Constraint Logic Programming over Infinite Domains with an Application to Proof

Sebastian Krings and Michael Leuschel

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
Germany

Background

- Model Checking
- Data Validation
- Proof and Disproof

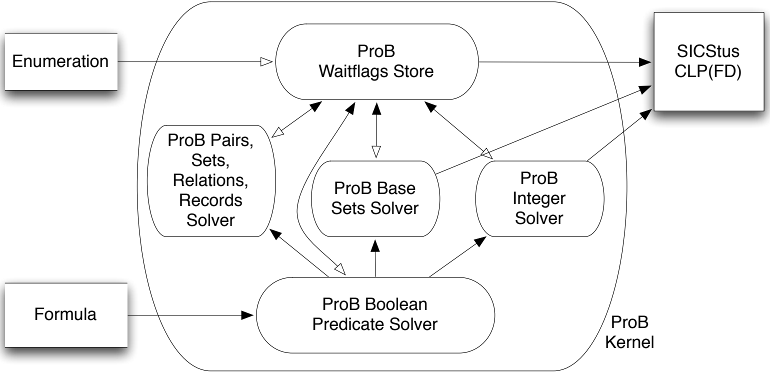
B Example

```
MACHINE BinarySearch
CONSTANTS arr, sze, goal
PROPERTIES
    sze : NAT & arr : 1..sze --> INT &
    !(i).(i:1..(sze-1) => arr(i) <= arr(i+1)) &
    goal : INT &
    arr = [-1,0,1,2,3,5] & goal : {5,4,2,6}
VARIABLES i,j,found
INVARIANT
    found:BOOL & i:NAT & j:NAT &
    (found=TRUE => arr(i)=goal) &
INITIALISATION found,i,j := FALSE,1,sze
OPERATIONS
    Step = SELECT found=FALSE & i<j THEN
        ...
```

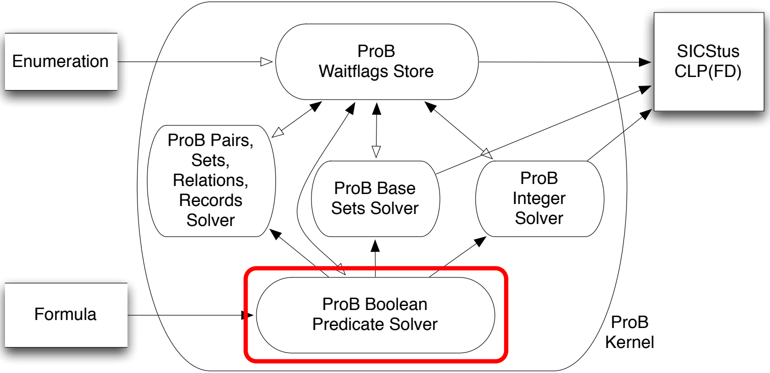
PROB Kernel

- Based on SICStus Prolog
  - 150k LoC
  - + Extensions in Tcl/Tk, Java, C, ...
- CLP(FD) + CHR
- Custom solvers for high-level structures
  - Sets, lists, tuples, records, ...
  - Nested quantification
  - Comprehensions & lambdas

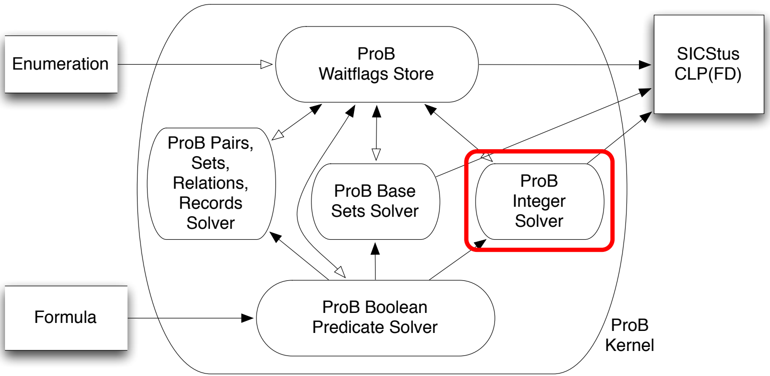
PROB Kernel


Enumeration
ProB
Waitflags Store
SICStus
CLP(FD)
ProB Pairs,
Sets,
Relations,
Records
Solver
ProB Base
Sets Solver
ProB
Integer
Solver
Formula
ProB Boolean
Predicate Solver
ProB
Kernel

PROB Kernel


Enumeration
ProB
Waitflags Store
SICStus
CLP(FD)
ProB Pairs,
Sets,
Relations,
Records
Solver
ProB Base
Sets Solver
ProB
Integer
Solver
Formula
ProB Boolean
Predicate Solver
ProB
Kernel

