# Summary

**Sink States**:$0(0 \times 10^0)$

Table 1: Pulse Analysis Summary

| Classes | Methods | States | Unsatisfiable Clauses | Unreachable States | Possible concurrent Methods | Total. no. of pairs | No. of concurrent pairs | Percentage of concurrent Methods |
|---|---|---|---|---|---|---|---|---|
| ConsListTest | 2 | 1 | 0 | 0 | 0 | 3 | 0 | 0 |
| ConsList | 28 | 1 | 0 | 0 | 27 | 406 | 257 | 63 |
| Triple | 5 | 1 | 0 | 0 | 5 | 15 | 13 | 87 |
| Pair | 10 | 1 | 0 | 0 | 9 | 55 | 19 | 35 |
| Empty | 11 | 1 | 0 | 0 | 10 | 66 | 52 | 79 |
| Utilities | 8 | 1 | 0 | 0 | 7 | 36 | 22 | 61 |
| CollectionMethods | 7 | 1 | 0 | 0 | 6 | 28 | 6 | 21 |
| Nonempty | 13 | 1 | 0 | 0 | 12 | 91 | 69 | 76 |
| Anonymous | 10 | 1 | 0 | 0 | 9 | 55 | 45 | 82 |
| PluralParseError | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| ImpossibleConstraint | 3 | 1 | 0 | 0 | 2 | 6 | 2 | 33 |
| ZeroFraction | 4 | 1 | 0 | 0 | 3 | 10 | 5 | 50 |
| AbstractFractionTermVisitor | 7 | 1 | 0 | 0 | 6 | 28 | 20 | 71 |
| VariableFraction | 8 | 1 | 0 | 0 | 7 | 36 | 22 | 61 |
| FractionConstraints | 28 | 1 | 0 | 0 | 27 | 406 | 48 | 12 |
| FractionConstraint | 5 | 1 | 0 | 0 | 4 | 15 | 10 | 67 |
| FractionAssignment | 23 | 1 | 0 | 0 | 22 | 276 | 28 | 10 |
| FractionRelation | 8 | 1 | 0 | 0 | 3 | 36 | 6 | 17 |
| Fraction | 14 | 1 | 0 | 0 | 13 | 105 | 76 | 72 |
| FractionSum | 6 | 1 | 0 | 0 | 2 | 21 | 3 | 14 |
| NamedFractionMapping | 2 | 1 | 0 | 0 | 0 | 3 | 0 | 0 |
| NamedFraction | 9 | 1 | 0 | 0 | 8 | 45 | 14 | 31 |
| VariableElimination | 14 | 1 | 0 | 0 | 13 | 105 | 16 | 15 |
| NormalizedFractionConstraint | 15 | 1 | 0 | 0 | 6 | 120 | 21 | 18 |
| Rational | 22 | 1 | 0 | 0 | 17 | 253 | 153 | 60 |
| GeneralizedSum | 8 | 1 | 0 | 0 | 1 | 36 | 1 | 3 |
| VariableRelativity | 5 | 1 | 0 | 0 | 0 | 15 | 0 | 0 |
| NormalizedFractionSum | 2 | 1 | 0 | 0 | 1 | 3 | 1 | 33 |
| SmtLibPrinter | 4 | 1 | 0 | 0 | 4 | 10 | 7 | 70 |
| SmtLibBenchmarkPrinter | 10 | 1 | 0 | 0 | 1 | 55 | 1 | 2 |
| Anonymous | 5 | 1 | 1 | 0 | 4 | 15 | 10 | 67 |
| OneFraction | 4 | 1 | 0 | 0 | 3 | 10 | 5 | 50 |
| FractionTerm | 4 | 1 | 0 | 0 | 3 | 10 | 6 | 60 |
| Anonymous | 3 | 1 | 1 | 0 | 2 | 6 | 3 | 50 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Anonymous | 5 | 1 | 5 | 0 | 4 | 15 | 10 | 67 |
| Anonymous | 5 | 1 | 5 | 0 | 4 | 15 | 10 | 67 |
| Relop | 2 | 1 | 0 | 0 | 2 | 3 | 2 | 67 |
| Anonymous | 5 | 1 | 5 | 0 | 4 | 15 | 10 | 67 |
| RelationFractionPair | 8 | 1 | 0 | 0 | 5 | 36 | 15 | 42 |
| Anonymous | 9 | 1 | 3 | 0 | 8 | 45 | 36 | 80 |
| SimpleFractionSum | 8 | 1 | 0 | 0 | 7 | 36 | 13 | 36 |
| Sumop | 2 | 1 | 0 | 0 | 2 | 3 | 2 | 67 |
| SimpleVariableRelativity | 4 | 1 | 0 | 0 | 1 | 10 | 1 | 10 |
| Anonymous | 6 | 1 | 6 | 0 | 5 | 21 | 15 | 71 |
| FractionElimination | 8 | 1 | 0 | 0 | 7 | 36 | 8 | 22 |
| FractionPair | 5 | 1 | 0 | 0 | 0 | 15 | 0 | 0 |
| Anonymous | 9 | 1 | 9 | 0 | 8 | 45 | 36 | 80 |
| Anonymous | 3 | 1 | 1 | 0 | 2 | 6 | 3 | 50 |
| SmtLibConstraintProcessor | 10 | 1 | 0 | 0 | 9 | 55 | 25 | 45 |
| Impossible | 4 | 1 | 0 | 0 | 3 | 10 | 6 | 60 |
| Total Classes=50 | 401 | 50 | 36 | 0 | 308 | 2747 | 1133 | 41 |

## Contents

## 1 ConsListTest

Table 2: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| ConsListTest | $\surd$ |
| testList | $\surd$ |

Table 3: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 4: Methods Concurrency Matrix

| | ConsListTest | testList |
|---|---|---|
| ConsListTest | $\nparallel$ | $\nparallel$ |
| testList | $\nparallel$ | $\nparallel$ |

## 2 ConsList

Table 5: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| ConsList | √ |
| removeElement | √ |
| cons | √ |
| empty | √ |
| singleton | √ |
| list | √ |
| concat | √ |
| removeElementOnce | √ |
| map | √ |
| filter | √ |
| foldl | √ |
| listIterator | √ |
| subListSameTail | √ |
| get | √ |
| iterator | √ |
| subList | √ |
| contains | √ |
| toArray | √ |
| impossible | √ |
| add | √ |
| addAll | √ |
| clear | √ |
| remove | √ |
| removeOverload | √ |
| removeAll | √ |
| retainAll | √ |
| set | √ |
| main | √ |

Table 6: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 7: Methods Concurrency Matrix

| | ConsList | removeElement | cons | empty | singleton | list | concat | removeElementOnce | map | filter | foldl | listIterator | subListSameTail | get | iterator | subList | contains | toArray | impossible | add | addAll | clear | remove |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ConsList | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| removeElement | ∦ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| cons | ∦ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| empty | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| singleton | ∦ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| list | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| concat | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| removeElementOnce | ∦ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| map | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| filter | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| foldl | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| listIterator | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| subListSameTail | ∦ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| get | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| iterator | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| subList | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| contains | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| toArray | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| impossible | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| add | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| addAll | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| clear | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| remove | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| removeOverload | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| removeAll | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| retainAll | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| set | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| main | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 3 Triple

Table 8: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Triple | $\surd$ |
| fst | $\surd$ |
| snd | $\surd$ |
| thrd | $\surd$ |
| createTriple | $\surd$ |

