Mining Definitions in KissAT with KITTENS

Mathias Fleury & Armin Biere

Pragmatics of SAT (POS'21)

affiliated to SAT'21

FWF
Der Wissenschaftsfonds.

Cyberspace
July 5, 2021

JKU
JOHANNES KEPLER
UNIVERSITÄT LINZ
LIT AI Laboratory

Tseitin encoding of AND-gate $x = a \wedge b$

$$F \equiv \underbrace{(\bar{a} \vee \bar{b} \vee x)}_{G_x} \wedge \underbrace{(a \vee \bar{x}) \wedge (b \vee \bar{x})}_{G_{\bar{x}}} \wedge \overbrace{(c \vee x) \wedge (d \vee x)}^{H_x} \wedge \overbrace{(e \vee \bar{x}) \wedge (f \vee \bar{x})}^{H_{\bar{x}}} \wedge (\bar{c} \vee \bar{d} \vee \bar{e} \vee \bar{f})$$

F': remaining clauses

non-gate clauses with "x"

Resolving all clauses with x or $\bar{x}$ results in the following CNF.

$$\begin{aligned} F'' \equiv \\ \cancel{(\bar{a} \vee \bar{b} \vee a) \wedge (\bar{a} \vee \bar{b} \vee b)} \wedge \\ (\bar{a} \vee \bar{b} \vee e) \wedge (\bar{a} \vee \bar{b} \vee f) \wedge \\ (a \vee c) \wedge (a \vee d) \wedge (b \vee c) \wedge (b \vee d) \wedge \\ (c \vee e) \wedge (c \vee f) \wedge (d \vee e) \wedge (d \vee f) \wedge \\ (\bar{c} \vee \bar{d} \vee \bar{e} \vee \bar{f}) \end{aligned}$$

anyhow not counted

tautological	$G_x \otimes G_{\bar{x}}$	resolvents
kept	$G_x \otimes H_{\bar{x}}$	resolvents
kept	$G_{\bar{x}} \otimes H_x$	resolvents
redundant	$H_x \otimes H_{\bar{x}}$	resolvents

Partition of F

$$F \equiv \overbrace{G_x \wedge H_x}^{F_x} \wedge \overbrace{G_{\bar{x}} \wedge H_{\bar{x}}}^{F_{\bar{x}}} \wedge F'$$

clauses without "x"

G_x: gate clauses

H_x: non-gate clauses

$$F \equiv (F_x \otimes F_{\bar{x}}) \wedge F' \equiv (G_x \otimes H_{\bar{x}}) \wedge (G_{\bar{x}} \otimes H_x) \wedge F'$$

standard elimination

gate elimination

gate def. extraction

Function `FindGateClauses`$(F,\ x)$

Input: The clauses F and the variable x

Output : The pair (G, H) of gate and non-gate clauses of F to be resolved

let E_x = clauses of F with x or $\bar{x}$

if F contains Tseitin encoding of a gate with output x **then**

  let G be the clauses of the Tseitin encoding of the gate

  return $(G, E_x \backslash G)$

OLD syntactic

if call to KITTEN on $(S_x)|_{\bar{x}} \wedge (S_{\bar{x}})|_x$ returns UNSAT **then**

  determine G from clausal core (adding back x and $\bar{x}$)

  return $(G, E_x \backslash G)$

NEW

Core based

return $(E_x, \emptyset)$

Function `BoundedVariableElimination`$(F,\ x, k)$

Input: The clauses F, the variable x, bound k on additional resolvents

Output : Simplify clauses of F in place if resolvents sufficiently bounded

let (G, H) = `FindGateClauses`$(F,\ x)$

let G_ℓ = clauses of G with ℓ $\quad (\ell \in \{x, \bar{x}\})$

let H_ℓ = clauses of H with ℓ $\quad (\ell \in \{x, \bar{x}\})$

let $R_x = (G_x \otimes G_{\bar{x}}) \wedge (G_x \otimes H_{\bar{x}}) \wedge (G_{\bar{x}} \otimes H_x)$

let E_x = clauses of F with x or $\bar{x}$

if $|R_x| - |E_x| \leq k$ **then**

  replace F by $R \wedge F'$ where F' are the clauses in F without x nor $\bar{x}$

resolvents on "x"

original clauses with "x"

