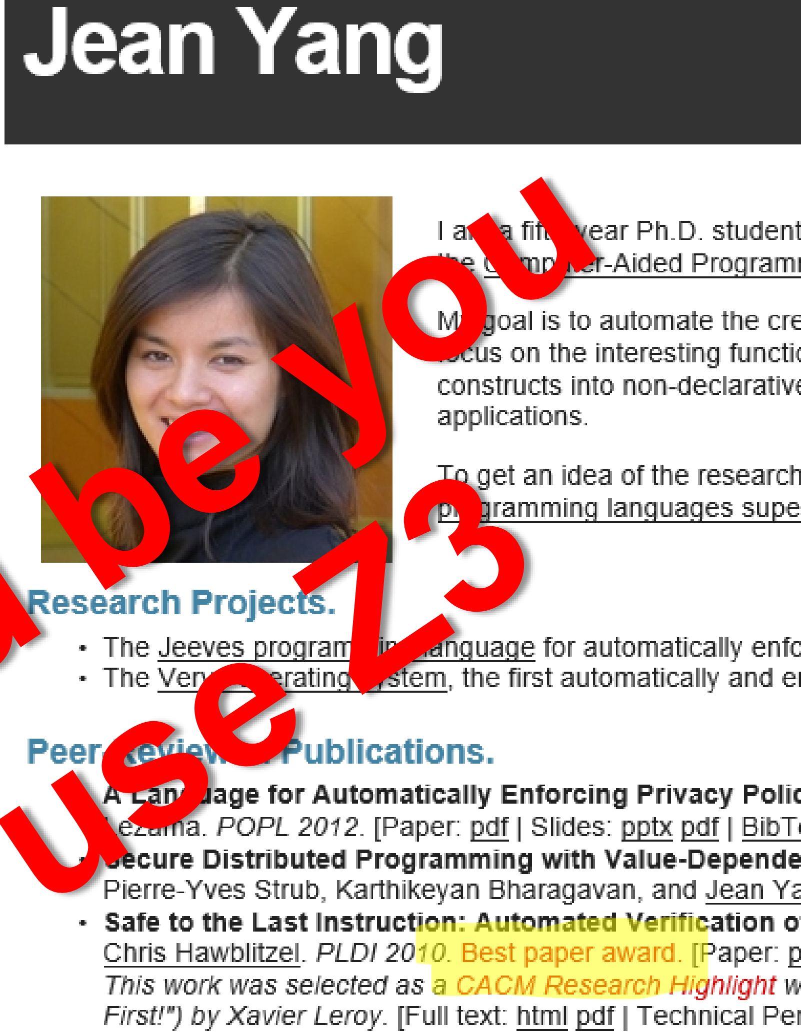
νZ – Maximal Satisfaciton with Z3

Nikolaj Bjørner
Microsoft Research

Phan Anh Dung
DTU

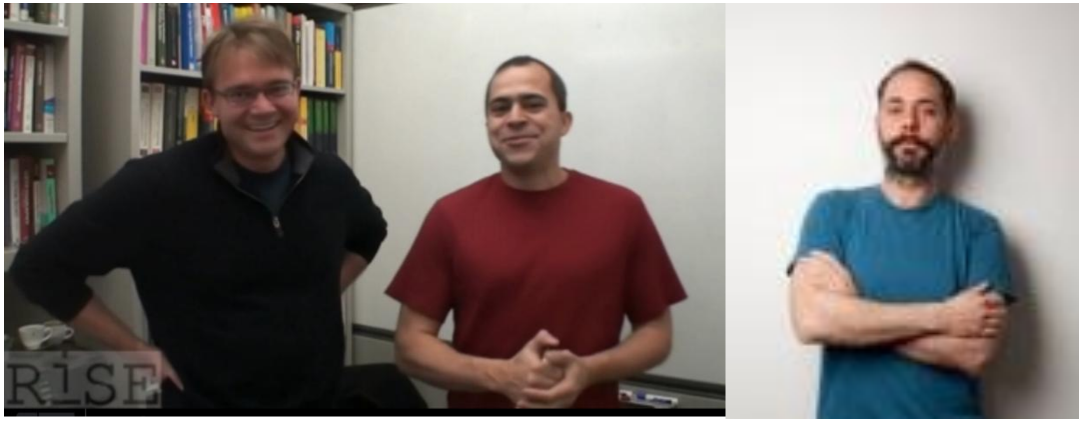
First some

Z3 Propaganda



This could be you
if you use Z3

Z3 – Backed by Proof Plumbers



Leonardo de Moura, Nikolaj Bjørner, Christoph Wintersteiger

Symbolic Analysis with Z3

Solution/Model

$$x^2 + y^2 < 1 \; and \; xy > 0.1 \Rightarrow \text{sat, } \underbrace{x = \frac{1}{8}, y = \frac{7}{8}}$$

$$x^2 + y^2 < 1 \; and \; xy > 1 \Rightarrow \text{unsat, Proof}$$

Is execution path *P* feasible?

SAGE

W
I
T
N

Z3 solved more than **10 billion** constraints created by SymEx tools including SAGE checking Win8 and Office

ti

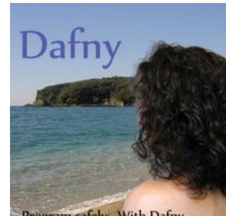
Does Policy Satisfy Contract?

Z3 used by Pex, Static Driver Verifier, many other tools

Pex

SLAM

Symbolic Analysis Engines



SecGuru

FORMULA
Modeling Foundations.

Vcc
Yogi
SLAM
SAGE
Pex
TERMINATOR
SLAYER
HAVOC
BOOGIE

SLS, floats

νZ: Opt+MaxSMT

μZ: Datalog

Generalized PDR

Existential Reals

Model Constructing SAT

CutSAT: Linear Integer Formulas

Quantified Bit-Vectors

Linear Quantifier Elimination

Model Based Quantifier Instantiation

Generalized, Efficient Array Decision Procedures

Engineering DPLL(T) + Saturation

Effectively Propositional Logic

Model-based Theory Combination.

Relevancy Propagation

Efficient E-matching for SMT solvers

Z3 Internals

SecGuru in WANetmon

Cluster dc/dm/cluster/dm1prdstr08

40,000 ACL checks per month
Each check 50-200ms
20 bugs/month (mostly for build-out)

Network ACL Validation Alerts for the cluster

This check validates the correctness of all the network ACLs in the devices in the cluster

		Device	Timestamp	Result
⌄	!	dm1-x3hl-cis-15-01	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacifi	

ACL Name	IP Address Range	Error
mgmt-only	10.143.197.208/28	Partially bl
mgmt-only	10.143.197.224/27	Partially
mgmt-only	10.143.198.0/26	Partially blocked
mgmt-only	10.143.198.64/27	Partially blocked
mgmt-only	10.143.198.96/28	Partially blocked
ssh-only	10.143.197.208/28	Blocked
ssh-only	10.143.197.224/27	Blocked

		Device	Timestamp	Result
⌃	!	dm1-x3hl-cis-15-03	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacifi	
⌃	!	dm1-x3hl-cis-15-04	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacifi	

Cluster dc/dm/cluster/dm1prdstr01

Network ACL Validation Alerts for the cluster

This check validates the correctness of all the network ACLs in the devices in the cluster

		Device	Timestamp	Result
		is-1-03	Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
			ep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔		14 2013 09:18:00 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3	14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis	Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-c 1-08	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-09	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-10	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-11	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-12	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-13	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-14	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-15	Sat Sep 14 2013 09:18:00 GMT-0700 (Pacific Daylight Time)	Success
⌃	✔	dm1-x3hl-cis-1-16	Sat Sep 14 2013 11:27:41 GMT-0700 (Pacific Daylight Time)	Success

