Binary Decision Diagrams in SBCL

Jim Newton

EPITA/LRDE

September 22, 2026

Representing and Computing with Types in Dynamically Typed Languages

Extending Dynamic Language Expressivity to Accommodate Rationally Typed Sequences

Jim Edward Newton

Thesis defense for the title: Doctorat de l'Université Sorbonne Université

Directory: Pr. Thierry Géraud, EPITA/LRDE

Advisor: Dr. Didier Verna, EPITA/LRDE

November 20, 2018

Representing and Computing with Types in Dynamically Typed Languages

Extending Dynamic Language Expressivity to Accommodate Rationally Typed Sequences

Jim Edward Newton

Thesis defense for the title: Doctorat de l'Université Sorbonne Université

Directory: Pr. Thierry Géraud, EPITA/LRDE

Advisor: Dr. Didier Verna, EPITA/LRDE

November 20, 2018

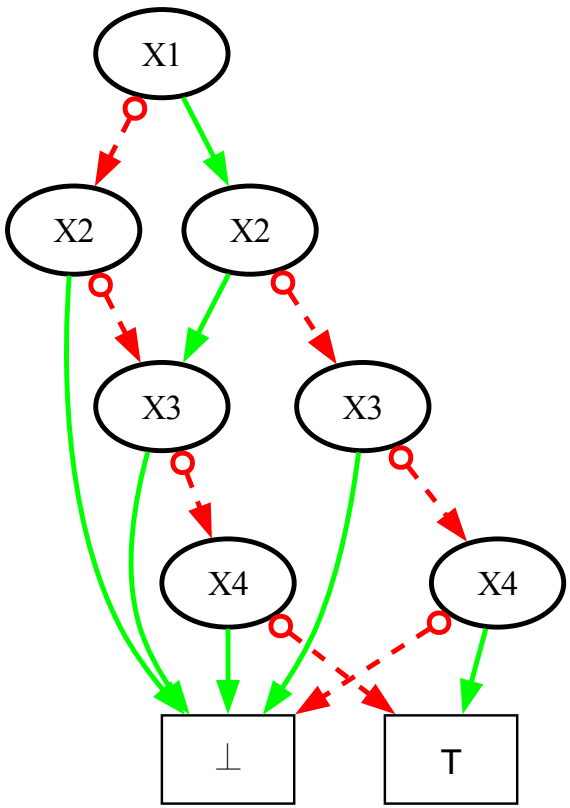
Many thanks to all the implementors and maintainers of SBCL!