Core Based Gate/Definition Extraction

$$F = \underbrace{(x \vee b) \wedge (\bar{x} \vee a) \wedge (\bar{x} \vee \bar{a} \vee \bar{b})}_{F_x} \wedge (x \vee \bar{a}) \wedge (\bar{a} \vee c) \wedge (a \vee \bar{b}) \wedge (b \vee \bar{c})$$

$$(F_x|_{\bar{x}}) \wedge (F_{\bar{x}}|_x) \equiv \overbrace{b \wedge a}^{1^{st}\text{ Core (found)}} \wedge \underbrace{(\bar{a} \vee \bar{b}) \wedge \bar{a}}_{\text{potential } 2^{nd} \text{ core (ignored)}}$$

Conjunction of cofactors of clauses with "x" UNSAT?

Thus:

$$F = \underbrace{(x \vee b)}_{G_x} \wedge \underbrace{(\bar{x} \vee a) \wedge (\bar{x} \vee \bar{a} \vee \bar{b})}_{G_{\bar{x}}} \wedge \underbrace{(x \vee \bar{a})}_{H_x} \wedge \overbrace{(\bar{a} \vee c) \wedge (a \vee \bar{b}) \wedge (b \vee \bar{c})}^{F'}$$

$$F = \underbrace{(x \vee b)}_{G_x} \wedge \underbrace{(\bar{x} \vee a) \wedge (\bar{x} \vee \bar{a} \vee \bar{b})}_{G_{\bar{x}}} \wedge \underbrace{(x \vee \bar{a})}_{H_x} \wedge \overbrace{(\bar{a} \vee c) \wedge (a \vee \bar{b}) \wedge (b \vee \bar{c})}^{F'}$$

gate resolvents: $G_x \otimes G_{\bar{x}} = a \vee b$ — non-tautological

mixed resolvents:

$$G_x \otimes H_{\bar{x}} = \top$$

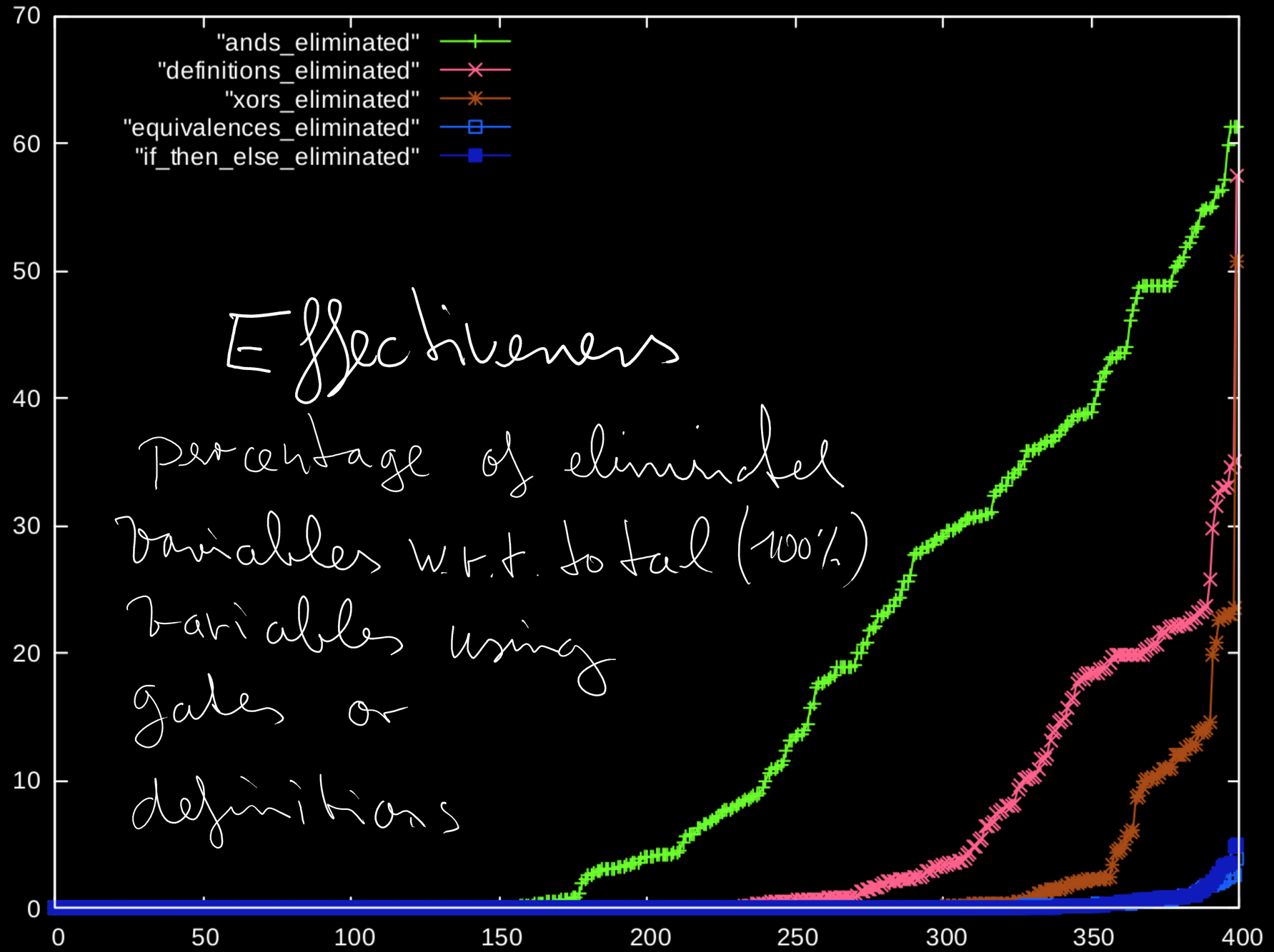
$$G_{\bar{x}} \otimes H_x = \cancel{(a \vee \bar{a}) \wedge} (\bar{a} \vee \bar{b})$$