Verifying Forwarding Rules with SecGuru

Routes

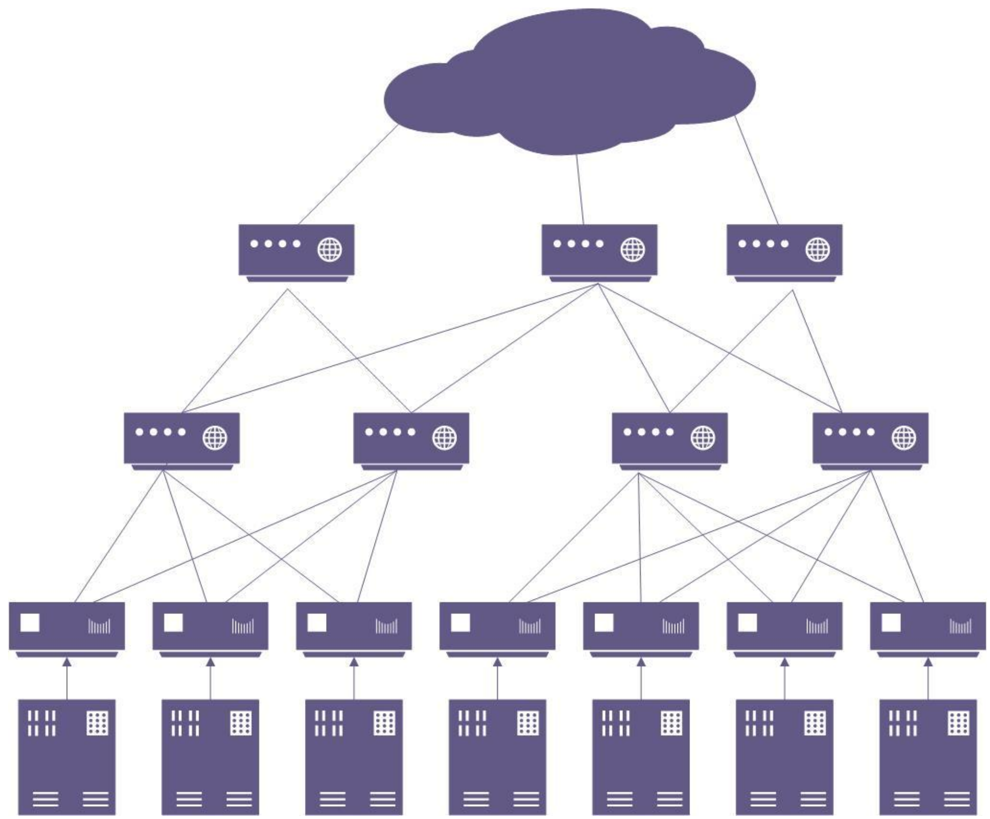
```
B E    0.0.0.0/0 [200/0] via 100.91.176.0, n1
                         via 100.91.176.2, n2

B E    10.91.114.0/25 [200/0] via 100.91.176.125, n3
                              via 100.91.176.127, n4
                              via 100.91.176.129, n5
                              via 100.91.176.131, n6
B E    10.91.114.128/25 [200/0] via 100.91.176.125, n3
                                via 100.91.176.131, n6
                                via 100.91.176.133, n7
...
```

Logic

$Router \equiv$
$\quad$ **if** $\ldots$
$\quad$ **if** $dst = 10.91.114.128/25$ **then** $n_3 \vee n_6 \vee n_7$ **else**
$\quad$ **if** $dst = 10.91.114.0/25$ **then** $n_3 \vee n_4 \vee n_5 \vee n_6$ **else**
$\quad n_1 \vee n_2$

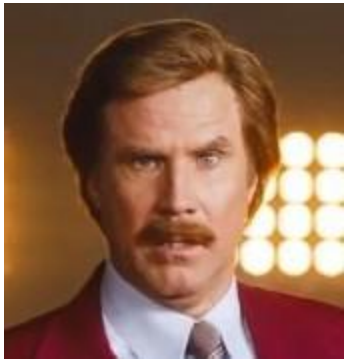
Contract

$$Cluster(dst) \Rightarrow \\ Router_1(dst) \equiv Router_2(dst)$$



Now on to

Z3 + Optimization



My νZ is bigger than your μZ!

- *νZ is a Z3 module for SMT with objective functions*

Vampire tuning

Theorem prover tuning

DNA alignment

- *A new tool for Z3 user scenarios*

RNA synth

Multi Proc. Code

Longest Paths for static analysis

Max profit goods packing

Chip Design for Energy Saving

Materials discovery

Introducing νZ

- SMT users want/need many things:
  - Solve classes of quantifiers (Horn, EPR)
  - Solve non-linear, transcendental arithmetic
  - Solve floating point numbers
  - Solve string and grammar constraints
  - Solve faster, bigger, better, …

- Sometimes some solutions are better than others.

Introducing νZ

- νZ integrates objective functions
  - νZ = SMT + optimization algorithms
  - Is meant to make objectives accessible to Z3 users
  - Based on CDCL technologies

- νZ is soliciting applications. It is "soon" integrated with main Z3 branch, but so-far available from a branch called "opt".

Roadmap

- νZ functionality
- Optimizing Solvers
- νZ for Linear Arithmetic
- νZ and MaxSAT
- Combining Objectives

νZ functionality

- Maximize, Minimize
  - Terms over integers, reals, bit-vectors
- Weighted Soft Constraints
- Combine Multiple Objectives
  - Lexicographic, Pareto, Box

A Transportation Planning Example

	Weight (kg)	Volume (m^3)	Delivery date	Firm delivery date	Product type	Post code
Order line 1	400	300	27/09/2013	29/09/2013	Dry	2112
Order line 2	300	350	26/09/2013	29/09/2013	Fresh	2100
Order line 3	200	160	28/09/2013	30/09/2013	Dry	2103

	Weight (kg)	Volume (m^3)	Delivery date	Post code range	Initial cost (USD)	Extra cost (post cost -> USD)	Transportation requirement
	777	700	28/09/2013	2100, 2103	100	{2100 -> 21, 2103 -> 31}	Dry
	450	1000	29/09/2013	2100, 2103, 2112	120	{2100 -> 5, 2103 -> 10, 2112 -> 15}	Fresh
	600	460	30/09/2013	2100, 2112	130	{2100 -> 1, 2103 -> 2, 2112 -> 4}	Dry

$$\min \sum_{j=1}^{n} use\_truck_j$$

$$\min \sum_{j=1}^{n} truck\_transportation\_cost_j$$

A Transportation Planning Example

$$1 = \sum_{j=1}^{n} order\_in\_truck_{ij}$$

$$use\_truck_j = \bigvee_{i=1}^{m} order\_in\_truck_{ij}$$

$$truck\_weight_j \geq \sum_{i=1}^{m} order\_in\_truck_{ij} * order\_weight_i$$

$$truck\_volume_j \geq \sum_{i=1}^{m} order\_in\_truck_{ij} * order\_volume_i$$

$$truck\_transportation\_cost_j = \sum_{i=1}^{m} order\_in\_truck_{ij} * extra\_cost_j$$

$$\min \sum_{j=1}^{n} use\_truck_j \qquad \min \sum_{j=1}^{n} truck\_transportation\_cost_j$$

νZ functionality

Three main SMT-LIB2 extensions:

```
(maximize (+ x (* 2 y)))

(minimize (- x z))

(assert-soft (> x y) :weight 4)
```

νZ by example

```
(declare-const x Int)
(declare-const y Int)
(assert (or (> x (+ y 2)) (> y (+ x 3))))
(assert (=> (> x y) (> 10 x)))
(assert (=> (> y x) (> 10 y)))
(check-sat)
(get-model)
```

```
(declare-const x Int)
(declare-const y Int)
(assert (or (> x (+ y 2)) (> y (+ x 3))))
(assert (=> (> x y) (> 10 x)))
(assert (=> (> y x) (> 10 y)))
(maximize (+ x (* 2 y)))
(check-sat)
(get-model)
```

```
sat
(model
  (define-fun y () Int
    0)
  (define-fun x () Int
    (- 4))
)
```

```
(+ x (* 2 y)) |-> 23
sat
(model
  (define-fun y () Int
    9)
  (define-fun x () Int
    5)
)
```

Python too

```
(declare-const x Int)
(declare-const y Int)
(assert (or (> x (+ y 2)) (> y (+ x 3))))
(assert (=> (> x y) (> 10 x)))
(assert (=> (> y x) (> 10 y)))
(check-sat)
(get-model)
```

```
from z3 import *
x, y = Ints('x y')
opt = Optimize()
opt.add(Or(x > y + 2, y > x + 3))
opt.add(Implies(x > y, 10 > x))
opt.add(Implies(y > x, 10 > y))
h = opt.maximize(x + 2 * y)
opt.check()
print opt.model()
print h.value()
```

```
sat
(model
  (define-fun y () Int
    0)
  (define-fun x () Int
    (- 4))
)
```

```
[y = 9, x = 5]
23
```