Overview

BDD Construction

Measuring randomly generated BDDs

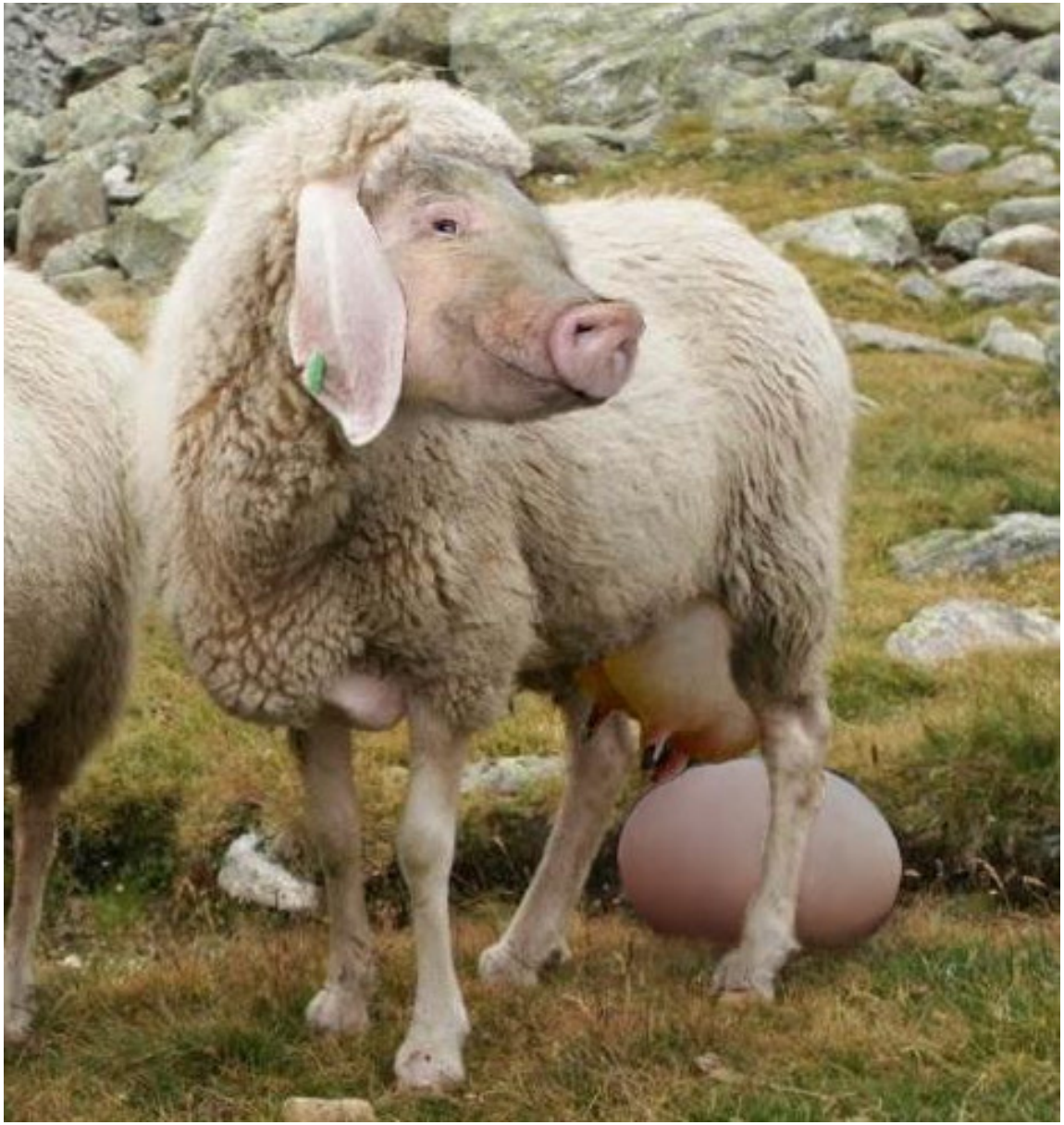
```
(or
  (and (not x1) (not x2) (not x3) (not x4))
  (and x1 x2 (not x3) (not x4))
  (and x1 (not x2) (not x3) x4))
```



This BDD contains 9 nodes.