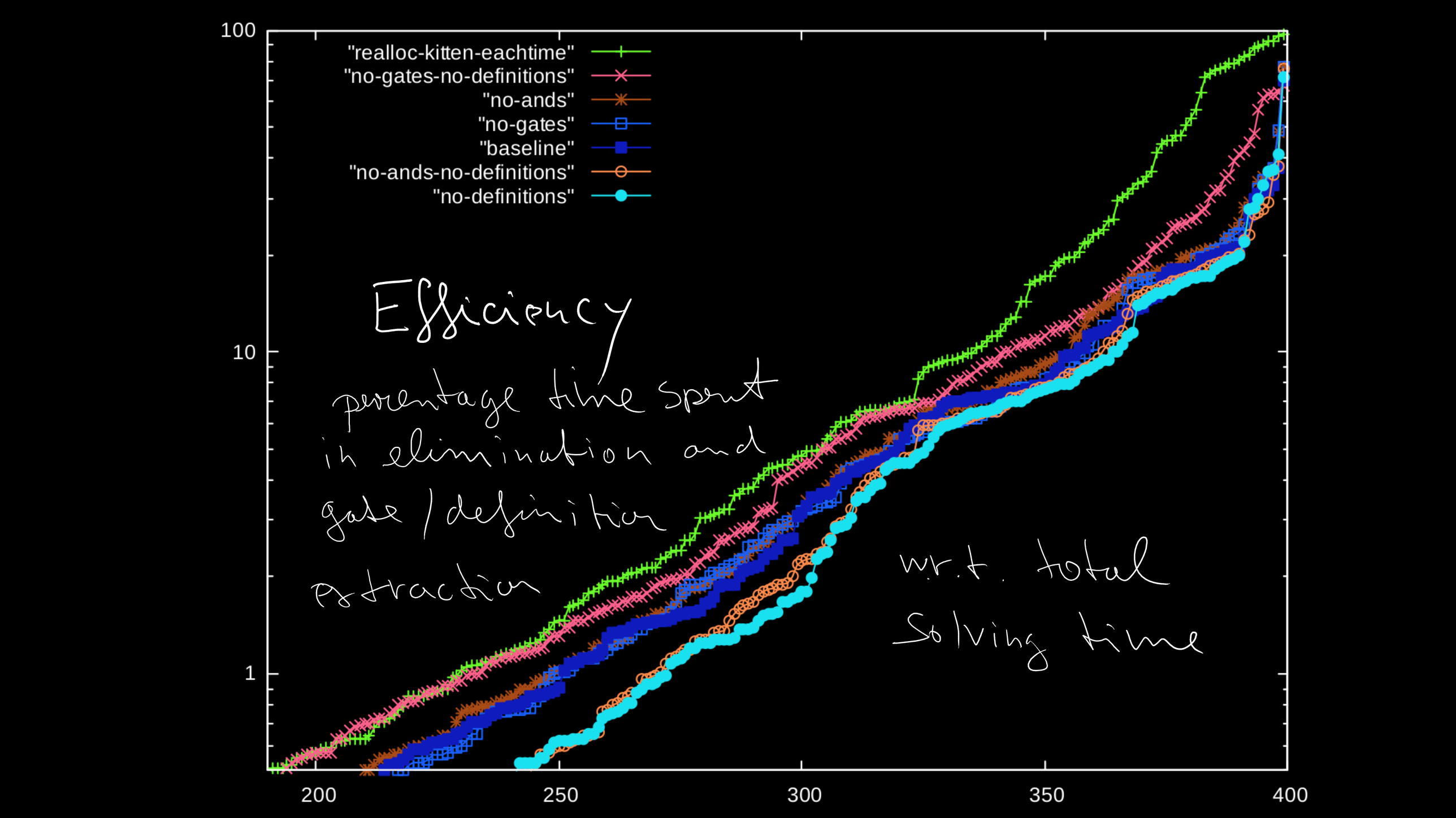
non-gate resolvents: $H_x \otimes H_{\bar{x}} = \top$ — anyhow redundant