Mixing Theories and Objective Functions

Taxonomy	Objective functions	Theories
MaxSAT	Cardinality	Any
WeightedMaxSAT	Pseudo-Boolean	Any
Difference Logic Optimization (Network Simplex)	Linear Arithmetic	Difference Logic
Linear Arithmetic Optimization	Linear Arithmetic	Linear Arithmetic

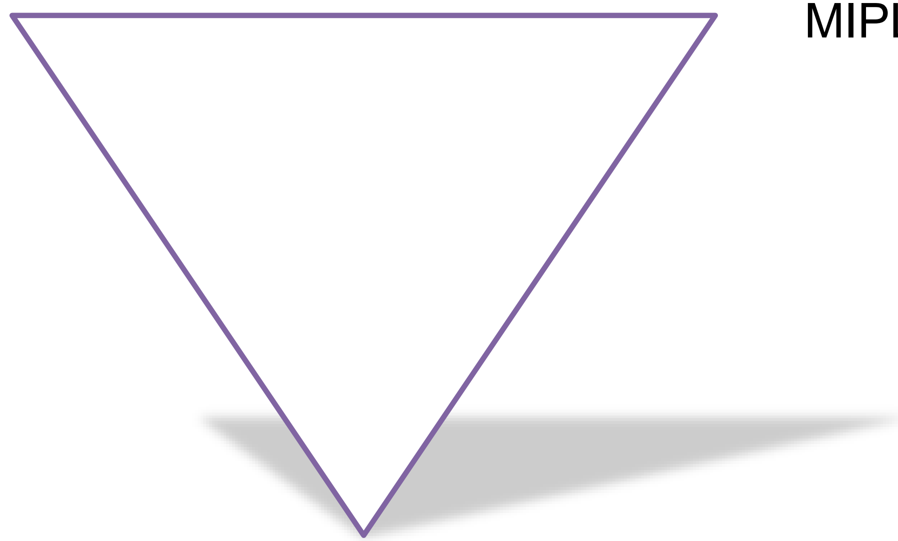
Possible future scenarios:

Quadratic Optimization	Non-linear Arithmetic	Linear Arithmetic
Non-linear Optimization	Non-linear Arithmetic	Non-linear Arithmetic