BDD sizes can be large

$\approx 2^{n-\log n}$ in size



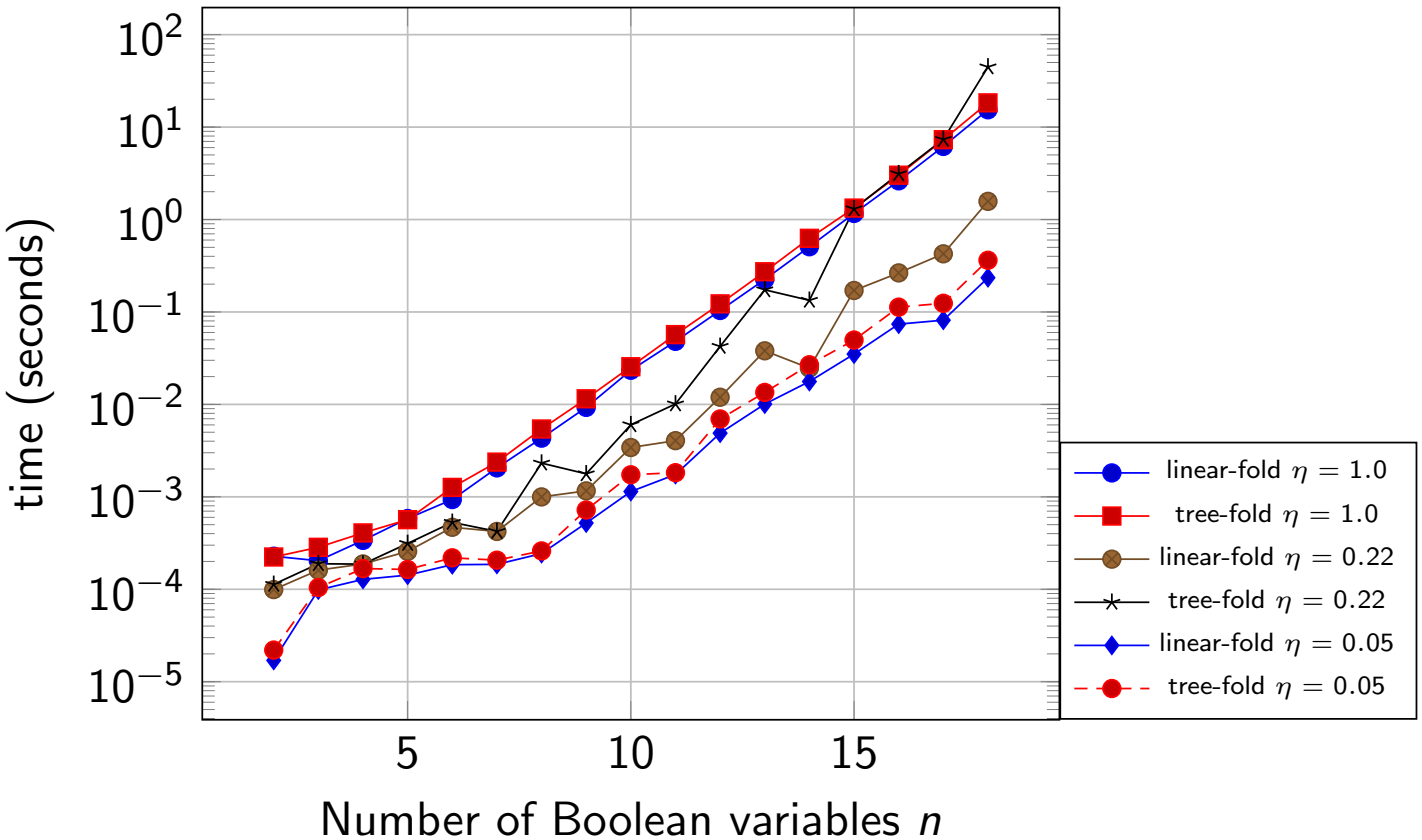
Creative Commons Attribution ShareAlike, Author: Georg Mittenecker

The BDD is the *Eierlegende Wollmilchsau* of Boolean algebra.

BDDs have many surprising features and uses.