Table 9: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 10: Methods Concurrency Matrix

| | Triple | fst | snd | thrd | createTriple |
|---|---|---|---|---|---|
| Triple | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ |
| fst | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| snd | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| thrd | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| createTriple | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 4 Pair

Table 11: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Pair | √ |
| fst | √ |
| setComponent1 | √ |
| snd | √ |
| setComponent2 | √ |
| clone | √ |
| toString | √ |
| hashCode | √ |
| equals | √ |
| create | √ |

Table 12: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 13: Methods Concurrency Matrix

| | Pair | fst | setComponent1 | snd | setComponent2 | clone | toString | hashCode | equals | create |
|---|---|---|---|---|---|---|---|---|---|---|
| Pair | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| fst | ∦ | ∥ | ∦ | ∥ | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ |
| setComponent1 | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| snd | ∦ | ∥ | ∦ | ∥ | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ |
| setComponent2 | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| clone | ∦ | ∥ | ∦ | ∥ | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ |
| toString | ∦ | ∥ | ∦ | ∥ | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ |
| hashCode | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| equals | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| create | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 5 Empty

Table 14: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Empty | √ |
| hd | √ |
| indexOf | √ |
| isEmpty | √ |
| lastIndexOf | √ |
| size | √ |
| tl | √ |
| indexOfHelper | √ |
| lastIndexOfHelper | √ |
| toString | √ |
| containsAll | √ |

Table 15: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 16: Methods Concurrency Matrix

| | Empty | hd | indexOf | isEmpty | lastIndexOf | size | tl | indexOfHelper | lastIndexOfHelper | toString | containsAll |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Empty | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| hd | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| indexOf | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| isEmpty | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| lastIndexOf | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| size | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| tl | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∦ |
| indexOfHelper | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| lastIndexOfHelper | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| toString | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| containsAll | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∦ |

## 6 Utilities

Table 17: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Utilities | √ |
| ASTNodeToString | √ |
| ModifierToString | √ |
| getMethodDeclaration | √ |
| methodDeclarationToString | √ |
| nyi | √ |
| nyiOverload | √ |
| main | √ |

Table 18: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 19: Methods Concurrency Matrix

| | Utilities | ASTNodeToString | ModifierToString | getMethodDeclaration | methodDeclarationToString | nyi | nyiOverload | main |
|---|---|---|---|---|---|---|---|---|
| Utilities | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| ASTNodeToString | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| ModifierToString | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| getMethodDeclaration | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ | ∥ | ∦ |
| methodDeclarationToString | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ | ∥ | ∦ |
| nyi | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| nyiOverload | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| main | ∦ | ∥ | ∥ | ∦ | ∦ | ∥ | ∥ | ∦ |

## 7 CollectionMethods

Table 20: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| CollectionMethods | $\surd$ |
| map | $\surd$ |
| concat | $\surd$ |
| union | $\surd$ |
| addToMultiMap | $\surd$ |
| createSetWithoutElement | $\surd$ |
| mutableSet | $\surd$ |

Table 21: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 22: Methods Concurrency Matrix

| | CollectionMethods | map | concat | union | addToMultiMap | createSetWithoutElement | mutableSet |
|---|---|---|---|---|---|---|---|
| CollectionMethods | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| map | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ |
| concat | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ |
| union | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ |
| addToMultiMap | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ |
| createSetWithoutElement | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| mutableSet | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ |

## 8 Nonempty

Table 23: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Nonempty | $\surd$ |
| hd | $\surd$ |
| indexOfHelper | $\surd$ |
| indexOf | $\surd$ |
| isEmpty | $\surd$ |
| lastIndexOf | $\surd$ |
| lastIndexOfHelper | $\surd$ |
| size | $\surd$ |
| tl | $\surd$ |
| toString | $\surd$ |
| hashCode | $\surd$ |
| equals | $\surd$ |
| containsAll | $\surd$ |

Table 24: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 25: Methods Concurrency Matrix

| | Nonempty | hd | indexOfHelper | indexOf | isEmpty | lastIndexOf | lastIndexOfHelper | size | tl | toString | hashCode | equals | containsAll |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nonempty | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| hd | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ |
| indexOfHelper | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| indexOf | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| isEmpty | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| lastIndexOf | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| lastIndexOfHelper | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| size | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ |
| tl | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ |
| toString | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ |
| equals | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ |
| containsAll | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 9 Anonymous

Table 26: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | $\surd$ |
| add | $\surd$ |
| hasNext | $\surd$ |
| hasPrevious | $\surd$ |
| next | $\surd$ |
| nextIndex | $\surd$ |
| previous | $\surd$ |
| previousIndex | $\surd$ |
| remove | $\surd$ |
| set | $\surd$ |

Table 27: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 28: Methods Concurrency Matrix

| | Anonymous | add | hasNext | hasPrevious | next | nextIndex | previous | previousIndex | remove | set |
|---|---|---|---|---|---|---|---|---|---|---|
| Anonymous | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| add | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| hasNext | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| hasPrevious | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| next | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| nextIndex | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| previous | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| previousIndex | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| remove | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| set | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 10 PluralParseError

Table 29: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| PluralParseError | √ |

Table 30: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

## 11 ImpossibleConstraint

Table 31: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| ImpossibleConstraint | $\surd$ |
| dispatch | $\surd$ |
| toString | $\surd$ |

Table 32: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 33: Methods Concurrency Matrix

| | ImpossibleConstraint | dispatch | toString |
|---|---|---|---|
| ImpossibleConstraint | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| dispatch | $\nparallel$ | $\nparallel$ | $\parallel$ |
| toString | $\nparallel$ | $\parallel$ | $\parallel$ |

## 12 ZeroFraction

Table 34: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| ZeroFraction | $\surd$ |
| isZero | $\surd$ |
| dispatch | $\surd$ |
| toString | $\surd$ |

Table 35: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 36: Methods Concurrency Matrix

| | ZeroFraction | isZero | dispatch | toString |
|---|---|---|---|---|
| ZeroFraction | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isZero | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| dispatch | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| toString | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 13 AbstractFractionTermVisitor

Table 37: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| AbstractFractionTermVisitor | √ |
| named | √ |
| one | √ |
| var | √ |
| zero | √ |
| literal | √ |
| sum | √ |

Table 38: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 39: Methods Concurrency Matrix

| | AbstractFractionTermVisitor | named | one | var | zero | literal | sum |
|---|---|---|---|---|---|---|---|
| AbstractFractionTermVisitor | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| named | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| one | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| var | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| zero | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| literal | ∦ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |
| sum | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 14 VariableFraction

Table 40: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| VariableFraction | √ |
| getVarName | √ |
| isVariable | √ |
| dispatch | √ |
| compareToVar | √ |
| toString | √ |
| hashCode | √ |
| equals | √ |

Table 41: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 42: Methods Concurrency Matrix

| | VariableFraction | getVarName | isVariable | dispatch | compareToVar | toString | hashCode | equals |
|---|---|---|---|---|---|---|---|---|
| VariableFraction | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| getVarName | ∦ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ |
| isVariable | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| dispatch | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ |
| compareToVar | ∦ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ |
| toString | ∦ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ |
| hashCode | ∦ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ |
| equals | ∦ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ |

## 15 FractionConstraints

Table 43: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionConstraints | √ |
| createMutable | √ |
| addConstraint | √ |
| testConstraint | √ |
| isConsistent | √ |
| isConsistentInternal | √ |
| isImpossible | √ |
| simplify | √ |
| simplifyInternal | √ |
| toString | √ |
| getConstraints | √ |
| mutableCopy | √ |
| mutableCopyOverload | √ |
| testConstraints | √ |
| addAll | √ |
| getVariables | √ |
| getConstants | √ |
| getUniversalParameters | √ |
| registerFractions | √ |
| atLeastAsPrecise | √ |
| freeze | √ |
| concat | √ |
| seemsConsistent | √ |
| hashCode | √ |
| equals | √ |
| newVariableFraction | √ |
| newNamedFraction | √ |
| isKnown | √ |

Table 44: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 45: Methods Concurrency Matrix

| | FractionConstraints | createMutable | addConstraint | testConstraint | isConsistent | isConsistentInternal | isImpossible | simplify | simplifyInternal | toString | getConstraints | mutableCopy | mutableCopyOverload | testConstraints | addAll | getVariables | getConstants | getUniversalParameters | registerFractions | atLeastAsPrecise | freeze |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FractionConstraints | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| createMutable | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| addConstraint | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| testConstraint | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isConsistent | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isConsistentInternal | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isImpossible | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ |
| simplify | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| simplifyInternal | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| toString | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| getConstraints | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ |
| mutableCopy | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| mutableCopyOverload | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| testConstraints | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addAll | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| getVariables | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ |
| getConstants | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ |
| getUniversalParameters | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ |
| registerFractions | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| atLeastAsPrecise | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ |
| freeze | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| concat | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| seemsConsistent | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| hashCode | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| equals | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| newVariableFraction | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| newNamedFraction | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isKnown | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |

## 16 FractionConstraint

Table 46: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionConstraint | $\surd$ |
| impossible | $\surd$ |
| createEquality | $\surd$ |
| createLessThan | $\surd$ |
| createLessThanOrEqual | $\surd$ |

Table 47: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 48: Methods Concurrency Matrix

| | FractionConstraint | impossible | createEquality | createLessThan | createLessThanOrEqual |
|---|---|---|---|---|---|
| FractionConstraint | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| impossible | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| createEquality | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| createLessThan | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| createLessThanOrEqual | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 17 FractionAssignment

Table 49: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionAssignment | √ |
| resetChangedFlag | √ |
| makeEquivalentOverload | √ |
| union | √ |
| mutableSet | √ |
| isZero | √ |
| getLiteral | √ |
| isOne | √ |
| areEquivalent | √ |
| areEquivalentOverload | √ |
| makeZero | √ |
| makeZeroOverload | √ |
| makeOne | √ |
| makeNonZero | √ |
| isNonZero | √ |
| isChanged | √ |
| isConsistent | √ |
| sumsToConstant | √ |
| equivalentLiteralValues | √ |
| makeEquivalent | √ |
| getConstant | √ |
| getRepresentative | √ |
| toString | √ |

Table 50: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 51: Methods Concurrency Matrix

| | FractionAssignment | resetChangedFlag | makeEquivalentOverload | union | mutableSet | isZero | getLiteral | isOne | areEquivalent | areEquivalentOverload | makeZero | makeZeroOverload | makeOne | makeNonZero | isNonZero | isChanged | isConsistent | sumsToConstant | equivalentLiteralValues | makeEquivalent | getConstant |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FractionAssignment | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| resetChangedFlag | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| makeEquivalentOverload | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| union | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mutableSet | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| isZero | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getLiteral | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isOne | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| areEquivalent | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| areEquivalentOverload | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| makeZero | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| makeZeroOverload | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| makeOne | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| makeNonZero | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isNonZero | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isChanged | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| isConsistent | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| sumsToConstant | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| equivalentLiteralValues | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| makeEquivalent | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getConstant | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getRepresentative | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| toString | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

## 18 FractionRelation

Table 52: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionRelation | $\surd$ |
| getRelop | $\surd$ |
| getTerms | $\surd$ |
| dispatch | $\surd$ |
| toString | $\surd$ |
| hashCode | $\surd$ |
| equals | $\surd$ |
| compareTo | $\surd$ |

Table 53: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 54: Methods Concurrency Matrix

| | FractionRelation | getRelop | getTerms | dispatch | toString | hashCode | equals | compareTo |
|---|---|---|---|---|---|---|---|---|
| FractionRelation | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getRelop | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getTerms | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| dispatch | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| toString | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| equals | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| compareTo | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

# 19 Fraction

Table 55: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Fraction | √ |
| zero | √ |
| one | √ |
| isZero | √ |
| isOne | √ |
| isVariable | √ |
| isNamed | √ |
| createNamed | √ |
| dispatch | √ |
| createExplicit | √ |
| isFixed | √ |
| isNeitherZeroNorOne | √ |
| isGuaranteedGreaterThanZero | √ |
| isPossiblyGreaterOrEqual | √ |

Table 56: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 57: Methods Concurrency Matrix

| | Fraction | zero | one | isZero | isOne | isVariable | isNamed | createNamed | dispatch | createExplicit | isFixed | isNeitherZeroNorOne | isGuaranteedGreaterThanZero | isPossiblyGreaterOrEqual |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fraction | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| zero | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ‖ | ‖ | ‖ | ∦ | ∦ |
| one | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ‖ | ‖ | ‖ | ∦ | ∦ |
| isZero | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |
| isOne | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |
| isVariable | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |
| isNamed | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |
| createNamed | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |
| dispatch | ∦ | ∦ | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ∦ | ‖ | ‖ | ∦ | ∦ |
| createExplicit | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ‖ | ‖ | ‖ | ∦ | ∦ |
| isFixed | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |
| isNeitherZeroNorOne | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ | ‖ |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| isGuaranteedGreaterThanZero | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| isPossiblyGreaterOrEqual | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |

## 20 FractionSum

Table 58: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionSum | $\surd$ |
| getSummands | $\surd$ |
| dispatch | $\surd$ |
| toString | $\surd$ |
| hashCode | $\surd$ |
| equals | $\surd$ |

Table 59: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 60: Methods Concurrency Matrix

| | FractionSum | getSummands | dispatch | toString | hashCode | equals |
|---|---|---|---|---|---|---|
| FractionSum | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getSummands | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| dispatch | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| toString | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| equals | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

## 21 NamedFractionMapping

Table 61: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| NamedFractionMapping | $\surd$ |
| map | $\surd$ |

Table 62: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 63: Methods Concurrency Matrix

| | NamedFractionMapping | map |
|---|---|---|
| NamedFractionMapping | $\nparallel$ | $\nparallel$ |
| map | $\nparallel$ | $\nparallel$ |

## 22 NamedFraction

Table 64: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| NamedFraction | √ |
| equals | √ |
| getVarName | √ |
| isVariable | √ |
| isJoinVariable | √ |
| isNamed | √ |
| dispatch | √ |
| toString | √ |
| hashCode | √ |

Table 65: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 66: Methods Concurrency Matrix

| | NamedFraction | equals | getVarName | isVariable | isJoinVariable | isNamed | dispatch | toString | hashCode |
|---|---|---|---|---|---|---|---|---|---|
| NamedFraction | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| equals | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ |
| getVarName | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ |
| isVariable | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ |
| isJoinVariable | ∦ | ∦ | ∥ | ∥ | ∥ | ∥ | ∦ | ∦ | ∦ |
| isNamed | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| dispatch | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ |
| toString | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ |
| hashCode | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ |

## 23 VariableElimination

Table 67: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| VariableElimination | √ |
| eliminateVariables | √ |
| getTimeout | √ |
| normalizeConstraints | √ |
| eliminationOrder | √ |
| addVariableConstraints | √ |
| eliminateFraction | √ |
| addConstConstraints | √ |
| isConsistent | √ |
| isSatisfiable | √ |
| isPrimitiveConstraintSatisfiable | √ |
| setTimeout | √ |
| collectVariables | √ |
| normalizeTerm | √ |

Table 68: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 69: Methods Concurrency Matrix

| | VariableElimination | eliminateVariables | getTimeout | normalizeConstraints | eliminationOrder | addVariableConstraints | eliminateFraction | addConstConstraints | isConsistent | isSatisfiable | isPrimitiveConstraintSatisfiable | setTimeout | collectVariables | normalizeTerm |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VariableElimination | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| eliminateVariables | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| getTimeout | ∦ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| normalizeConstraints | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| eliminationOrder | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| addVariableConstraints | ∦ | ∦ | ∥ | ∦ | ∦ | ∥ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| eliminateFraction | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| addConstConstraints | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| isConsistent | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| isSatisfiable | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| isPrimitiveConstraintSatisfiable | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| setTimeout | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| collectVariables | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| normalizeTerm | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 24 NormalizedFractionConstraint

Table 70: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| NormalizedFractionConstraint | √ |
| createConstraintOverload | √ |
| isolateFraction | √ |
| getRelop | √ |
| createConstraint | √ |
| isTrueWithAssumptions | √ |
| isTriviallyTrue | √ |
| dominates | √ |
| equals | √ |
| isRangeConstraint | √ |
| isPrimitive | √ |
| getRight | √ |
| getLeft | √ |
| toString | √ |
| hashCode | √ |

Table 71: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 72: Methods Concurrency Matrix

| | NormalizedFractionConstraint | createConstraintOverload | isolateFraction | getRelop | createConstraint | isTrueWithAssumptions | isTriviallyTrue | dominates | equals | isRangeConstraint | isPrimitive | getRight | getLeft | toString | hashCode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NormalizedFractionConstraint | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| createConstraintOverload | ∦ | ‖ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ‖ | ‖ | ∦ | ∦ |
| isolateFraction | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| getRelop | ∦ | ‖ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ‖ | ‖ | ∦ | ∦ |
| createConstraint | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isTrueWithAssumptions | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isTriviallyTrue | ∦ | ‖ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ‖ | ‖ | ∦ | ∦ |
| dominates | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| equals | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isRangeConstraint | ∦ | ‖ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ∦ | ‖ | ∦ | ‖ | ‖ | ∦ | ∦ |
| isPrimitive | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| getRight | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| getLeft | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| toString | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

## 25 Rational

Table 73: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Rational | √ |
| one | √ |
| minusOne | √ |
| isZero | √ |
| abs | √ |
| isPositive | √ |
| negation | √ |
| gcd | √ |
| div | √ |
| plus | √ |
| zero | √ |
| minus | √ |
| isNegative | √ |
| isSmallerThan | √ |
| times | √ |
| inverse | √ |
| getP | √ |
| getQ | √ |
| isOne | √ |
| toString | √ |
| hashCode | √ |
| equals | √ |

Table 74: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 75: Methods Concurrency Matrix

| | Rational | one | minusOne | isZero | abs | isPositive | negation | gcd | div | plus | zero | minus | isNegative | isSmallerThan | times | inverse | getP | getQ | isOne | toString | hashCode | equals |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rational | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| one | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| minusOne | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| isZero | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| abs | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| isPositive | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| negation | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| gcd | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| div | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∦ | ∦ | ∥ | ∥ | ∥ | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| plus | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| zero | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| minus | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isNegative | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| isSmallerThan | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| times | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| inverse | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| getP | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| getQ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| isOne | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| toString | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| hashCode | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| equals | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 26 GeneralizedSum

Table 76: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| GeneralizedSum | $\surd$ |
| getFractions | $\surd$ |
| getCoefficient | $\surd$ |
| equals | $\surd$ |
| getConstant | $\surd$ |
| isGround | $\surd$ |
| toString | $\surd$ |
| hashCode | $\surd$ |

Table 77: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 78: Methods Concurrency Matrix

| | GeneralizedSum | getFractions | getCoefficient | equals | getConstant | isGround | toString | hashCode |
|---|---|---|---|---|---|---|---|---|
| GeneralizedSum | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getFractions | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getCoefficient | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| equals | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getConstant | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isGround | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| toString | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

## 27 VariableRelativity

Table 79: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| VariableRelativity | $\surd$ |
| addRight | $\surd$ |
| addLeft | $\surd$ |
| dumpRelations | $\surd$ |
| dumpRelation | $\surd$ |

Table 80: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 81: Methods Concurrency Matrix

| | VariableRelativity | addRight | addLeft | dumpRelations | dumpRelation |
|---|---|---|---|---|---|
| VariableRelativity | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| addRight | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| addLeft | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| dumpRelations | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| dumpRelation | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

## 28 NormalizedFractionSum

Table 82: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| NormalizedFractionSum | $\surd$ |
| zero | $\surd$ |

Table 83: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 84: Methods Concurrency Matrix

| | NormalizedFractionSum | zero |
|---|---|---|
| NormalizedFractionSum | $\nparallel$ | $\nparallel$ |
| zero | $\nparallel$ | $\parallel$ |

## 29 SmtLibPrinter

Table 85: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| SmtLibPrinter | $\surd$ |
| toString | $\surd$ |
| printStatus | $\surd$ |
| getInverse | $\surd$ |

Table 86: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 87: Methods Concurrency Matrix

| | SmtLibPrinter | toString | printStatus | getInverse |
|---|---|---|---|---|
| SmtLibPrinter | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ |
| toString | $\nparallel$ | $\nparallel$ | $\parallel$ | $\parallel$ |
| printStatus | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| getInverse | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 30 SmtLibBenchmarkPrinter

Table 88: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| SmtLibBenchmarkPrinter | √ |
| addLineComment | √ |
| addStatus | √ |
| addFormula | √ |
| addUnknown | √ |
| addAssumption | √ |
| appendConjunction | √ |
| addNegatedQuantifiedImplicationFormula | √ |
| appendExists | √ |
| getResult | √ |

Table 89: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 90: Methods Concurrency Matrix

| | SmtLibBenchmarkPrinter | addLineComment | addStatus | addFormula | addUnknown | addAssumption | appendConjunction | addNegatedQuantifiedImplicationFormula | appendExists | getResult |
|---|---|---|---|---|---|---|---|---|---|---|
| SmtLibBenchmarkPrinter | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addLineComment | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addStatus | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addFormula | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addUnknown | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addAssumption | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| appendConjunction | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| addNegatedQuantifiedImplicationFormula | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| appendExists | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| getResult | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |

## 31 Anonymous

Table 91: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| named | √ |
| one | √ |
| var | √ |
| zero | √ |

Table 92: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 93: Methods Concurrency Matrix

| | Anonymous | named | one | var | zero |
|---|---|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ | ∦ | ∦ |
| named | ∦ | ∥ | ∥ | ∥ | ∥ |
| one | ∦ | ∥ | ∥ | ∥ | ∥ |
| var | ∦ | ∥ | ∥ | ∥ | ∥ |
| zero | ∦ | ∥ | ∥ | ∥ | ∥ |

## 32 OneFraction

Table 94: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| OneFraction | $\surd$ |
| isOne | $\surd$ |
| dispatch | $\surd$ |
| toString | $\surd$ |