Optimizing Solvers

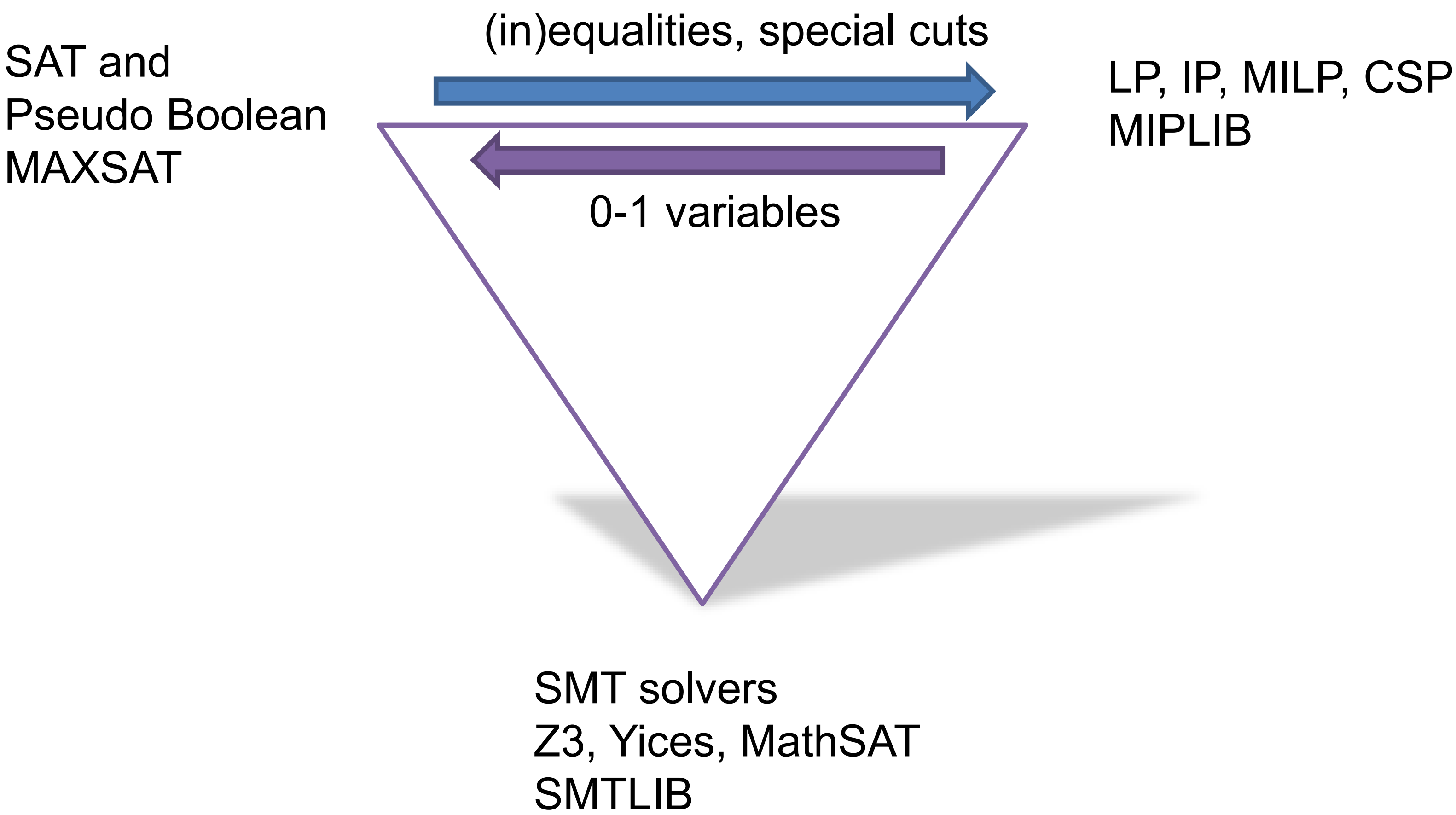
SAT and
Pseudo Boolean
MAXSAT

LP, IP, MILP, CSP
MIPLIB

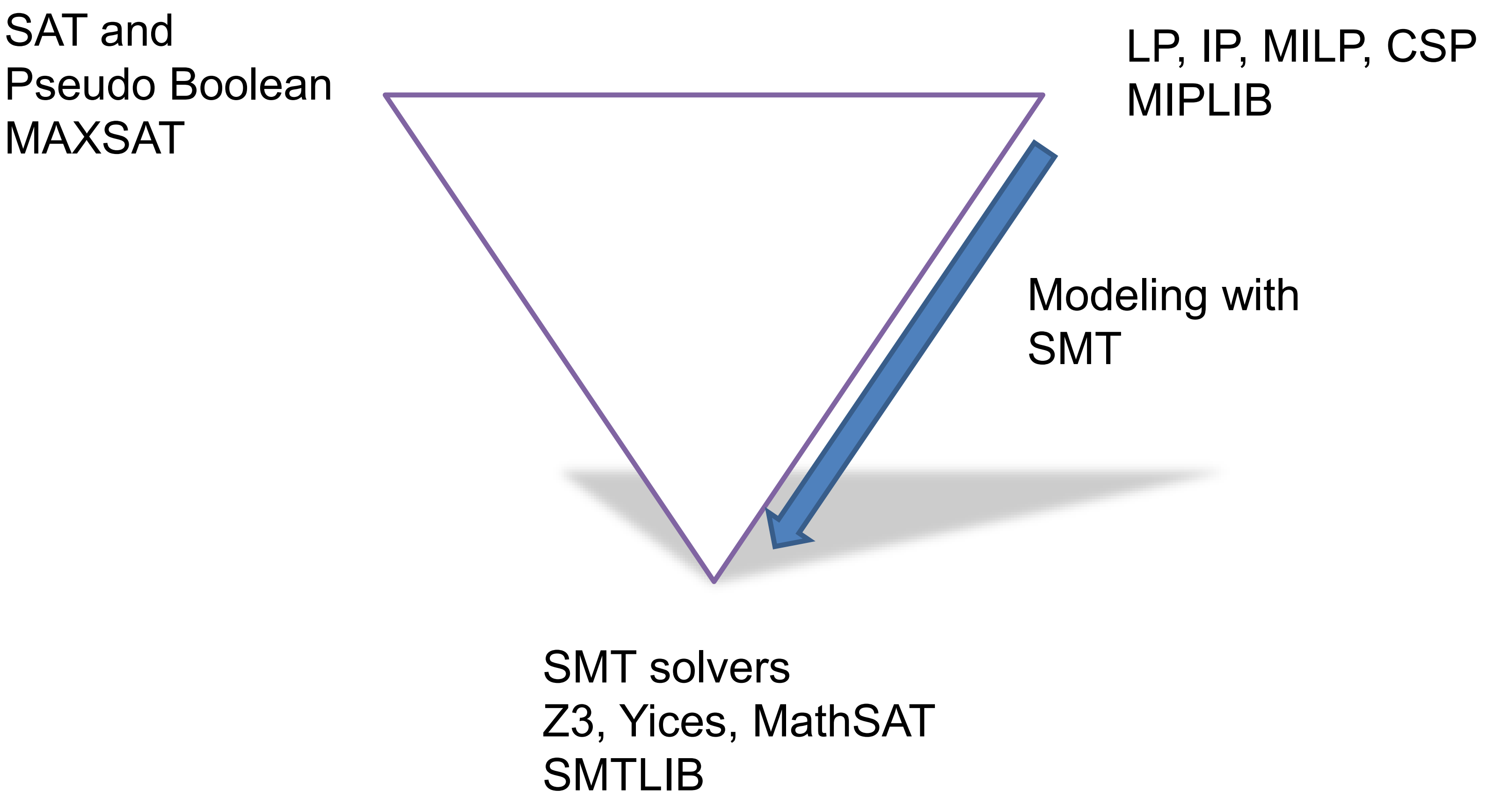


SMT solvers
Z3, Yices, MathSAT
SMTLIB

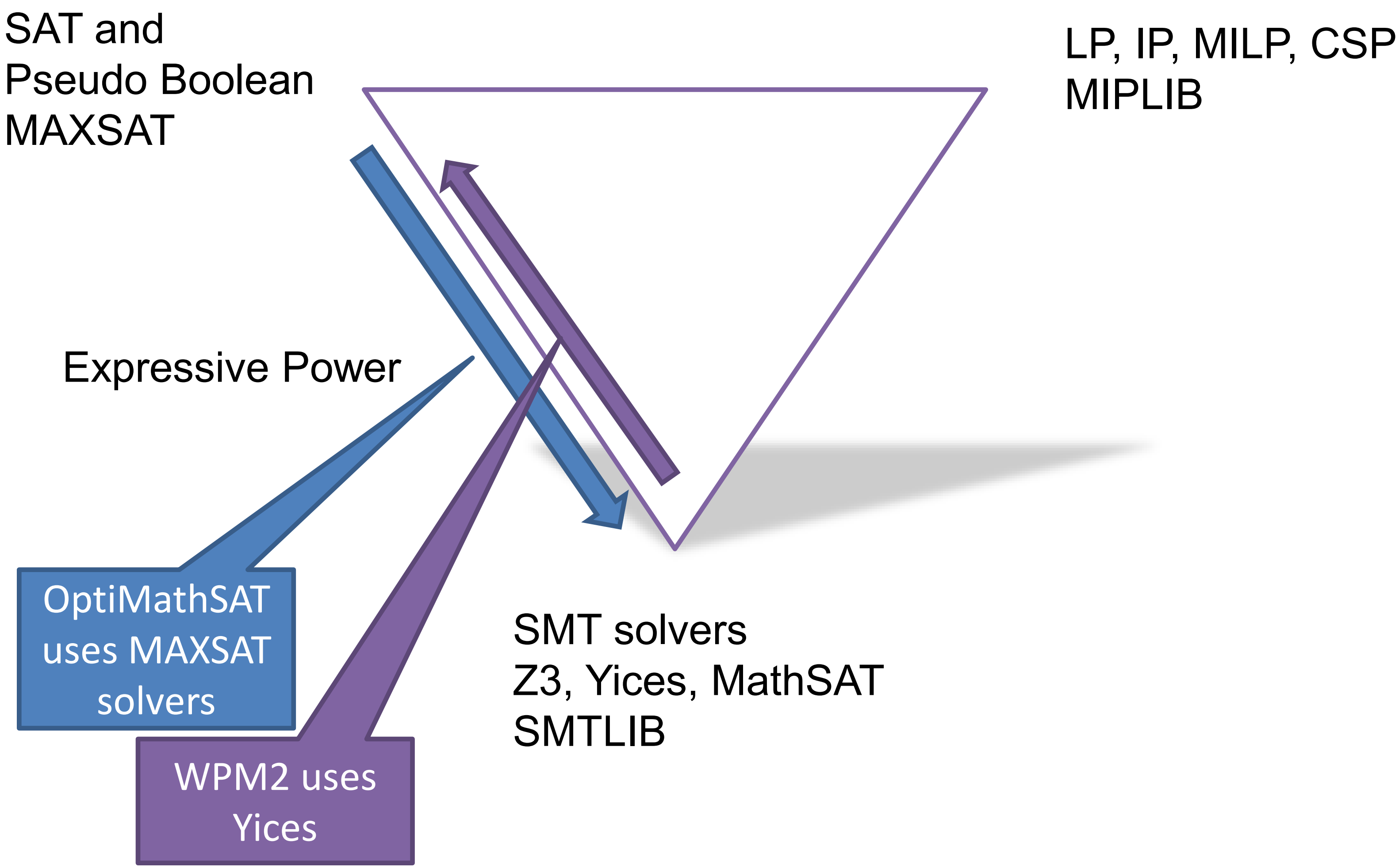
Optimizing Solvers



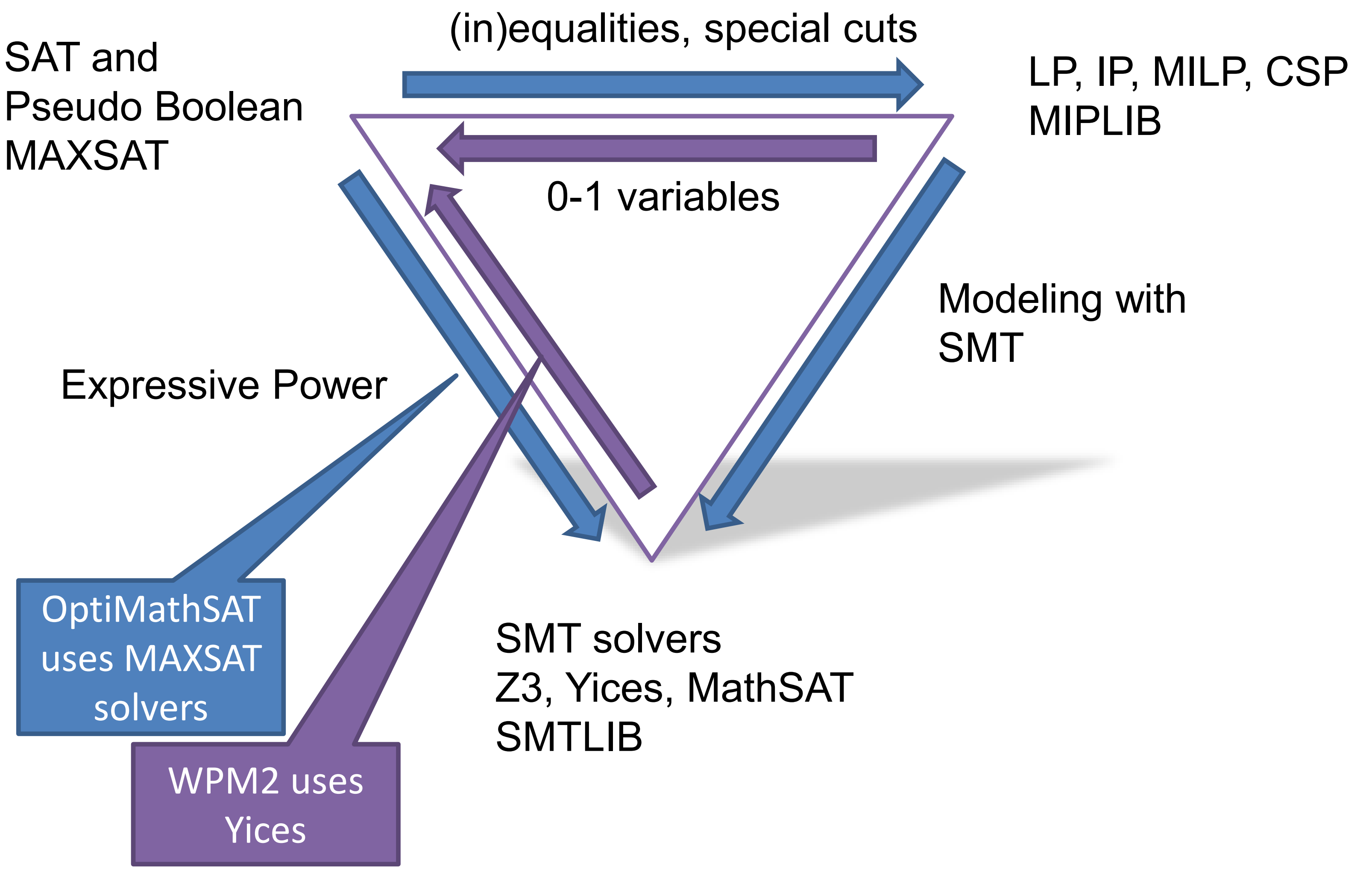
Optimizing Solvers



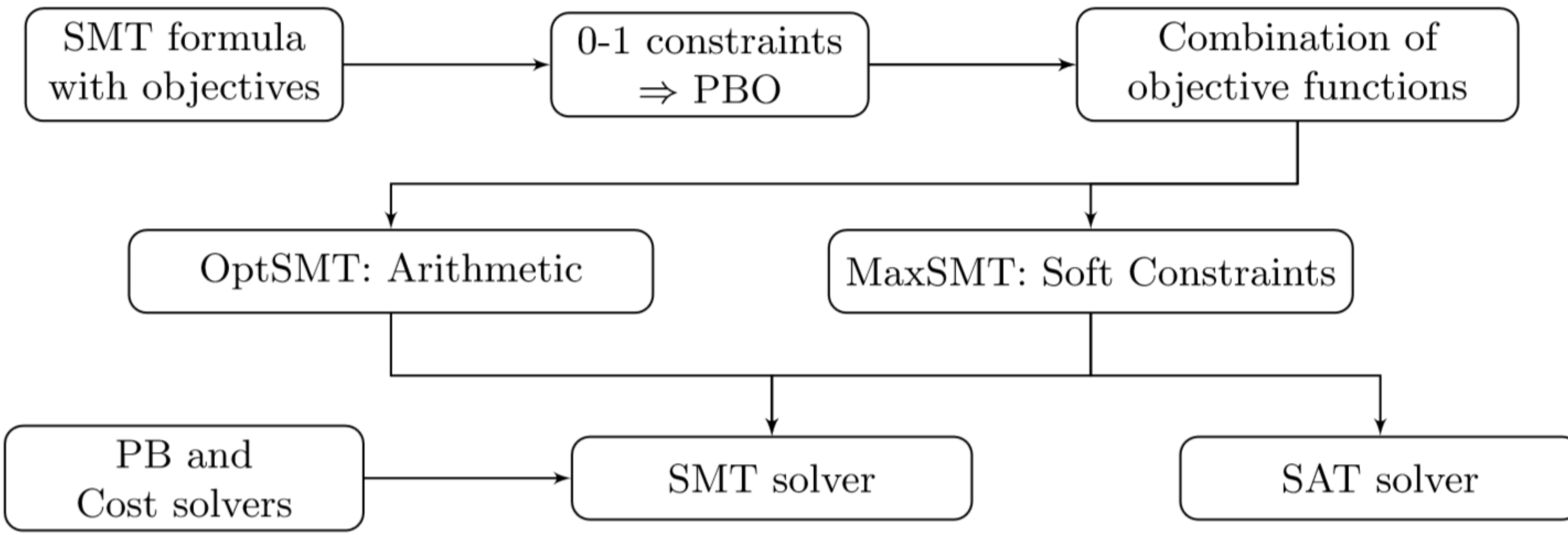
Optimizing Solvers



Optimizing Solvers



νZ system architecture



νZ for Arithmetic

Main approaches:

- Tune satisfying solutions using Primal Simplex
- Discover unbounded variables using non-standard arithmetic $\infty \leq x$
- *Maximization* Directed Clause Learning to enable expressive objective functions for simple solvers
- Bender inspired bounds strengthening using Farkas coefficients

νZ Basic Arithmetic

$\boldsymbol{Input}$: $Objective\ t\ to\ maximize$
$\boldsymbol{Input}$: $Formula\ \phi$
$\boldsymbol{Output}$: $Maximal\ value\ v, such\ that\ \phi \wedge t = \ v\ is\ satisfiable$
$v \leftarrow -\infty$
$\boldsymbol{while}\ \phi\ is\ satisfiable\ \boldsymbol{do}$
$\quad \boldsymbol{let}\ M\ be\ an\ evaluation\ that\ satisfies\ \phi\ and\ maximizes\ t$
$\quad v\ \leftarrow M(t)$
$\quad \phi\ \leftarrow\ \phi \wedge t\ >\ v$
$\boldsymbol{end}$
$\boldsymbol{return}\ v$

Use primal Simplex to maximize t

[Explored in OptiMathSAT]

νZ Non-standard Arithmetic

$\boldsymbol{Input}$: $Objective\ t\ to\ maximize$
$\boldsymbol{Input}$: $Formula\ \phi$