Pointer equality,
Boolean Operations,
Canonical form,
Vacuity check

BDD construction times

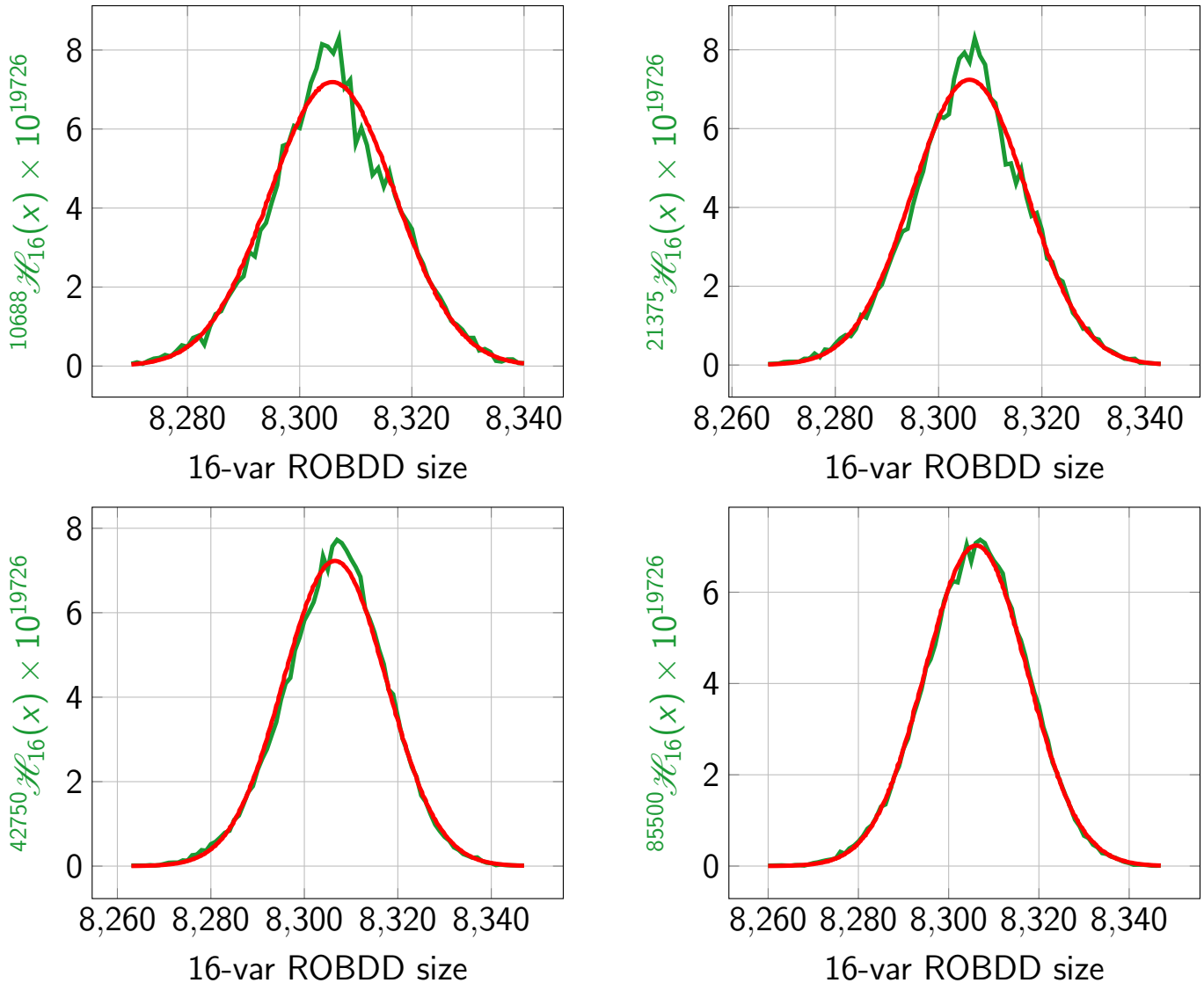


Construction times (from scratch) is necessarily exponential.

Can be amortized with caching, because the number of distinct subtrees is smaller than the number of trees.

More later.

BDD Size Histograms



Size distributions of 16-variable BDDs with stepped sample sizes.

Serializing BDD Histogram Data

Choosing a random 4-variable Boolean function is equivalent to filling out the 2^4 rows of an 4-Boolean variable by randomly selecting 0 or 1 for each row.

Minterm index	x_4	x_3	x_2	x_1	F	**Minterm**
0	0	0	0	0	1	$\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4$
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	1	1	1	$x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4$
4	0	1	0	0	0	
5	0	1	0	1	0	
6	0	1	1	0	0	
7	0	1	1	1	0	
8	1	0	0	0	0	
9	1	0	0	1	1	$x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4$
10	1	0	1	0	0	
11	1	0	1	1	0	
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	0	

Serializing BDD Histogram Data

Minterm index	x_4	x_3	x_2	x_1	F	**Minterm**
0	0	0	0	0	1	$\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4$
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	1	1	1	$x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4$
4	0	1	0	0	0	
5	0	1	0	1	0	
6	0	1	1	0	0	
7	0	1	1	1	0	
8	1	0	0	0	0	
9	1	0	0	1	1	$x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4$
10	1	0	1	0	0	
11	1	0	1	1	0	
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	0	

Serializing BDD Histogram Data

Minterm index	x_4	x_3	x_2	x_1	F	**Minterm**
0	0	0	0	0	1	$\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4$
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	1	1	1	$x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4$
4	0	1	0	0	0	
5	0	1	0	1	0	
6	0	1	1	0	0	
7	0	1	1	1	0	
8	1	0	0	0	0	
9	1	0	0	1	1	$x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4$
10	1	0	1	0	0	
11	1	0	1	1	0	
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	0	

This 4-variable truth table is characterized by the number in the F column.

Serializing BDD Histogram Data

Minterm index	x_4	x_3	x_2	x_1	F	**Minterm**
0	0	0	0	0	1	$\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4$
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	1	1	1	$x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4$
4	0	1	0	0	0	
5	0	1	0	1	0	
6	0	1	1	0	0	
7	0	1	1	1	0	
8	1	0	0	0	0	
9	1	0	0	1	1	$x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4$
10	1	0	1	0	0	
11	1	0	1	1	0	
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	0	

This 4-variable truth table is characterized by the number in the F column.
There are $2^4 = 16$ rows, in the truth table