Table 95: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 96: Methods Concurrency Matrix

| | OneFraction | isOne | dispatch | toString |
|---|---|---|---|---|
| OneFraction | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| isOne | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| dispatch | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| toString | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 33 FractionTerm

Table 97: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionTerm | $\surd$ |
| createSum | $\surd$ |
| createSumOverload | $\surd$ |
| compareTo | $\surd$ |

Table 98: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 99: Methods Concurrency Matrix

| | FractionTerm | createSum | createSumOverload | compareTo |
|---|---|---|---|---|
| FractionTerm | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| createSum | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| createSumOverload | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| compareTo | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 34 Anonymous

Table 100: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| literal | √ |
| sum | √ |

Table 101: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 102: Methods Concurrency Matrix

| | Anonymous | literal | sum |
|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ |
| literal | ∦ | ∥ | ∥ |
| sum | ∦ | ∥ | ∥ |

## 35 Anonymous

Table 103: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| named | × |
| one | × |
| var | × |
| zero | × |

Table 104: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 105: Methods Concurrency Matrix

| | Anonymous | named | one | var | zero |
|---|---|---|---|---|---|
| Anonymous | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| named | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| one | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| var | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| zero | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 36 Anonymous

Table 106: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| one | × |
| zero | × |
| named | × |
| var | × |

Table 107: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 108: Methods Concurrency Matrix

| | Anonymous | one | zero | named | var |
|---|---|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ | ∦ | ∦ |
| one | ∦ | ∥ | ∥ | ∥ | ∥ |
| zero | ∦ | ∥ | ∥ | ∥ | ∥ |
| named | ∦ | ∥ | ∥ | ∥ | ∥ |
| var | ∦ | ∥ | ∥ | ∥ | ∥ |

## 37 Relop

Table 109: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Relop | $\surd$ |
| toString | $\surd$ |

Table 110: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 111: Methods Concurrency Matrix

| | Relop | toString |
|---|---|---|
| Relop | $\nparallel$ | $\parallel$ |
| toString | $\parallel$ | $\parallel$ |

## 38 Anonymous

Table 112: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| named | × |
| one | × |
| var | × |
| zero | × |

Table 113: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 114: Methods Concurrency Matrix

| | Anonymous | named | one | var | zero |
|---|---|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ | ∦ | ∦ |
| named | ∦ | ∥ | ∥ | ∥ | ∥ |
| one | ∦ | ∥ | ∥ | ∥ | ∥ |
| var | ∦ | ∥ | ∥ | ∥ | ∥ |
| zero | ∦ | ∥ | ∥ | ∥ | ∥ |

## 39 RelationFractionPair

Table 115: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| RelationFractionPair | √ |
| createEqual | √ |
| createLeq | √ |
| createLess | √ |
| getRelop | √ |
| toString | √ |
| hashCode | √ |
| equals | √ |

Table 116: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 117: Methods Concurrency Matrix

| | RelationFractionPair | createEqual | createLeq | createLess | getRelop | toString | hashCode | equals |
|---|---|---|---|---|---|---|---|---|
| RelationFractionPair | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| createEqual | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ∦ |
| createLeq | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ∦ |
| createLess | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ∦ |
| getRelop | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ∦ |
| toString | ∦ | ‖ | ‖ | ‖ | ‖ | ‖ | ∦ | ∦ |
| hashCode | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| equals | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |

## 40 Anonymous

Table 118: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| createRelation | √ |
| compare | √ |
| getRepresentative | √ |
| getRepresentatives | √ |
| impossible | √ |
| relation | √ |
| literal | × |
| sum | × |

Table 119: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 120: Methods Concurrency Matrix

| | Anonymous | createRelation | compare | getRepresentative | getRepresentatives | impossible | relation | literal | sum |
|---|---|---|---|---|---|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| createRelation | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| compare | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| getRepresentative | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| getRepresentatives | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| impossible | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| relation | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| literal | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| sum | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 41 SimpleFractionSum

Table 121: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| SimpleFractionSum | $\surd$ |
| createAdd | $\surd$ |
| createSub | $\surd$ |
| getSumop | $\surd$ |
| toString | $\surd$ |
| dispatch | $\surd$ |
| hashCode | $\surd$ |
| equals | $\surd$ |

Table 122: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 123: Methods Concurrency Matrix

| | SimpleFractionSum | createAdd | createSub | getSumop | toString | dispatch | hashCode | equals |
|---|---|---|---|---|---|---|---|---|
| SimpleFractionSum | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| createAdd | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| createSub | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| getSumop | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| toString | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| dispatch | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |
| equals | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ |

## 42 Sumop

Table 124: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Sumop | $\surd$ |
| toString | $\surd$ |

Table 125: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 126: Methods Concurrency Matrix

| | Sumop | toString |
|---|---|---|
| Sumop | $\nparallel$ | $\parallel$ |
| toString | $\parallel$ | $\parallel$ |

## 43 SimpleVariableRelativity

Table 127: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| SimpleVariableRelativity | $\surd$ |
| addRight | $\surd$ |
| addLeft | $\surd$ |
| dumpRelations | $\surd$ |

Table 128: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 129: Methods Concurrency Matrix

| | SimpleVariableRelativity | addRight | addLeft | dumpRelations |
|---|---|---|---|---|
| SimpleVariableRelativity | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| addRight | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| addLeft | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| dumpRelations | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ |

## 44 Anonymous

Table 130: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| named | × |
| one | × |
| sum | × |
| var | × |
| zero | × |

Table 131: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 132: Methods Concurrency Matrix

| | Anonymous | named | one | sum | var | zero |
|---|---|---|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| named | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| one | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| sum | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| var | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |
| zero | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 45 FractionElimination

Table 133: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionElimination | √ |
| eliminateVariableOverload | √ |
| containsVariable | √ |
| subtractVariable | √ |
| eliminateVariables | √ |
| normalizeConstraints | √ |
| collectVariables | √ |
| isConsistent | √ |

Table 134: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 135: Methods Concurrency Matrix

| | FractionElimination | eliminateVariableOverload | containsVariable | subtractVariable | eliminateVariables | normalizeConstraints | collectVariables | isConsistent |
|---|---|---|---|---|---|---|---|---|
| FractionElimination | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| eliminateVariableOverload | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| containsVariable | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| subtractVariable | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| eliminateVariables | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| normalizeConstraints | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ |
| collectVariables | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∥ | ∥ |
| isConsistent | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 46 FractionPair

Table 136: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| FractionPair | $\surd$ |
| getComponent1 | $\surd$ |
| getComponent2 | $\surd$ |
| hashCode | $\surd$ |
| equals | $\surd$ |

Table 137: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 138: Methods Concurrency Matrix

| | FractionPair | getComponent1 | getComponent2 | hashCode | equals |
|---|---|---|---|---|---|
| FractionPair | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getComponent1 | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| getComponent2 | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| hashCode | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| equals | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

## 47 Anonymous

Table 139: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| named | × |
| one | × |
| sum | × |
| var | × |
| zero | × |
| literal | × |
| impossible | × |
| relation | × |

Table 140: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 141: Methods Concurrency Matrix

| | Anonymous | named | one | sum | var | zero | literal | impossible | relation |
|---|---|---|---|---|---|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| named | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| one | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| sum | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| var | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| zero | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| literal | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| impossible | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |
| relation | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ | ∥ |

## 48 Anonymous

Table 142: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Anonymous | × |
| printStatus | √ |
| getInverse | √ |