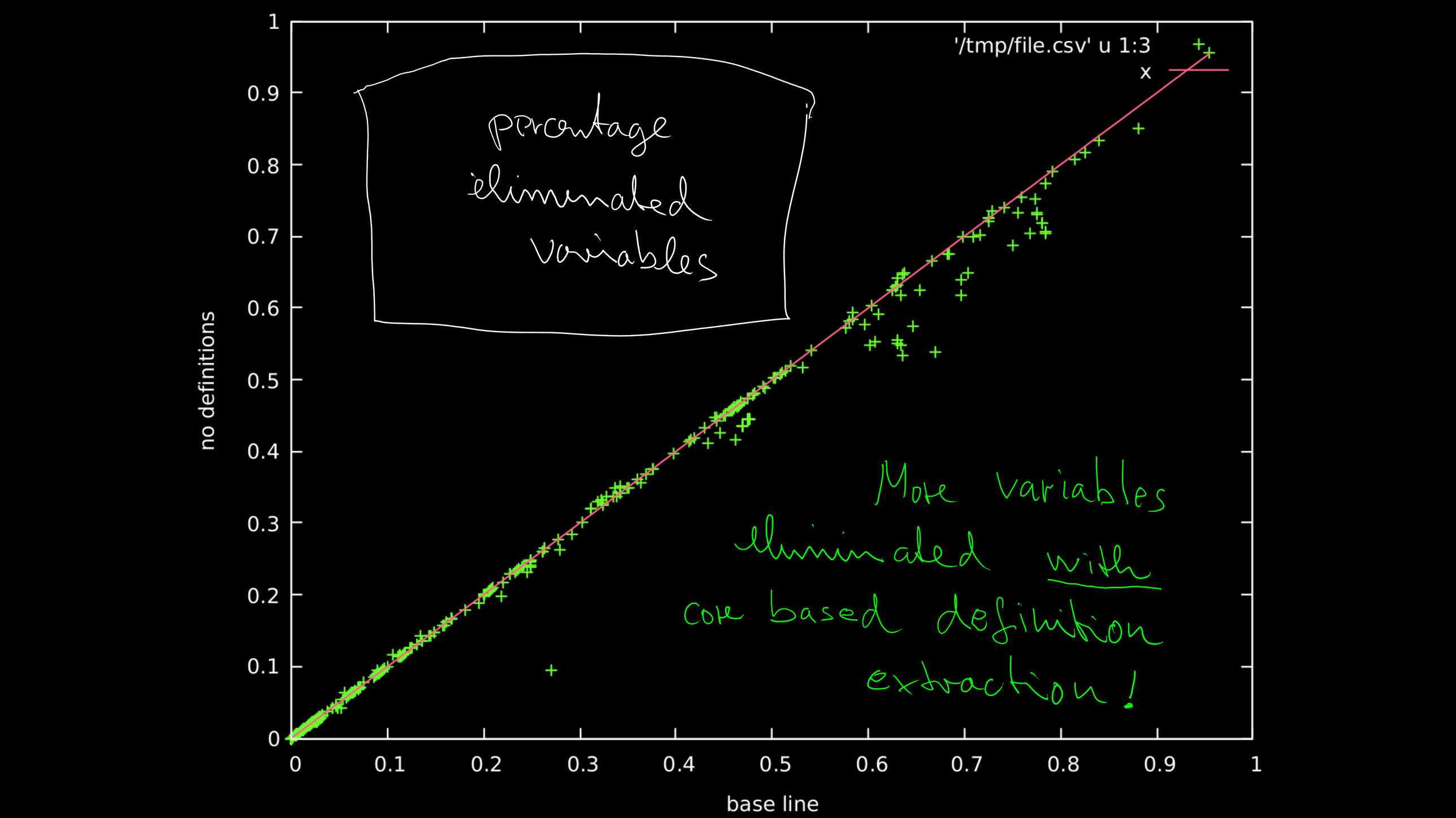
NOT redundant!