$\boldsymbol{Output}$: $Maximal\ value\ v, such\ that\ \phi \wedge t = v\ is\ satisfiable$
$v \leftarrow -\infty$
$\boldsymbol{if}\ \phi \wedge t \geq \infty\ is\ satisfiable\ \boldsymbol{then\ return}\ \infty$
$\boldsymbol{while}\ \phi\ is\ satisfiable\ \boldsymbol{do}$
  $\boldsymbol{let}\ M\ be\ an\ evaluation\ that\ satisfies\ \phi\ and\ maximizes\ t$
  $v \leftarrow M(t)$
  $\phi \leftarrow \phi \wedge t > v$
$\boldsymbol{end}$
$\boldsymbol{return}\ v$

Requires special version of Simplex solver using non-standard numbers: $a\epsilon + b + c\infty$

Alternative to Symba ray solving algorithm [Li, Albarghouthi, Kincaid, Gurfinkel, Chechik, POPL 2014]

νZ for Weak Arithmetic

$\boldsymbol{Input}$: $Objective\ t\ to\ maximize$
$\boldsymbol{Input}$: $Formula\ \phi$
$\boldsymbol{Output}$: $Maximal\ value\ v, such\ that\ \phi \wedge t = v\ is\ satisfiable$
$v \leftarrow -\infty$
$\boldsymbol{while}\ \phi\ is\ satisfiable\ \boldsymbol{do}$
$\quad \boldsymbol{let}\ L\ be\ consistent\ literals\ that\ imply\ \phi$
$\quad v \leftarrow \max(v, \max\{\ t \mid L\ \})$
$\quad L' \leftarrow subset\ of\ L, such\ that\ L' \rightarrow t \leq \max\{t \mid L\ \}$
$\quad \phi \leftarrow \phi \wedge \neg L'$
$\boldsymbol{end}$
$\boldsymbol{return}\ v$

Used when ϕ is difference logic and t is beyond difference logic

Use model for ϕ as starting point for primal Simplex that solves $\max\{\ t \mid L\ \}$