Table 143: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 144: Methods Concurrency Matrix

| | Anonymous | printStatus | getInverse |
|---|---|---|---|
| Anonymous | ∦ | ∦ | ∦ |
| printStatus | ∦ | ∥ | ∥ |
| getInverse | ∦ | ∥ | ∥ |

## 49 SmtLibConstraintProcessor

Table 145: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| SmtLibConstraintProcessor | $\surd$ |
| impossible | $\surd$ |
| relation | $\surd$ |
| formatRelation | $\surd$ |
| literal | $\surd$ |
| sum | $\surd$ |
| named | $\surd$ |
| one | $\surd$ |
| var | $\surd$ |
| zero | $\surd$ |

Table 146: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 147: Methods Concurrency Matrix

| | SmtLibConstraintProcessor | impossible | relation | formatRelation | literal | sum | named | one | var | zero |
|---|---|---|---|---|---|---|---|---|---|---|
| SmtLibConstraintProcessor | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| impossible | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| relation | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| formatRelation | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| literal | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| sum | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| named | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| one | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |
| var | $\nparallel$ | $\parallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\parallel$ | $\nparallel$ | $\parallel$ |
| zero | $\nparallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ | $\parallel$ |

## 50 Impossible

Table 148: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Impossible | √ |
| getInstance | √ |
| equals | √ |
| hashCode | √ |

Table 149: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 150: Methods Concurrency Matrix

| | Impossible | getInstance | equals | hashCode |
|---|---|---|---|---|
| Impossible | ∦ | ∦ | ∦ | ∦ |
| getInstance | ∦ | ∥ | ∥ | ∥ |
| equals | ∦ | ∥ | ∥ | ∥ |
| hashCode | ∦ | ∥ | ∥ | ∥ |

## 51 Abbreviation

Table 151: Used Abbreviation

| Symbol | Meaning |
|---|---|
| $\surd$ | requires clause of the method is satisfiable |
| $\times$ | requires clause of the method is unsatisfiable |
| $\uparrow$ | The row-state can be transitioned to the column-state |
| $\times$ | The row-state cannot be transitioned to the column-state |
| $\parallel$ | The row-method can be possibly executed parallel with the column-method |
| $\nparallel$ | The row-method cannot be executed parallel with the column-method |

# 52 Annotated Version of Sequential Java Program generated by Sip4j

```
package outputs;
import edu.cmu.cs.plural.annot.*;

@ClassStates({@State(name = "alive")})
class ConsListTest {
@Perm(ensures="unique(this) in alive")
ConsListTest() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void testList() {

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class ConsList {
@Perm(ensures="unique(this) in alive")
ConsList() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  ConsList<T> removeElement(T t) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  ConsList<T> cons(T hd, ConsList<T> tl) {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
  ConsList<T> empty() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  ConsList<T> singleton(T hd) {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
  ConsList<T> list(T... ts) {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
  ConsList<T> concat(ConsList<T> front, ConsList<T> back) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  ConsList<T> removeElementOnce(T t) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  ConsList<O> map(Lambda<? super T,? extends O> lam) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  ConsList<T> filter(Lambda<? super T,? extends Boolean> lam) {
 return null;
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  O foldl(Lambda2<? super T,? super O,? extends O> lam, O o) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  ListIterator<T> listIterator(final int index) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 private ConsList<T> subListSameTail(int fromIndex) {
 return null;

}

  T get(int index) {
 return null;

}

  Iterator<T> iterator() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  ConsList<T> subList(int fromIndex, int toIndex) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  boolean contains(Object o) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  Object[] toArray() {
 return null;

}

  R impossible() {
 return null;

}

  void add(int index, T element) {

}

  boolean addAll(Collection<? extends T> c) {
 return 0;

}

  void clear() {

}

  T remove(int index) {
 return null;

}

  T removeOverload(int index) {
 return null;

}

  boolean removeAll(Collection<?> c) {
 return 0;

}
```

```
  boolean retainAll(Collection<?> c) {
 return 0;

}

  T set(int index, T element) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void main(String[] args) {

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Triple {
@Perm(ensures="unique(this) in alive")
Triple() {   }

@Perm(ensures="none(this) in alive")
 public F fst() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public S snd() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public T thrd() {
 return null;

}

  Triple<F,S,T> createTriple(F f, S s, T t) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Pair {
@Perm(ensures="unique(this) in alive")
Pair() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public A fst() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setComponent1(A component1) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public B snd() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setComponent2(B component2) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 protected Object clone() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

  Pair<A,B> create(A component1, B component2) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Empty {
@Perm(ensures="unique(this) in alive")
Empty() {   }


 public T hd() {
 return null;

}

 public int indexOf(Object o) {
 return 0;

}

 public boolean isEmpty() {
 return 0;

}

 public int lastIndexOf(Object o) {
 return 0;

}

 public int size() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public ConsList<T> tl() {
 return null;

}

 protected int indexOfHelper(int cur_index, Object o) {
 return 0;

}

 protected int lastIndexOfHelper(boolean found, int cur_index, int cur_last, Object o) {
 return 0;

}

 public String toString() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean containsAll(Collection<?> c) {
 return 0;
```

```
}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Utilities {
@Perm(ensures="unique(this) in alive")
Utilities() {   }


  String ASTNodeToString(ASTNode node) {
 return null;

}

  String ModifierToString(int modifier) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  MethodDeclaration getMethodDeclaration(ASTNode node) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  String methodDeclarationToString(MethodDeclaration md) {
 return null;

}

  T nyi() {
 return null;

}

  T nyiOverload(String err_msg) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void main(String[] args) {

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class CollectionMethods {
@Perm(ensures="unique(this) in alive")
CollectionMethods() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  List<O> map(List<? extends I> list, Mapping<I,O> fun) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  List<T> concat(List<? extends T> l1, List<? extends T> l2) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  Map<K,V> union(Map<? extends K,? extends V> m1, Map<? extends K,? extends V> m2) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void addToMultiMap(K key, V val, Map<K,List<V>> map) {

}
```

```
  Set<T> createSetWithoutElement(Set<T> s, T element) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  Set<T> mutableSet(T... elements) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Nonempty {
@Perm(ensures="unique(this) in alive")
Nonempty() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public T hd() {
 return null;

}

 protected int indexOfHelper(int cur_index, Object o) {
 return 0;

}

 public int indexOf(Object o) {
 return 0;

}

 public boolean isEmpty() {
 return 0;

}

 public int lastIndexOf(Object o) {
 return 0;

}

 protected int lastIndexOfHelper(boolean found, int cur_index, int cur_last, Object o) {
 return 0;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public int size() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public ConsList<T> tl() {
 return null;

}

 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

 public boolean containsAll(Collection<?> c) {
```

```
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }


 public void add(T e) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public boolean hasNext() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean hasPrevious() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T next() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int nextIndex() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public T previous() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int previousIndex() {
 return 0;

}

 public void remove() {

}

 public void set(T e) {

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class PluralParseError {
@Perm(ensures="unique(this) in alive")
PluralParseError() {   }


}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class ImpossibleConstraint {
@Perm(ensures="unique(this) in alive")
ImpossibleConstraint() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionConstraintVisitor<T> visitor) {
```

```
 return null;

}

 public String toString() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class ZeroFraction {
@Perm(ensures="unique(this) in alive")
ZeroFraction() {   }


 public boolean isZero() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionVisitor<T> visitor) {
 return null;

}

 public String toString() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class AbstractFractionTermVisitor {
@Perm(ensures="unique(this) in alive")
AbstractFractionTermVisitor() {   }


 public T named(NamedFraction fract) {
 return null;

}

 public T one(OneFraction fract) {
 return null;

}

 public T var(VariableFraction fract) {
 return null;

}

 public T zero(ZeroFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T literal(Fraction fract) {
 return null;

}

 public T sum(FractionSum fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class VariableFraction {
@Perm(ensures="unique(this) in alive")
VariableFraction() {   }
```

```
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public String getVarName() {
 return null;

}

 public boolean isVariable() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionVisitor<T> visitor) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public int compareToVar(VariableFraction other) {
 return 0;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionConstraints {
@Perm(ensures="unique(this) in alive")
FractionConstraints() {   }


  FractionConstraints createMutable() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public FractionConstraints addConstraint(FractionConstraint newConstraint) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean testConstraint(FractionConstraint test) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean isConsistent() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private boolean isConsistentInternal() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isImpossible() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public FractionAssignment simplify() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private FractionAssignment simplifyInternal() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Collection<FractionConstraint> getConstraints() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public FractionConstraints mutableCopy() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public FractionConstraints mutableCopyOverload(Set<NamedFraction> universals) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean testConstraints(FractionConstraint... test) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void addAll(FractionConstraints moreConstraints) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Set<VariableFraction> getVariables() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Set<NamedFraction> getConstants() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public Set<NamedFraction> getUniversalParameters() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void registerFractions(Set<Fraction> fractions) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean atLeastAsPrecise(FractionConstraints other) {
 return 0;

}
```