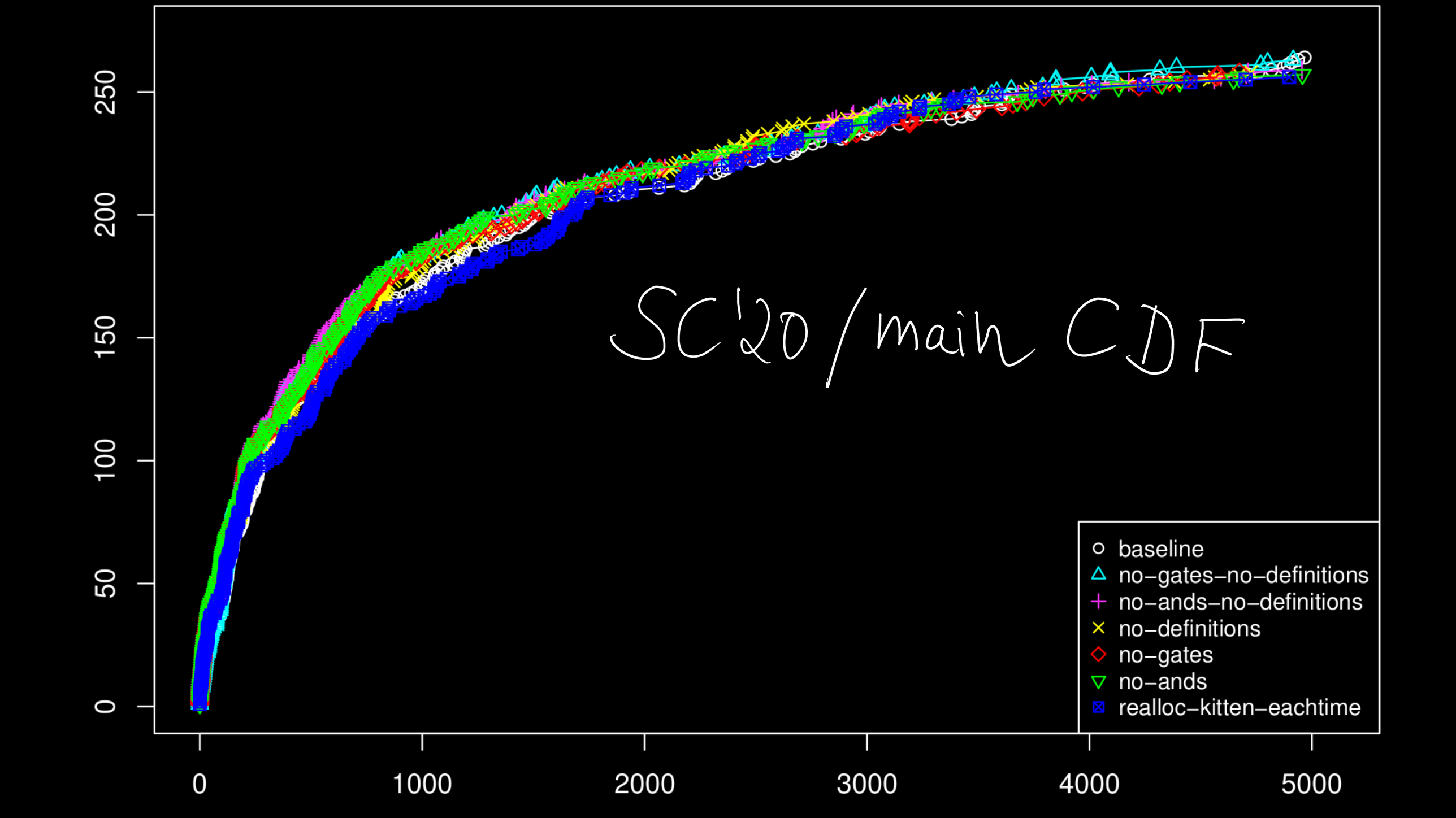
$$F'' = \underbrace{(a \vee b)}_{G_x \otimes G_{\bar{x}}} \wedge \underbrace{(\bar{a} \vee \bar{b})}_{G_{\bar{x}} \otimes H_x} \wedge \overbrace{(\bar{a} \vee c) \wedge (a \vee \bar{b}) \wedge (b \vee \bar{c})}^{F'}$$