Learn from failure, a lá Bender

Input: Objective t to maximize
Input: Formula F
Output: Maximal value v, such that $v = t \wedge F$ is satisfiable
$\mathsf{lo} \leftarrow -\infty, \mathsf{hi} \leftarrow \infty$
while $\mathsf{lo} < \mathsf{hi}$ **do**
    Pick mi such that $\mathsf{lo} < \mathsf{mi} < \mathsf{hi}$
    if $\mathsf{mi} < t \wedge F$ *is satisfiable* **then**
        Let M be an evaluation that satisfies F and maximizes t
        $\mathsf{lo} \leftarrow M(t)$
    end
    else
        Let $(A_i x \leq b_i \rightarrow t \leq \mathsf{mi})$ be T-lemmas for $i \in \mathcal{I}$
        That is $F \rightarrow \bigvee_i A_i x \leq b_i$
        Let r_i be Farkas coefficients for the T-lemmas, such that $r_i A_i > r_i b_i$, $r_i A_i = t$
        $\mathsf{hi} \leftarrow \max\{r_i b_i \mid i \in \mathcal{I}\}$
    end
end
return hi

Preliminary experience: unsat calls (else branch) are too expensive νZ and MaxSAT

νZ and MaxSAT/SMT

Maximize $\varphi_1 w_1 + \cdots + \varphi_n w_n$

MaxSAT

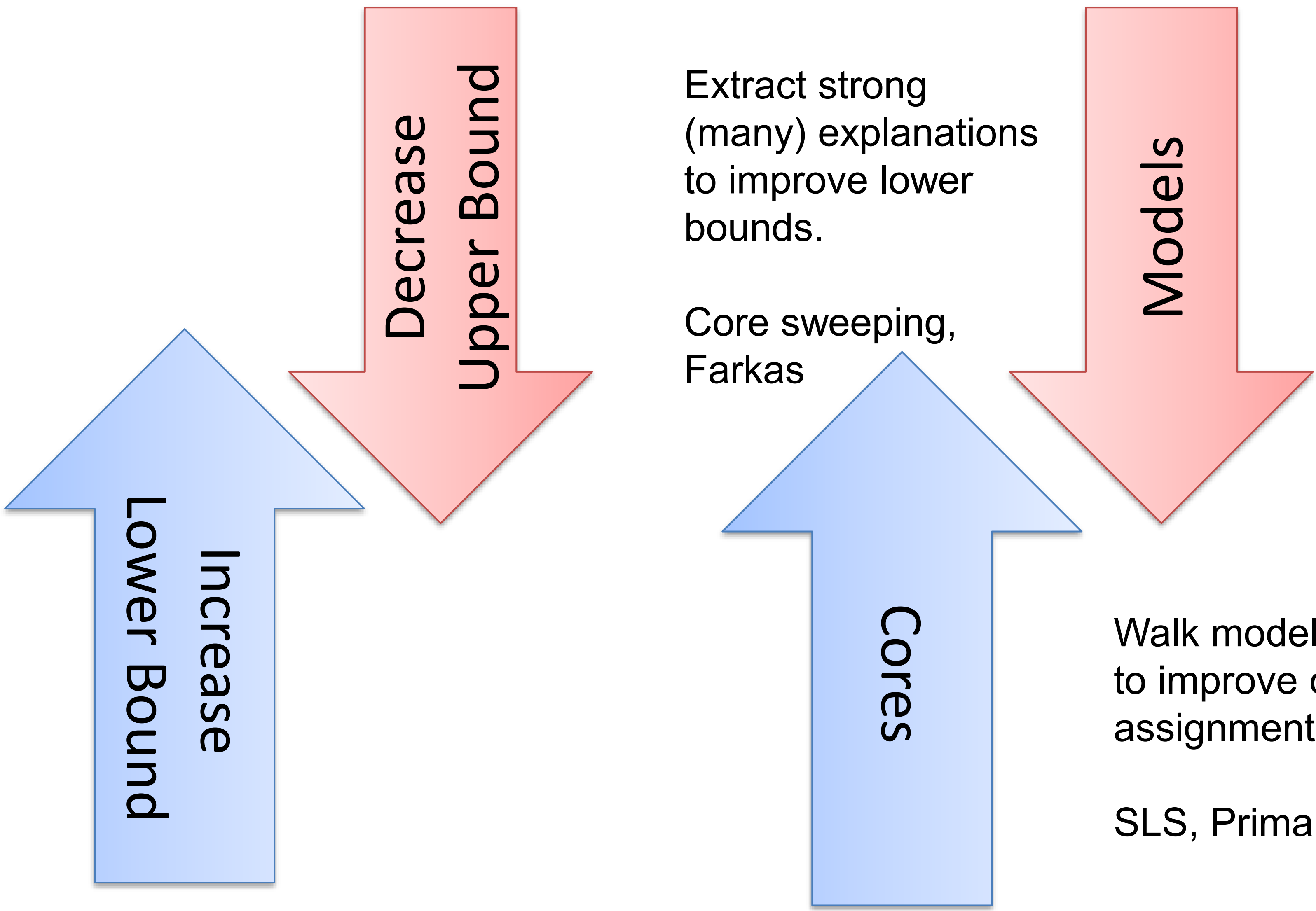
$$\varphi_1 \; with\; penalty\; w_1$$
$$\cdots$$
$$\varphi_n \; with\; penalty\; w_n$$

νZ for MaxSMT

Main approaches:

- Portfolio of Algorithms for (weighted) MaxSAT:
  - Reduction to Pseudo Boolean [Chai, Kuehlmann ’03]
  - Solver-based [Nieuwenhuis, Oliveras ‘06]
  - Core-based [Fu, Malik ‘06]
  - Core and model guided [Heras et al. ‘10]
  - Core guided WPM2 [Ansótegui et al. ‘13]
  - Model and core guided BCD2 [Heras et al. ‘13]
  - Core, hitting sets, model [Davies et al. ‘13]
  - MaxRes [Narodytska et al. ‘14]
- Custom CDCL(PB) solver
- Stochastic Local Search to aid model-based approaches [Wintersteiger, Frolich]

Portfolio of strategies



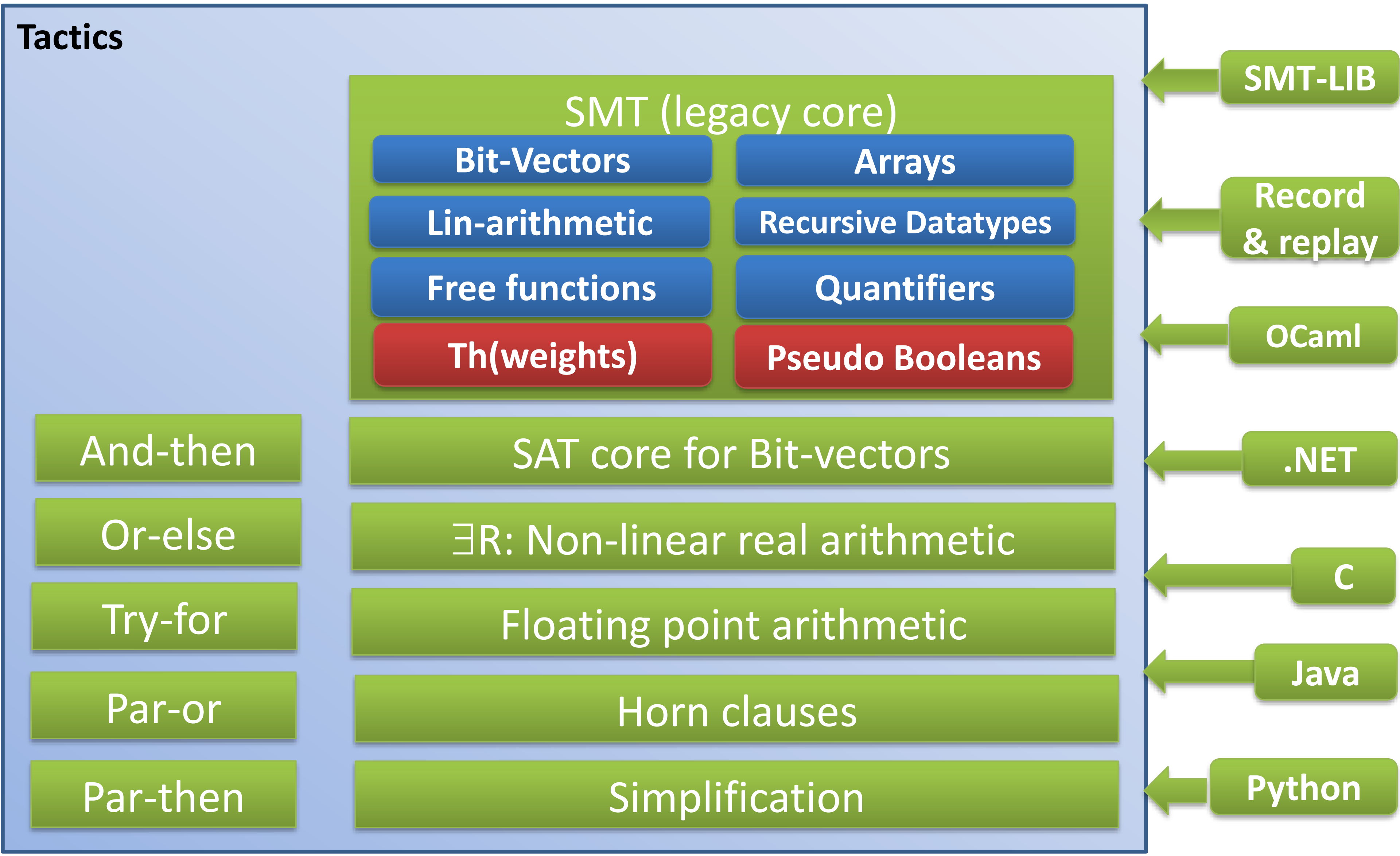
Extract strong (many) explanations to improve lower bounds.

