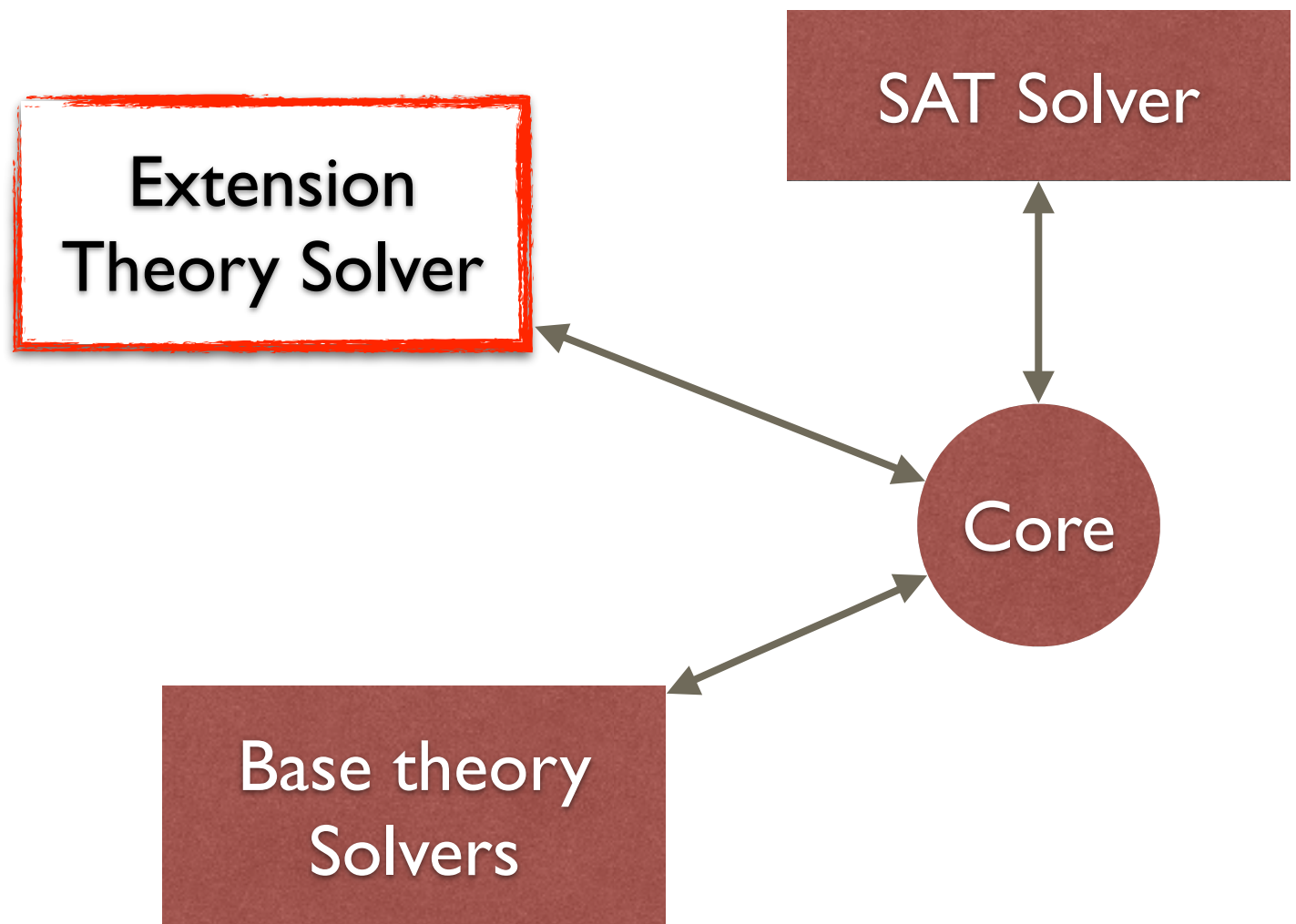
Core



EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$a + b \approx 1,$
 $f(a) + f(b) \approx 0,$
 $a + c \approx b + d,$
 $c \approx d,$
 $a \approx b.$



EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

SAT Solver

Core

$$\begin{aligned} K\sigma_1 &= a \leq b \implies f(a) \leq f(b) \text{ where } \sigma_1 = \{x \mapsto a, y \mapsto b\} \\ K\sigma_2 &= b \leq a \implies f(b) \leq f(a) \text{ where } \sigma_2 = \{x \mapsto b, y \mapsto a\} \\ K\sigma_3 &= a \leq a \implies f(a) \leq f(a) \text{ where } \sigma_3 = \{x \mapsto a, y \mapsto a\} \\ K\sigma_4 &= b \leq b \implies f(b) \leq f(b) \text{ where } \sigma_4 = \{x \mapsto b, y \mapsto b\}. \end{aligned}$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

SAT Solver

Core

$$\begin{aligned} K\sigma_1 &= a \leq b \implies f(a) \leq f(b) \text{ where } \sigma_1 = \{x \mapsto a, y \mapsto b\} \\ K\sigma_2 &= b \leq a \implies f(b) \leq f(a) \text{ where } \sigma_2 = \{x \mapsto b, y \mapsto a\} \\ K\sigma_3 &= a \leq a \implies f(a) \leq f(a) \text{ where } \sigma_3 = \{x \mapsto a, y \mapsto a\} \\ K\sigma_4 &= b \leq b \implies f(b) \leq f(b) \text{ where } \sigma_4 = \{x \mapsto b, y \mapsto b\}. \end{aligned}$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

SAT Solver

Core

$$K\sigma_1 = a \leq b \implies f(a) \leq f(b) \text{ where } \sigma_1 = \{x \mapsto a, y \mapsto b\}$$

$$K\sigma_2 = b \leq a \implies f(b) \leq f(a) \text{ where } \sigma_2 = \{x \mapsto b, y \mapsto a\}$$

$$K\sigma_3 = a \leq a \implies f(a) \leq f(a) \text{ where } \sigma_3 = \{x \mapsto a, y \mapsto a\}$$

$$K\sigma_4 = b \leq b \implies f(b) \leq f(b) \text{ where } \sigma_4 = \{x \mapsto b, y \mapsto b\}.$$

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

E-matching

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

$$G = \{a, b, c, d, a+c, b+d, \\ 0, 1, f(a), f(b)\}$$

E-matching

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

$$\begin{aligned} G &= \{a, b, c, d, a+c, b+d, \\ &\quad 0, 1, f(a), f(b)\} \\ E &= \{a+c \approx b+d, c \approx d, a \approx b\} \end{aligned}$$

E-matching

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

$$G = \{a, b, c, d, a+c, b+d, \\ 0, 1, f(a), f(b)\}$$

$$E = \{a+c \approx b+d, c \approx d, a \approx b\}$$

$$P = \{f(x), f(y)\}$$

E-matching

EXAMPLE

$$K = \forall x, y. x \leq y \implies f(x) \leq f(y)$$

$$\begin{aligned} a + b &\approx 1, \\ f(a) + f(b) &\approx 0, \\ a + c &\approx b + d, \\ c &\approx d, \\ a &\approx b. \end{aligned}$$

Extension
Theory Solver

$$\begin{aligned} G &= \{a, b, c, d, a+c, b+d, \\ &\quad 0, 1, f(a), f(b)\} \\ E &= \{a+c \approx b+d, c \approx d, a \approx b\} \\ P &= \{f(x), f(y)\} \end{aligned}$$

E-matching

$$\begin{aligned} x &\rightarrow a, \\ y &\rightarrow a \end{aligned}$$

1

ALGORITHM

Input: $\varphi, \mathcal{K}, Z, G, E$

Local variable: $Z' = \{\}$

1. For each K in \mathcal{K} :

1. **Define patterns** P to be the function symbols in K containing variables.

2. Run **E-matching** algorithm with input (E, G, P) . Obtain **substitutions** S .

3. For each $\sigma \in S$, if there exists no $K\sigma'$ in Z such that $\sigma \sim_E \sigma'$, then **add** **$K\sigma$** to Z' .

2. If Z' is empty, return sat, else return Z' .

ALGORITHM

Input: $\varphi, \mathcal{K}, Z, G, E$
Local variable: $Z' = \{\}$

Handled by incremental E-matching procedures, which are well-studied, already implemented in SMT Solvers

1. For each K in \mathcal{K} :

1. **Define patterns** P to be the function symbols in K containing variables.