Consider $a = b = c = 1$


"ands_eliminated"
"definitions_eliminated"
"xors_eliminated"
"equivalences_eliminated"
"if_then_else_eliminated"
Effectiveness
percentage of eliminated variables w.r.t. total (100%) variables using gates or definitions
0
10
20
30
40
50
60
70
0
50
100
150
200
250
300
350
400


"realloc-kitten-eachtime"
"no-gates-no-definitions"
"no-ands"
"no-gates"
"baseline"
"no-ands-no-definitions"
"no-definitions"
Efficiency
percentage time spent in elimination and gate/definition extraction
w.r.t. total solving time
100
10
1
200
250
300
350
400


'/tmp/file.csv' u 1:3
x
percentage eliminated variables
More variables eliminated with core based definition extraction!
no definitions
base line
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


SC'20 / main CDF
0
50
100
150
200
250
0
1000
2000
3000
4000
5000
baseline
no-gates-no-definitions
no-ands-no-definitions
no-definitions
no-gates
no-ands
realloc-kitten-eachtime

```
kitten *kitten_init (void);
void kitten_clear (kitten *);
void kitten_release (kitten *);

void kitten_track_antecedents (kitten *);
void kitten_shuffle_clauses (kitten *);
void kitten_randomize_phases (kitten *);
void kitten_flip_phases (kitten *);

void kitten_assume (kitten *, unsigned lit);

void kitten_clause (kitten *, size_t size, unsigned *);
void kitten_unit (kitten *, unsigned);
void kitten_binary (kitten *, unsigned, unsigned);

void kitten_clause_with_id_and_exception (kitten *, unsigned id,
                                          size_t size, const unsigned *,
                                          unsigned except);
```

no allocation!

in memory proofs!

randomize!

clause ids

map back core

```
void kitten_set_ticks_limit (kitten *, uint64_t);

int kitten_solve (kitten *);

signed char kitten_value (kitten *, unsigned);
bool kitten_failed (kitten *, unsigned);

unsigned kitten_compute_clausal_core (kitten *, uint64_t * learned);
void kitten_shrink_to_clausal_core (kitten *);

void kitten_traverse_core_ids (kitten *, void *state,
                               void (*traverse) (void *state, unsigned id));

void kitten_traverse_core_clauses (kitten *, void *state,
                                   void (*traverse) (void *state,
                                                     bool learned, size_t,
                                                     const unsigned *));
```

time limit based on "tick" (similar to "mems")

← core shrinking (solve twice)

core clauses → gate clauses

Conclusion

- embedded SAT solver KITTEN
- semantically extract gates (← core definitions)
- efficient & effective
- eliminates more variables
- more potential use: sweeping