Core sweeping, Farkas

Walk model space to improve current assignment.

SLS, Primal Simplex

CDCL(Weights & PB)



CDCL(Weights)

$Name$	$Formula$	$weight$
F_0	$a \vee b \vee x \geq 2$	∞
F_1	$\neg a \vee x \geq 3$	3
F_2	$\neg b \vee x \geq 3$	4
F_3	$x < 2$	5

Unsat

CDCL(Weights)

$Name$	$Formula$	$weight$
F_0	$a \vee b \vee x \geq 2$	∞
F_1	$\neg a \vee x \geq 3$	3
F_2	$\neg b \vee x \geq 3$	4
F_3	$x < 2$	5

Penalty: ∞

Sat $\neg \boldsymbol{a} \wedge \neg \boldsymbol{b} \wedge \boldsymbol{x} < \mathbf{2}$

CDCL(Weights)

$Name$	$Formula$	$weight$
F_0	$a \vee b \vee x \geq 2$	∞
F_1	$\neg a \vee x \geq 3$	3
F_2	$\neg b \vee x \geq 3$	4
F_3	$x < 2$	5

Sat $\boldsymbol{\neg a \wedge b \wedge x = 2}$

Penalty: 9 = 4 + 5

CDCL(Weights)

$Name$	$Formula$	$weight$
F_0	$a \vee b \vee x \geq 2$	∞
F_1	$\neg a \vee x \geq 3$	3
F_2	$\neg b \vee x \geq 3$	4
F_3	$x < 2$	5

Sat $\neg a \wedge \neg b \wedge x \geq 2$

Penalty: 5

CDCL(Weights)

$Name$	$Formula$	$weight$
F_0	$a \vee b \vee x \geq 2$	∞
F_1	$\neg a \vee x \geq 3$	3
F_2	$\neg b \vee x \geq 3$	4
F_3	$x < 2$	5

Penalty: 3

Sat $a \wedge \neg b \wedge x < 2$

CDCL(Weights)

$Formula$	$weight$
$a \vee b \vee x \geq 2$	∞
$F_1 \vee \neg a \vee x \geq 3$	3
$F_2 \vee \neg b \vee x \geq 3$	4
$F_3 \vee x < 2$	5

```
let block() =
    let offender = optimize_cost ctx costs min_cost
    th.AssertTheoryAxiom(ctx.MkNot(ctx.MkAnd offender))
let Assign p vl =
    if vl then
       let w = weights.[p]
       cost <- cost + w;          trail Add (fun () -> cost <- cost - w)
       costs <- (w,p)::costs;     trail.Add (fun () -> costs <- List.tail costs)
       if cost > min_cost then
          block()
let _ = th.NewAssignment <- (fun p vl -> Assign p vl)
```

Initially: All atoms are unassigned

$Cost \; = \; 0$

Assert $\boldsymbol{\neg a \wedge b \wedge x < 2}$

Propagate $\boldsymbol{F_2}$: $Cost \; := Cost + 4 \; := 4$

Best so far: $MinCost \; = \; 4$

Add Axiom $\boldsymbol{\neg F_2}$ *- backtrack*

Assert $\boldsymbol{F_3}$ $Cost$ = 5 > $MinCost$

Add Axiom $\boldsymbol{\neg F_3}$ *- backtrack*

…. **Assert** $\boldsymbol{a \wedge \neg b \wedge x < 2 \wedge F_1}$

[Oliveras, Nieuenhuis, SAT 06]

Core Engine in Z3: Modern DPLL/CDCL

Initialize	$\epsilon \mid F$	F *is a set of clauses*
Decide	$M \mid F \implies M, \ell \mid F$	ℓ *is unassigned*
Propagate	$M \mid F, C \vee \ell \implies M, \ell^{C\vee\ell} \mid F, C \vee \ell$	C *is false under* M
Sat	$M \mid F \implies M$	F *true under* M
Conflict	$M \mid F, C \implies M \mid F, C \mid C$	C *is false under* M
Learn	$M \mid F \mid C \implies M \mid F, C \mid C$	
Unsat	$M \mid F \mid \emptyset \implies Unsat$	
Backjump	$MM' \mid F \mid C \vee \ell \implies M\ell^{C\vee\ell} \mid F$	$\bar{C} \subseteq M, \neg\ell \in M'$
Resolve	$M \mid F \mid C' \vee \neg\ell \implies M \mid F \mid C' \vee C$	$\ell^{C\vee\ell} \in M$
Forget	$M \mid F, C \implies M \mid F$	C is a learned clause
Restart	$M \mid F \implies \epsilon \mid F$	

Model

Proof

Conflict Resolution

[Nieuwenhuis, Oliveras, Tinelli J.ACM 06] customized

CDCL($\boldsymbol{T}$) solver interaction

T- Propagate $\quad M \mid F, C \vee \ell \implies M, \ell^{C \vee \ell} \mid F, C \vee \ell \quad C \text{ } is \text{ } false \text{ } under \text{ } T + M$

T- Conflict $\quad M \mid F \implies M \mid F \mid \neg M' \quad M' \subseteq M \text{ } and \text{ } M' is \text{ } false \text{ } under \text{ } T$

T- Propagate $\quad a > b, b > c \mid F, a \leq c \vee b \leq d \implies$

$$a > b, b > c, b \leq d^{a \leq c \vee b \leq d} \mid F, a \leq c \vee b \leq d$$

T- Conflict $\quad M \mid F \implies M \mid F, a \leq b \vee b \leq c \vee c < a$

$$where \text{ } a > b, b > c, a \leq c \subseteq M$$

Fu & Malik 2006

A: $\underbrace{F_0, F_1, F_2, F_3,}_{core} F_4$

A': $F_0 \wedge (B_1 + B_2 + B_3 \leq 1),$
$B_1 \vee F_1, B_2 \vee F_2, B_3 \vee F_3, F_4$

B's are fresh

cost(M, F_0, F_1, F_2, F_3, F_4) $\stackrel{\text{def}}{=}$
value of $4 - F_1 + F_2 + F_3 + F_4$ under M

Lemma: for any M of **A**, there is M' of **A'**:
cost(M, **A**) = 1 + cost(M', **A'**)

[Narodytska & Bacchus. '14, AAAI] (implicit MaxRes version)

MaxRes

A: $\underbrace{F, F_1, F_2, F_3, F_4,}_{core}\ F_5$

A': $F,\ F_2 \vee F_1,\ \ F_3 \vee (F_1 \wedge F_2),\ \ F_4 \vee \big((F_1 \wedge F_2) \wedge F_3\big),\ F_5$

Lemma: for any model M, cost(M, **A**) = 1 + cost(M, **A'**)