2. Run **E-matching** algorithm with input (E, G, P) . Obtain **substitutions** S .

3. For each $\sigma \in S$, if there exists no $K\sigma'$ in Z such that $\sigma \sim_E \sigma'$, then **add** **$K\sigma$** to Z' .

2. If Z' is empty, return sat, else return Z' .

ALGORITHM

- Minimal work while using existing solvers to get complete decision procedure.
 - Solver improvements if told axioms encode local theory extension
 - Complete, stop search early when SAT
 - Further optimizations (see Section 6 in paper)
 - Can be extended to Psi-local extensions (see Section 5 in paper)
-

EXPERIMENTS



Benchmarks: generated by Grasshopper
[Piskac, Wies, Zufferey, 2013; 2014]

UFLIA

|

Graph Reachability and Stratified Sets

|

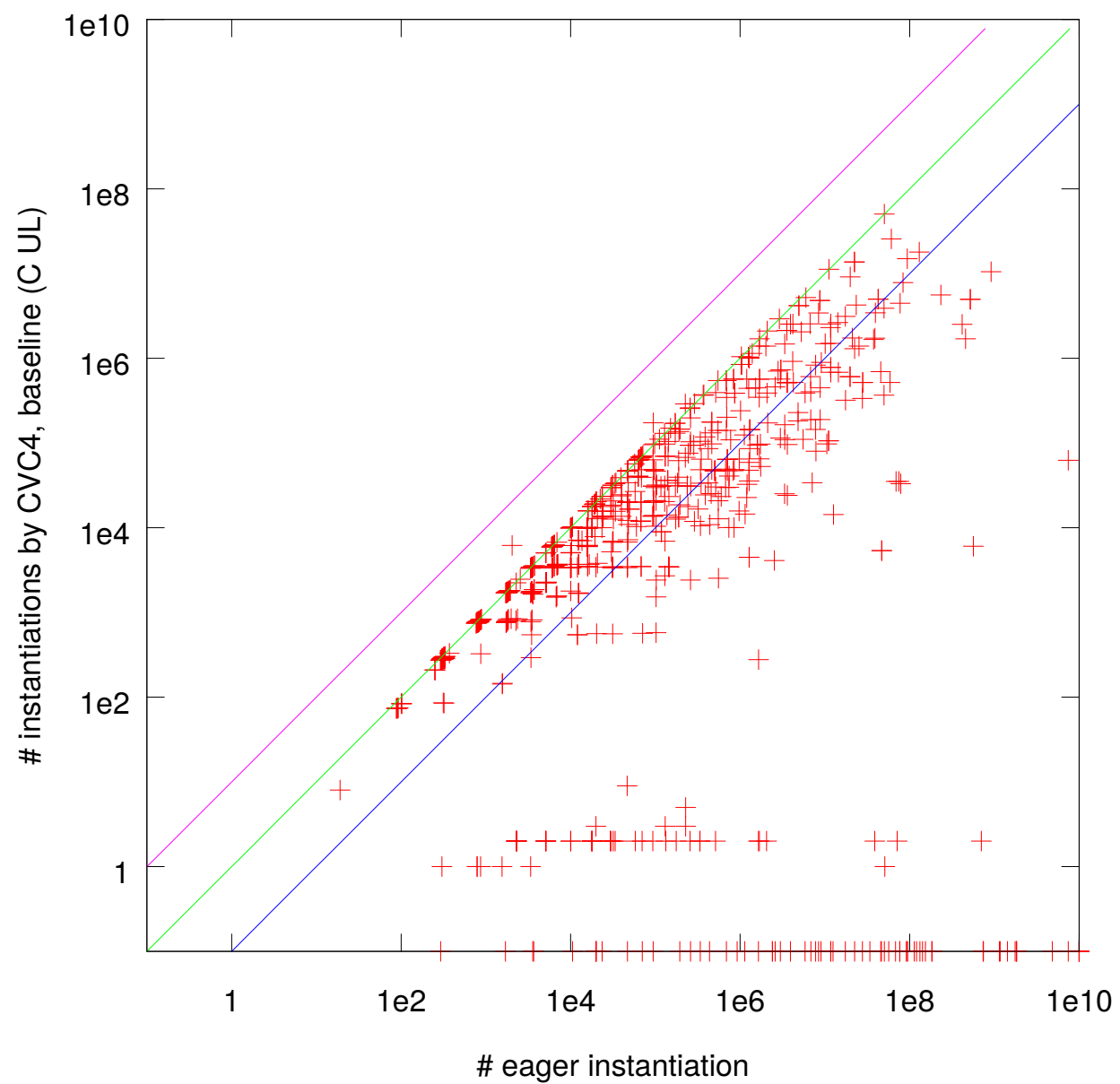
Frame axioms

|

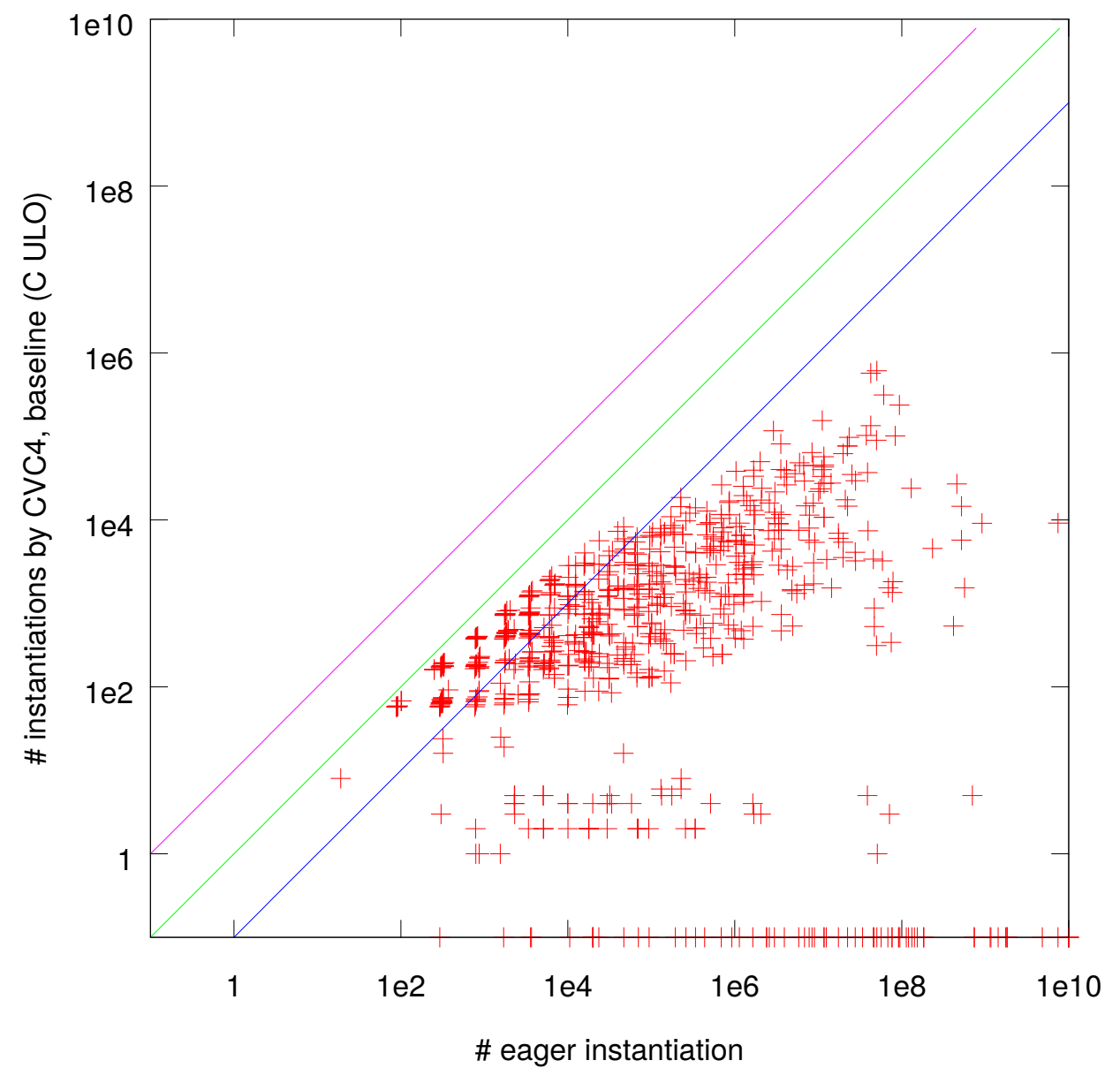
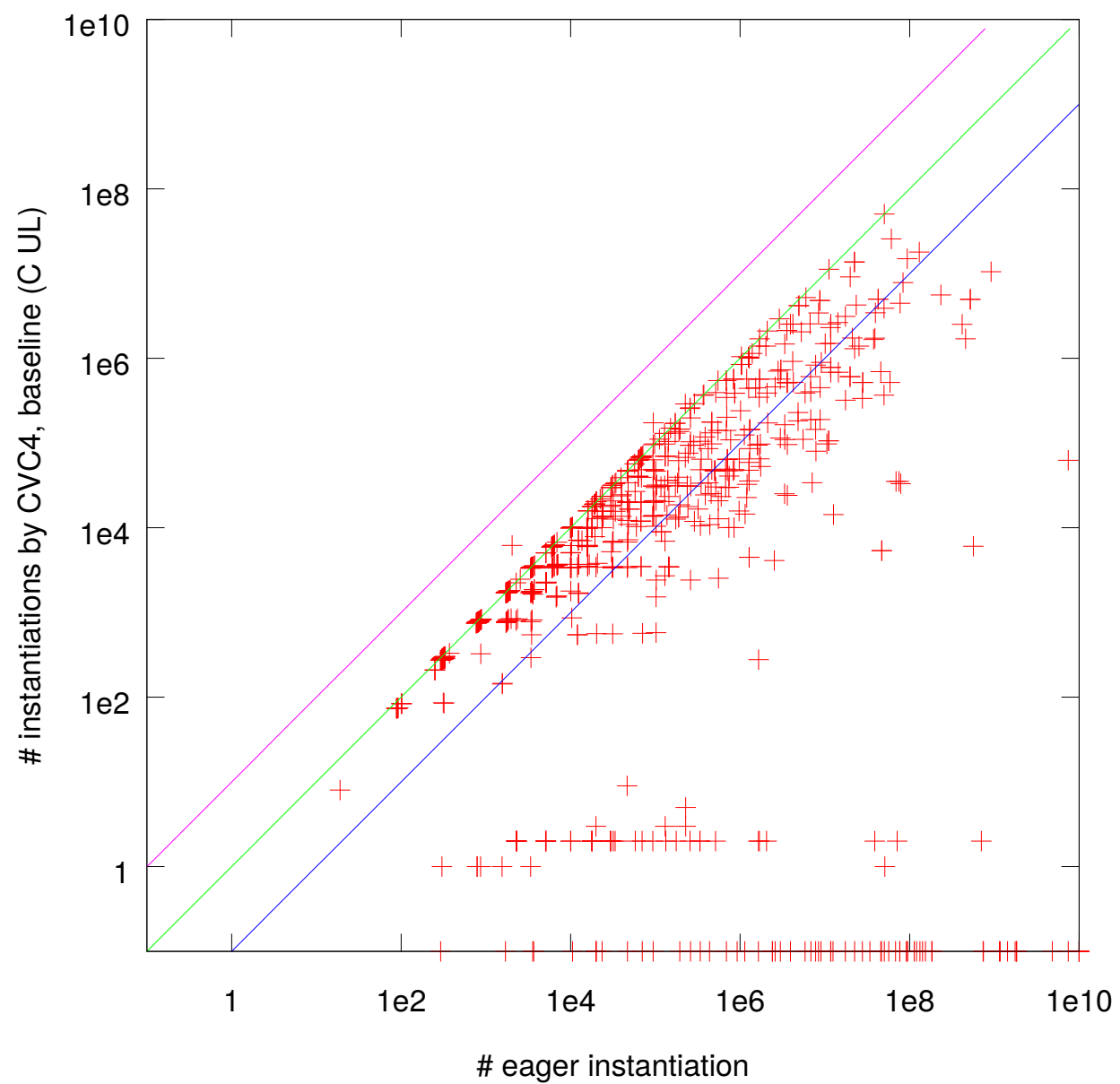
Program specific extensions