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public FractionConstraints freeze() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public FractionConstraints concat(FractionConstraints other) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean seemsConsistent() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public VariableFraction newVariableFraction() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public NamedFraction newNamedFraction() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean isKnown(Fraction f) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionConstraint {
@Perm(ensures="unique(this) in alive")
FractionConstraint() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  FractionConstraint impossible() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  FractionConstraint createEquality(FractionTerm... terms) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  FractionConstraint createLessThan(FractionTerm... terms) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  FractionConstraint createLessThanOrEqual(FractionTerm... terms) {
 return null;

}
```

```
}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionAssignment {
@Perm(ensures="unique(this) in alive")
FractionAssignment() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void resetChangedFlag() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void makeEquivalentOverload(Iterable<FractionTerm> terms) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void union(FractionTerm t1, FractionTerm t2) {

}

  SortedSet<FractionTerm> mutableSet(FractionTerm... initialElements) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean isZero(FractionTerm f) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Fraction getLiteral(FractionTerm f) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean isOne(FractionTerm f) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean areEquivalent(FractionTerm t1, FractionTerm t2) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean areEquivalentOverload(FractionTerm t1, FractionTerm t2, NamedFractionMapping mapping) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void makeZero(FractionTerm f) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void makeZeroOverload(List<FractionTerm> terms) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void makeOne(FractionTerm f) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void makeNonZero(FractionTerm t) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
```

```
 public boolean isNonZero(FractionTerm f) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  boolean isChanged() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean isConsistent() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  boolean sumsToConstant(FractionSum sum) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  boolean equivalentLiteralValues(FractionTerm t1, FractionTerm t2) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void makeEquivalent(FractionTerm... terms) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Fraction getConstant(FractionTerm f) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Fraction getRepresentative(Fraction f) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public String toString() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionRelation {
@Perm(ensures="unique(this) in alive")
FractionRelation() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Relop getRelop() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public List<FractionTerm> getTerms() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionConstraintVisitor<T> visitor) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
```

```
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public int compareTo(FractionRelation o) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Fraction {
@Perm(ensures="unique(this) in alive")
Fraction() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  Fraction zero() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  Fraction one() {
 return null;

}

 public boolean isZero() {
 return 0;

}

 public boolean isOne() {
 return 0;

}

 public boolean isVariable() {
 return 0;

}

 public boolean isNamed() {
 return 0;

}

  Fraction createNamed(String name) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  T dispatch(FractionTermVisitor<T> visitor) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  Fraction createExplicit(int p, int q) {
 return null;

}

  boolean isFixed() {
```

```
 return 0;

}

  boolean isNeitherZeroNorOne() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  boolean isGuaranteedGreaterThanZero() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  boolean isPossiblyGreaterOrEqual(final Fraction other) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionSum {
@Perm(ensures="unique(this) in alive")
FractionSum() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public List<Fraction> getSummands() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionTermVisitor<T> visitor) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class NamedFractionMapping {
@Perm(ensures="unique(this) in alive")
NamedFractionMapping() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean map(NamedFraction f1, NamedFraction f2) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class NamedFraction {
@Perm(ensures="unique(this) in alive")
```

```
NamedFraction() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getVarName() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  boolean isVariable(ASTNode node) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  boolean isJoinVariable() {
 return 0;

}

 public boolean isNamed() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionVisitor<T> visitor) {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class VariableElimination {
@Perm(ensures="unique(this) in alive")
VariableElimination() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Set<NormalizedFractionConstraint> eliminateVariables(Collection<FractionConstraint> constraints,
      FractionAssignment a) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public long getTimeout() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private Set<NormalizedFractionConstraint> normalizeConstraints(Collection<FractionConstraint>
     constraints) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private List<VariableFraction> eliminationOrder(Set<VariableFraction> vars) {
```

```
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 private Set<NormalizedFractionConstraint> addVariableConstraints(Set<NormalizedFractionConstraint> rels
      , Iterable<VariableFraction> vars) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private Set<NormalizedFractionConstraint> eliminateFraction(Set<NormalizedFractionConstraint> rels,
      Fraction x, boolean populateGroundRels) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private Set<NormalizedFractionConstraint> addConstConstraints(Set<NormalizedFractionConstraint> rels,
      Iterable<NamedFraction> vars) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean isConsistent() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean isSatisfiable(Set<NormalizedFractionConstraint> rels, Set<? extends Fraction> vars) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  boolean isPrimitiveConstraintSatisfiable(NormalizedFractionConstraint c) {
 return 0;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setTimeout(long timeout) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private SortedSet<T> collectVariables(Set<NormalizedFractionConstraint> rels, Class<T> variableType) {
 return null;

}

 private NormalizedFractionSum normalizeTerm(final FractionTerm term) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class NormalizedFractionConstraint {
@Perm(ensures="unique(this) in alive")
NormalizedFractionConstraint() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  NormalizedFractionConstraint createConstraintOverload(Fraction left, Relop relop, Fraction right) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Pair<NormalizedFractionSum,Boolean> isolateFraction(Fraction x) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
```

```
 public Relop getRelop() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  NormalizedFractionConstraint createConstraint(GeneralizedSum left, Relop relop, GeneralizedSum right)
      {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean isTrueWithAssumptions(Map<NamedFraction,NamedFraction> upperBounds) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isTriviallyTrue() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean dominates(NormalizedFractionConstraint other) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 private boolean isRangeConstraint() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean isPrimitive() {
 return 0;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public GeneralizedSum getRight() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public GeneralizedSum getLeft() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Rational {
@Perm(ensures="unique(this) in alive")
Rational() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  Rational one() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  Rational minusOne() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public boolean isZero() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Rational abs() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isPositive() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Rational negation() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  int gcd(int a, int b) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Rational div(Rational r) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Rational plus(Rational r) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  Rational zero() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Rational minus(Rational r) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isNegative() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isSmallerThan(Rational other) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Rational times(int i) {
 return null;
```