Serializing BDD Histogram Data

Minterm index	x_4	x_3	x_2	x_1	F	**Minterm**
0	0	0	0	0	1	$\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4$
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	1	1	1	$x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4$
4	0	1	0	0	0	
5	0	1	0	1	0	
6	0	1	1	0	0	
7	0	1	1	1	0	
8	1	0	0	0	0	
9	1	0	0	1	1	$x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4$
10	1	0	1	0	0	
11	1	0	1	1	0	
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	0	

$$0000001000001001_2 = 1 \times 2^9 + 1 \times 2^3 + 1 \times 2^0$$
$$= 521_{10}$$

Serializing BIGNUM computations

$$\begin{aligned} 521_{10} &= 1 \times 2^9 + 1 \times 2^3 + 1 \times 2^0 = 0000001000001001_2 \\ &= 17 \times 36^1 + 14 \times 36^0 \\ &= EH_{36} \end{aligned}$$

```
(with-output-to-string (stream)
  (format stream "~36R ;" 41555856710009 4532))
;; returns the string "35NR7U935D1G ;"

(let ((*read-base* 36))
  (with-input-from-string (stream "35NR7U935D1G")
    (read stream)))
;; returns the integer 41555856710009 4532
```

Serializing histogram data to text file

```
8 76 3HYKNOKIES21OSIUF8EDK73P345TMJIEHAMM4SU1377EE6MZ2C ;
8 77 3FPE2XYQEDV5P8RUJIVANLRODX92ORAD8MKFJMEF6SOS6TUBOR ;
8 75 1NIYCE4MQBCFI6FO7F3WIFLUOBOPN1WF5SS12BZBZ4VI72NWN6 ;
8 77 D6S407RY5ST4SNIUV8CIGEOI2TTGVXGSGIO4RQITCM7VN91QJ ;
8 76 6868C2K8WMXDT5WM1YR4DG698VRSOZC9HFCU6J4U8RYOIKYK8D ;
8 72 644RONLW7O9FPNRP9U6ONC4N8X7BK7DQ2MOD88ONECAN36P7UB ;
8 74 TH4WLQZPMWYWUDRNYBAPQARM5BP9IP5PXHSN9BEME8KS5Q8P ;
8 75 2LHRTNM41J7MFQZI6A4O9GQJJUZJ3U3ZPETONZEV3EEILH7B3N ;
8 75 4QJG869771XXQ4PTMK1MQ5J58RWF5HWO35QUGZ6G9P76LKEV2I ;
8 72 4WCY1O43GHSP153B2RV5HBS5ZPBTAQWC6ZXIXG5X6NBSS8FJD ;
```

Uniqifying the entries

Using the `sort -u` and `sort -m -T` flags to keep histogram entries unique across multiple runs.

```
sh> mv file-1 tmp
sh> sort -u -T $PWD tmp > file-1
sh> sort -m -u -T tmp-dir file-1 file-2 ... > file-combo
```

Weak Hash Tables

Weak Hash Tables

```
(defvar *bdd-hash* nil)

(defun bdd-call-with-new-hash-strong (thunk)
  (let ((*bdd-hash* (make-hash-table
                     :test #'equal)))
    (funcall thunk)))

(defun bdd-call-with-new-hash-weak (thunk)
  (let ((*bdd-hash* (make-hash-table
                     :test #'equal
                     :weakness :value)))
    (funcall thunk)))
```