SMT Solvers: CVC4 & Z3

EXPERIMENT I



EXPERIMENT I



EXPERIMENT 2

family	#	C #	UD time
sl lists	139	127	70
dl lists	70	66	1717
sl nested	63	63	1060
sls lists	208	181	6046
trees	243	229	2121
soundness	79	76	17
sat	14	-	-
total	816	742	11032

EXPERIMENT 2

family	#	C #	UD time	C #	UL time
sl lists	139	127	70	139	383
dl lists	70	66	1717	70	843
sl nested	63	63	1060	63	307
sls lists	208	181	6046	204	11230
trees	243	229	2121	228	22042
soundness	79	76	17	79	1533
sat	14	-	-	14	670
total	816	742	11032	797	37009

EXPERIMENT 2

family	#	C #	UD time	C #	UL time	C #	ULO time
sl lists	139	127	70	139	383	139	17
dl lists	70	66	1717	70	843	70	33
sl nested	63	63	1060	63	307	63	13
sls lists	208	181	6046	204	11230	208	3401
trees	243	229	2121	228	22042	239	7187
soundness	79	76	17	79	1533	79	70
sat	14	-	-	14	670	14	12
total	816	742	11032	797	37009	812	10732

EXPERIMENT 2

family	#	C #	UD time	C #	UL time	C #	ULO time	Z3 #	UD time	Z3 #	UL time	Z3 #	ULO time
sl lists	139	127	70	139	383	139	17	138	1955	138	1950	139	68
dl lists	70	66	1717	70	843	70	33	56	11375	56	11358	70	2555
sl nested	63	63	1060	63	307	63	13	52	6999	52	6982	59	1992
sls lists	208	181	6046	204	11230	208	3401	182	20596	182	20354	207	4486
trees	243	229	2121	228	22042	239	7187	183	41208	183	40619	236	27095
soundness	79	76	17	79	1533	79	70	76	7996	76	8000	79	336
sat	14	-	-	14	670	14	12	-	-	10	3964	14	898
total	816	742	11032	797	37009	812	10732	687	90130	697	93228	804	37430

EXPERIMENT 3

family	#	C #	PL time	C #	PLO time	Z3 #	PM time	Z3 #	PL time	Z3 #	PLO time
sl lists	139	139	664	139	20	139	9	139	683	139	29
dl lists	70	70	3352	70	50	70	41	67	12552	70	423
sl nested	63	63	2819	63	427	63	182	56	7068	62	804
sls lists	208	206	14222	207	3086	208	37	203	17245	208	1954
trees	243	232	7185	243	6558	243	663	222	34519	242	8089
soundness	79	78	156	79	49	79	23	79	2781	79	39
sat	14	14	85	14	22	13	21	12	1329	14	109
total	816	802	28484	815	10213	815	976	778	76177	814	11447

Comparison of solvers on partially instantiated benchmarks

BIBLIOGRAPHY

- de Moura, L., Bjørner, N.S.: *Efficient e-matching for SMT solvers*. CADE 2007.
 - Ge, Y., de Moura, L.: *Complete instantiation for quantified formulas in satisfiability modulo theories*. CAV 2009.
 - Ihlemann, C., Sofronie-Stokkermans, V.: *System description: H-PILoT*. CADE 2009.
 - Nelson, C.G.: *Techniques for Program Verification*. Ph.D. thesis, 1980.
 - Piskac, R., Wies, T., Zufferey, D.: *Automating Separation Logic Using SMT*. CAV 2013.
 - Sofronie-Stokkermans, V.: *Hierarchic reasoning in local theory extensions*. CADE 2005.
-

CONCLUSION

- Algorithm for deciding local theory extensions using E-matching
 - Uses existing SMT solvers: simple syntactic modifications to input
For users: <http://cs.nyu.edu/~kshitij/localtheories/>
 - Explored additional optimizations for SMT solvers
 - Future directions: combining with model-based instantiation techniques.
-