Proof: $M\left(F_j\right) = \perp, j\ is\ least \Rightarrow$

$$M(F_i') = M(F_{i+1}) \vee (i = j \neq 1), \qquad \forall i.$$

Cores and Correction Sets

$$\underbrace{F_1, F_2, F_3}_{core}, F_4$$

$$\underbrace{F_4, F_2, F_3}_{core}, F_1$$

$$\underbrace{F_1, F_4, F_3}_{core}, F_2$$

$$F_1, F_2, \underbrace{F_3}_{correction\ set}, F_4$$

$$\underbrace{F_1, F_2}_{correction\ set}, F_3, F_4$$

$$\underbrace{F_1, F_4}_{correction\ set}, F_3, F_2$$

$$\underbrace{F_2, F_4}_{correction\ set}, F_3, F_1$$

Dual MaxRes

A: $\underbrace{F, F_1, F_2, F_3, F_4,}_{correction\ set}\ F_5$

A': $F \wedge (F_1 \vee F_2 \vee F_3 \vee F_4)$,

$F_2 \wedge F_1, \quad F_3 \wedge (F_1 \vee F_2), \quad F_4 \wedge \big((F_1 \vee F_2) \vee F_3\big), \quad F_5$

Lemma: for any M of **A'**, cost(M, **A**) = cost(M, **A'**)

Proof: $M\left(F_j\right) = \top, j\ is\ least \Rightarrow$

$$M(F_i{'}) = M(F_{i+1}) \wedge (i \neq j \vee j = 1), \qquad \forall i.$$

CDCL(PB)

- PB Constraints: $\sum_i a_i \ell_i \geq b, \;\; \sum_i a_i \ell_i = b$

  Cardinality is a special case: $\sum_i \ell_i \geq c$

- PB solver is a satellite theory to Z3's core CDCL engine.

- Integration with other theories "for free".

CDCL(PB)

- Propagation
- Conflict Resolution [Chai et.al. ‘03]
- Sorting Circuits [Abío et.al. ‘13]
- Simplex over 0-1 constraints
  - Extract clauses from LP infeasible certificates

Observations

- Resolution poor at equalities (well known)
  - Simplex helps

- PB Conflict Resolution is pretty expensive

- Sorting Circuits help (of course), but
  - Simplex helps elsewhere

Combining Objectives

Box$(\boldsymbol{x}, \boldsymbol{y})$**:**

$$v_x := \max\{x \mid \varphi(x,y)\}$$
$$v_y := \max\{y \mid \varphi(x,y)\}$$

Lex$(\boldsymbol{x}, \boldsymbol{y})$**:**

$$v_x := \max\{x \mid \varphi(x,y)\}$$
$$v_y := \max\{y \mid \varphi(v_x,y)\}$$

Pareto$(\boldsymbol{x}, \boldsymbol{y})$**:**

$$v_x, v_y := \varphi\left(v_x, v_y\right),$$
$$\forall x, y.\, \varphi(x,y) \rightarrow \neg\left((x,y) > \left(v_x, v_y\right)\right)$$

νZ Basic Box Optimization

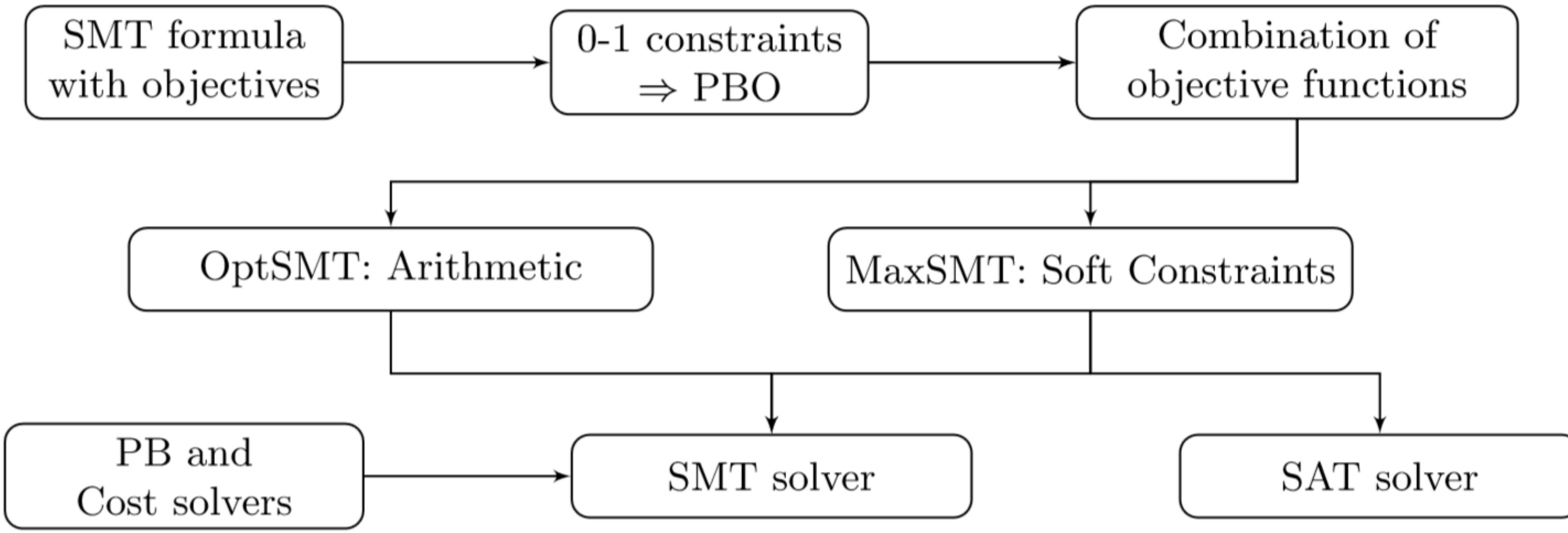
$\boldsymbol{Input}$: $Objective\ t_1, t_2\ to\ maximize$
$\boldsymbol{Input}$: $Formula\ \phi$
$\boldsymbol{Output}$: $Min\ values\ v_1, v_2\ , such\ that\ \phi \rightarrow t_1 \leq v_1 \wedge t_2 \leq v_2\ is\ valid$
$v_1, v_2 \leftarrow -\infty$
$\boldsymbol{while}\ \phi\ is\ satisfiable\ \boldsymbol{do}$
$\quad let\ M_i\ be\ evaluations\ that\ satisfy\ \phi\ and\ maximize\ t_i$
$\quad v_1\ \leftarrow M_1(t_1), v_2 \leftarrow M_2(t_2),$
$\quad \phi\ \leftarrow\ \phi \wedge (t_1\ >\ v_1 \vee t_2 > v_2)$
$\boldsymbol{end}$
$\boldsymbol{return}\ v_1, v_2$

νZ Pareto Optimization

$\boldsymbol{Input}$: $Objective\ t_1, t_2\ to\ maximize$
$\boldsymbol{Input}$: $Formula\ \phi$
$\boldsymbol{Output}$: $Pareto \max front$
$\boldsymbol{while}\ \phi\ is\ satisfiable\ \boldsymbol{do}$
$\quad \psi \leftarrow \phi$
$\quad \boldsymbol{while}\ \psi\ is\ satisfiable\ \boldsymbol{do}$
$\quad\quad \boldsymbol{let}\ M\ be\ an\ evaluation\ that\ satisfies\ \psi$
$\quad\quad v_1 \leftarrow M(t_1), v_2 \leftarrow M(t_2),$
$\quad\quad \psi \leftarrow \psi \wedge t_1 \geq v_1 \wedge t_2 \geq v_2 \wedge (t_1 > v_1 \vee t_2 > v_2)$
$\quad \phi \leftarrow \phi \wedge (t_1 > v_1 \vee t_2 > v_2)$
$\quad \boldsymbol{output}\ v_1, v_2$
$\boldsymbol{end}$

Guided Improvement Algorithm [Rayside, Estler, Jackson, MIT-TR 2009]
Our incarnation is simple. Current work on GIA includes parallelization, incrementality, check-pointing [at U. Waterloo]. For linear arithmetic, a possible enhancement is to use *primal simplex* to improve *M.*

νZ - Summary



νZ integrates and extends many new algorithms from MaxSAT, linear optimization, combination methods.

νZ is still very actively being improved based on benchmarks and case studies obtained from Z3 users.