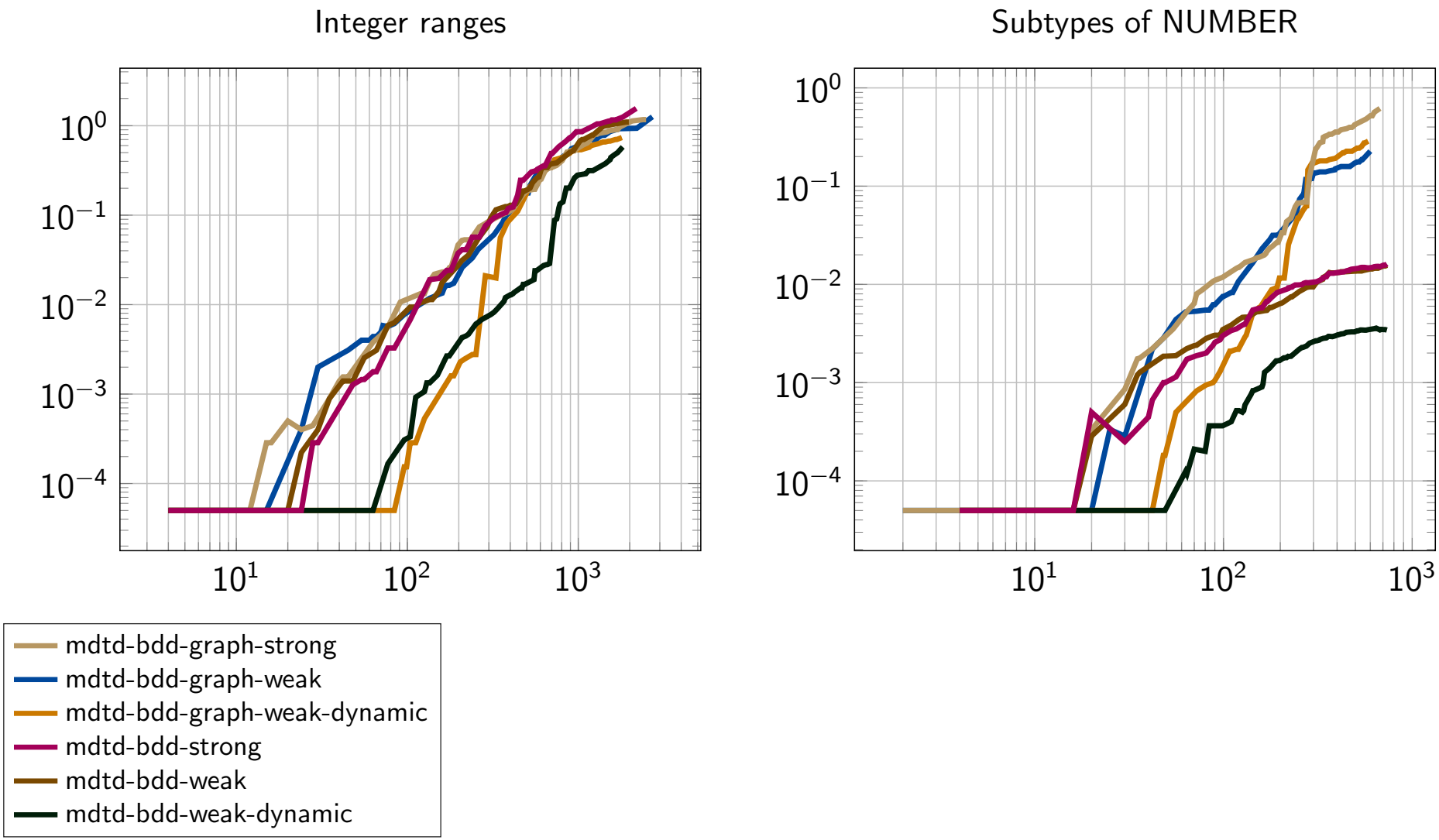
Weak Hash Tables

```
(defun bdd-call-with-new-hash-weak-dynamic (thunk)
  (if *bdd-hash*
      (funcall thunk)
      (bdd-call-with-new-hash-weak thunk)))

(defvar *bdd-call-with-new-hash*
    #'bdd-call-with-new-hash-weak-dynamic)

(defun bdd-call-with-new-hash (thunk)
  (funcall *bdd-call-with-new-hash* thunk))
```