```
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Rational inverse() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public int getP() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getQ() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isOne() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class GeneralizedSum {
@Perm(ensures="unique(this) in alive")
GeneralizedSum() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Set<Fraction> getFractions() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Rational getCoefficient(Fraction f) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Rational getConstant() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isGround() {
 return 0;
```

```
}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class VariableRelativity {
@Perm(ensures="unique(this) in alive")
VariableRelativity() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean addRight(Relop relop, NormalizedFractionSum term) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean addLeft(NormalizedFractionSum term, Relop relop) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Set<NormalizedFractionConstraint> dumpRelations() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void dumpRelation(NormalizedFractionSum less, Relop relop, NormalizedFractionSum more) {

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class NormalizedFractionSum {
@Perm(ensures="unique(this) in alive")
NormalizedFractionSum() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  NormalizedFractionSum zero() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class SmtLibPrinter {
@Perm(ensures="unique(this) in alive")
SmtLibPrinter() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String toString(FractionConstraints constraints, Boolean satisfiable) {
 return null;

}
@Perm(ensures="none(this) in alive")
 public String printStatus() {
 return null;

}
```

```
@Perm(ensures="none(this) in alive")
 public SmtBenchmarkStatus getInverse() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class SmtLibBenchmarkPrinter {
@Perm(ensures="unique(this) in alive")
SmtLibBenchmarkPrinter() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void addLineComment(String commentLine) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void addStatus(SmtBenchmarkStatus status) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void addFormula(String formula) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void addUnknown(String name, String sort) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void addAssumption(Set<String> formulae) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void appendConjunction(Set<String> preds) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void addNegatedQuantifiedImplicationFormula(Set<String> assumptions, Set<String> exists, Set<
      String> conclusion) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void appendExists(Set<String> exists, Set<String> preds) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getResult() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean named(NamedFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean one(OneFraction fract) {
 return null;
```

```
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean var(VariableFraction fract) {
 return null;

}

 public Boolean zero(ZeroFraction fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class OneFraction {
@Perm(ensures="unique(this) in alive")
OneFraction() {   }


 public boolean isOne() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(FractionVisitor<T> visitor) {
 return null;

}

 public String toString() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionTerm {
@Perm(ensures="unique(this) in alive")
FractionTerm() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  FractionTerm createSum(Fraction... summands) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  FractionTerm createSumOverload(List<Fraction> summands) {
 return null;

}

 public int compareTo(final FractionTerm o) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Integer literal(Fraction fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Integer sum(FractionSum fract) {
```

```
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean named(NamedFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean one(OneFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean var(VariableFraction fract) {
 return null;

}

 public Boolean zero(ZeroFraction fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean one(OneFraction fract) {
 return null;

}

 public Boolean zero(ZeroFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean named(NamedFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean var(VariableFraction fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Relop {
@Perm(ensures="unique(this) in alive")
Relop() {   }

@Perm(ensures="none(this) in alive")
 public String toString() {
 return null;

}
```

```
}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean named(NamedFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean one(OneFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean var(VariableFraction fract) {
 return null;

}

 public Boolean zero(ZeroFraction fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class RelationFractionPair {
@Perm(ensures="unique(this) in alive")
RelationFractionPair() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  RelationFractionPair createEqual(NormalizedFractionTerm c1, NormalizedFractionTerm c2) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  RelationFractionPair createLeq(NormalizedFractionTerm c1, NormalizedFractionTerm c2) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  RelationFractionPair createLess(NormalizedFractionTerm c1, NormalizedFractionTerm c2) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Relop getRelop() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;
```

```
}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private NormalizedFractionConstraint createRelation(NormalizedFractionSum left, Relop relop,
      NormalizedFractionSum right) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int compare(VariableFraction o1, VariableFraction o2) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private Fraction getRepresentative(Fraction fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private Fraction[] getRepresentatives(List<Fraction> summands) {
 return null;

}

 public Boolean impossible(ImpossibleConstraint fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Boolean relation(FractionRelation fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Integer literal(Fraction fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Integer sum(FractionSum fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class SimpleFractionSum {
@Perm(ensures="unique(this) in alive")
SimpleFractionSum() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  SimpleFractionSum createAdd(Fraction c1, Fraction c2) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  SimpleFractionSum createSub(Fraction c1, Fraction c2) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Sumop getSumop() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String toString() {
 return null;

}

 public T dispatch(NormalizedFractionVisitor<T> visitor) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Sumop {
@Perm(ensures="unique(this) in alive")
Sumop() {   }

@Perm(ensures="none(this) in alive")
 public String toString() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class SimpleVariableRelativity {
@Perm(ensures="unique(this) in alive")
SimpleVariableRelativity() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean addRight(Relop relop, NormalizedFractionTerm term) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean addLeft(NormalizedFractionTerm term, Relop relop) {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Set<RelationFractionPair> dumpRelations(Set<RelationFractionPair> result) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean named(NamedFraction fract) {
```

```
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean one(OneFraction fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Integer sum(FractionSum fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean var(VariableFraction fract) {
 return null;

}

 public Boolean zero(ZeroFraction fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class FractionElimination {
@Perm(ensures="unique(this) in alive")
FractionElimination() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private Set<RelationFractionPair> eliminateVariableOverload(Set<RelationFractionPair> rels,
      VariableFraction x) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  int containsVariable(final NormalizedFractionTerm t, final VariableFraction x) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  RelationFractionPair subtractVariable(final RelationFractionPair rel, final VariableFraction x, final
       int sign) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Set<RelationFractionPair> eliminateVariables(Set<FractionConstraint> constraints) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private Set<RelationFractionPair> normalizeConstraints(Set<FractionConstraint> constraints) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 private Set<VariableFraction> collectVariables(Set<RelationFractionPair> rels) {
 return null;

}

 public boolean isConsistent() {
 return 0;

}

}ENDOFCLASS
```

```
@ClassStates({@State(name = "alive")})

class FractionPair {
@Perm(ensures="unique(this) in alive")
FractionPair() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public T getComponent1() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public T getComponent2() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public int hashCode() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean named(NamedFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean one(OneFraction fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Integer sum(FractionSum fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Boolean var(VariableFraction fract) {
 return null;

}

 public Boolean zero(ZeroFraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Integer literal(Fraction fract) {
 return null;

}

 public Boolean impossible(ImpossibleConstraint fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
```

```
 public Boolean relation(FractionRelation fract) {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Anonymous {
@Perm(ensures="unique(this) in alive")
Anonymous() {   }


 public String printStatus() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public SmtBenchmarkStatus getInverse() {
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class SmtLibConstraintProcessor {
@Perm(ensures="unique(this) in alive")
SmtLibConstraintProcessor() {   }


 public Boolean impossible(ImpossibleConstraint fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Boolean relation(FractionRelation fract) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 private String formatRelation(String term1, Relop relop, String term2) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Pair<String,Boolean> literal(Fraction fract) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Pair<String,Boolean> sum(FractionSum fract) {
 return null;

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public Pair<String,Boolean> named(NamedFraction fract) {
 return null;

}

 public Pair<String,Boolean> one(OneFraction fract) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Pair<String,Boolean> var(VariableFraction fract) {
 return null;

}

 public Pair<String,Boolean> zero(ZeroFraction fract) {
```

```
 return null;

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class Impossible {
@Perm(ensures="unique(this) in alive")
Impossible() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  Impossible getInstance() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public boolean equals(Object obj) {
 return 0;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public int hashCode() {
 return 0;

}

}ENDOFCLASS
```