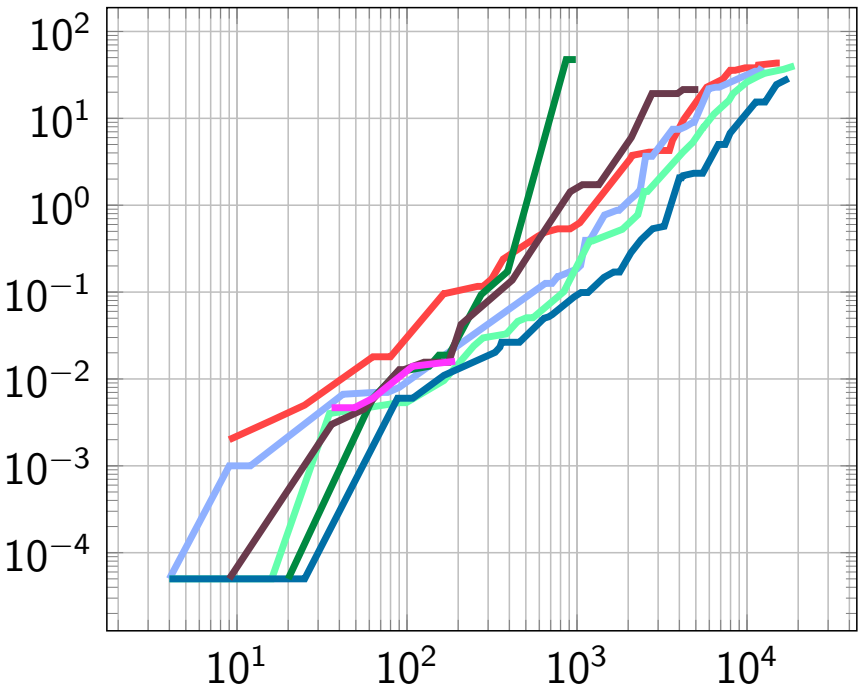
Performance of Weak Hash Tables



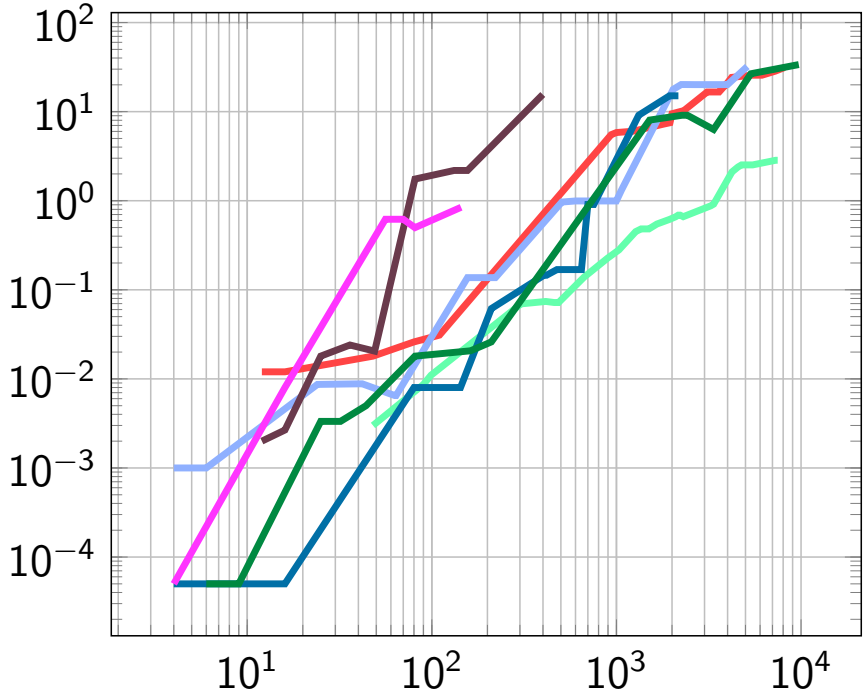
Performance Graphs

Performance Graphs

Real number ranges



Subtypes of NUMBER or CONDITION



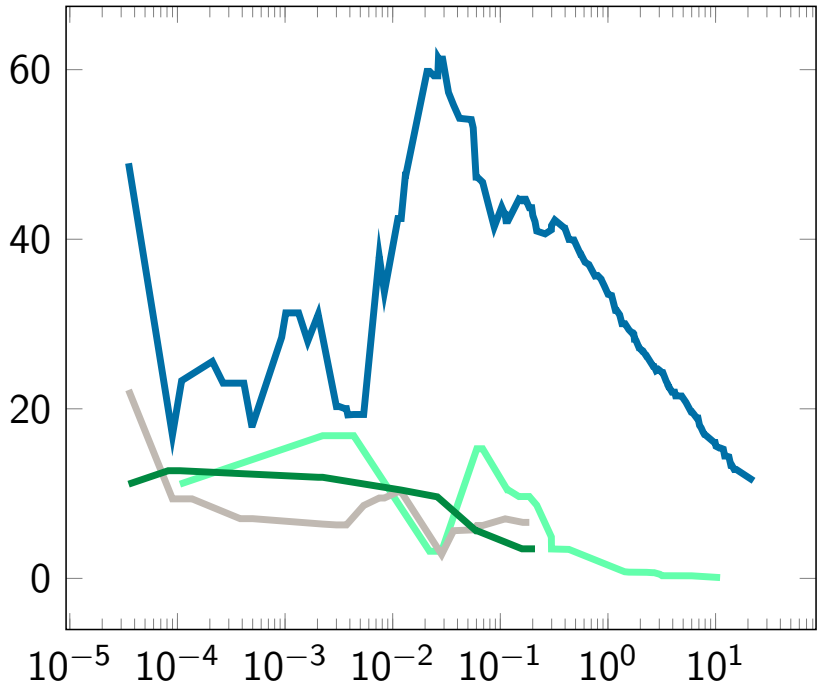
- mdtd-baseline
- mdtd-bdd
- mdtd-bdd-graph
- mdtd-graph
- mdtd-rtev2
- mdtd-sat
- parameterized-mdtd-bdd-graph

Parsing text output of profiler

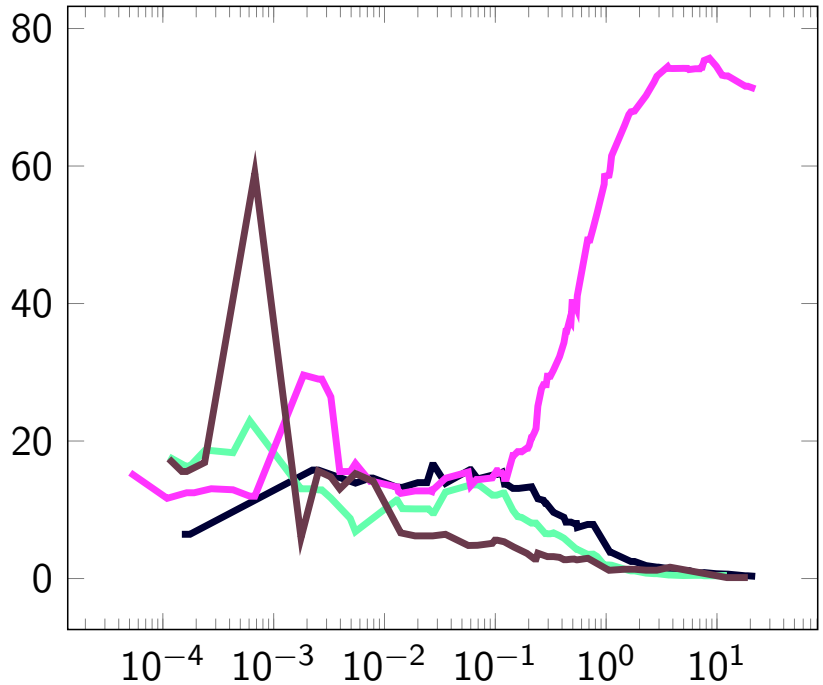
seconds	gc	consed	calls	sec/call	name
1.314	0.000	763,854,800	14,718	0.000089	RND-ELEMENT
0.974	0.967	0	10	0.097396	GARBAGE-COLLECT
0.317	0.000	293,328	20	0.015849	RUN-PROGRAM
0.007	0.000	360,448	10	0.000707	CHOOSE-RANDOMLY
0.001	0.000	0	2,120	0.000000	FIXED-POINT
0.000	0.000	0	520	0.000001	CACHED-SUBTYPEP
0.000	0.000	0	520	0.000000	ALPHABETIZE
0.000	0.000	0	840	0.000000	CMP-OBJECTS
0.000	0.000	0	520	0.000000	REDUCE-MEMBER-TYPE
0.000	0.000	0	10	0.000000	BDD-RECENT-COUNT
0.000	0.000	1,622,880	1,040	0.000000	CACHING-CALL
0.000	0.000	0	3,160	0.000000	ALPHABETIZE-TYPE
0.000	0.000	0	520	0.000000	DISJOINT-TYPES-P
2.618	0.967	766,131,472	26,699		Total

Parsing text output of profiler

Real number ranges



Integer ranges



`cached-subtypep-caching-call`	3342.67 seconds	245,140,975 calls
`subtypep-wrapper`	256.06 seconds	74,825,642 calls
`smarter-subtypep-caching-call`	231.86 seconds	11,203,966 calls
`bdd-find`	182.64 seconds	98,612,308 calls
`alphabetize-type`	163.27 seconds	277,943,114 calls
`fixed-point`	130.82 seconds	97,174,155 calls
`bdd-find-int-int`	125.61 seconds	67,442,808 calls
`slow-disjoint-types-p`	118.71 seconds	7,281,882 calls

Q/A

Questions?