PROB Kernel


Enumeration
ProB
Waitflags Store
SICStus
CLP(FD)
ProB Pairs,
Sets,
Relations,
Records
Solver
ProB Base
Sets Solver
ProB
Integer
Solver
Formula
ProB Boolean
Predicate Solver
ProB
Kernel

Techniques for Large or Infinite Domains

- Tracking of enumeration scopes
- Randomized enumeration
- High-level reasoning

Types of Enumeration

Distinguish based on effect on overall result:

- No enumeration, result not influenced
  - e.g., no valuation $\Rightarrow$ formula unsatisfiable
- Exhaustive enumeration
- Non-exhaustive enumeration with or w/o result
  - $\Rightarrow$ consider scope

Nested Enumeration Scopes

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

$\exists x.(x \in [-10, 10] \wedge \forall y.y \in [-15, 5] \Rightarrow y \leq x)$

existential scope

x can be enumerated exhaustively but is not

> $\forall y.y \in [-15, 5] \Rightarrow y \leq x$
>
> universal scope
>
> y is enumerated exhaustively
>
> > $y \leq x$
> >
> > CLP(FD) constraint

Nested Enumeration Scopes

$\exists x.(\neg \forall y.y \in [-15, 5] \Rightarrow y \leq x)$

existential scope

non-exhaustively search for x

$\forall y.y \in [-15, 5] \Rightarrow y \leq x$

existential scope

y can be enumerated exhaustively but is not

$y \leq x$

CLP(FD) constraint

Examples

- `x*x=10000`, solution found w/o enumeration
- `{x|x*x=10000}`, (all) solutions found w/o enumeration
- `x>10000 & x mod 1234 = 1`, solution found despite enumeration
- `{x|x>10000 & x mod 1234 = 1}` can thus not be solved
- `x*x = 10001` unsat by CLP(FD) propagation w/o enumeration
- `x>10000 & x mod 1234 = 1 & x*x = 10*x` partial enumeration, no result

Randomized Enumeration

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Try values of different size / characteristics
- Less likely to get stuck in some part of the search space
- Increase variance, e.g., for test case generation

Randomized Enumeration - Challenges

- Random permutation
  - $\Rightarrow$ avoid duplicates
- Detect exhaustive traversal

Fisher-Yates / Knuth Shuffle!

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Data: List a
Result: Random permutation of a
procedure shuffle(a)
  for $i \in [0, length(a) - 1]$ **do**
    chose j randomly such that $0 \leq j \leq i$
    if $j \neq i$ **then**
      $perm[i] := perm[j]$
    end
    $perm[j] := a[i]$
  end
  return $perm$

Fisher-Yates / Knuth Shuffle :(

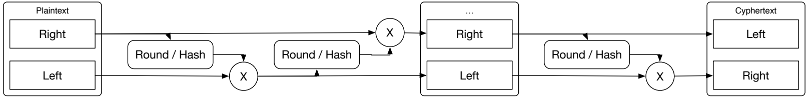
HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Data: List *a*
Result: Random permutation of *a*
procedure shuffle(*a*)
  for $i \in [0, length(a) - 1]$ **do**
    chose *j* randomly such that $0 \leq j \leq i$
    if $j \neq i$ **then**
      $perm[i] := perm[j]$
    end
    $perm[j] := a[i]$
  end
  return *perm*

Randomized Enumeration - Solution

- Use techniques from cryptography!
- "Encrypt" $[l, u] \rightarrow [l, u]$
- Using random encryption key
  - $\Rightarrow$ random enumeration of $[l, u]$
- Unique encryption
  - $\Rightarrow$ no duplicates, random permutation

Feistel Network



High-Level Reasoning using CHR

- CLP(FD) weak without bounds
- e.g. no unsat for $X < Y \wedge Y < X$
- High-level reasoning:
  - Infer falsity
  - Find new CLP(FD) constraints

CHR Rules (excerpt)

```
reflexivity  @ leq(X,X) <=> true.
antisymmetry @ leq(X,Y), leq(Y,X) <=> X = Y.
idempotence  @ leq(X,Y) \ leq(X,Y) <=> true.
transitivity @ leq(X,Y), leq(Y,Z) ==> leq(X,Z).

antireflexivity @ lt(X,X) <=> fail.
idempotence     @ lt(X,Y) \ lt(X,Y) <=> true.
transitivity    @ lt(X,Y), leq(Y,Z) ==> lt(X,Z).
transitivity    @ leq(X,Y), lt(Y,Z) ==> lt(X,Z).
transitivity    @ lt(X,Y), lt(Y,Z)  ==> lt(X,Z).
```

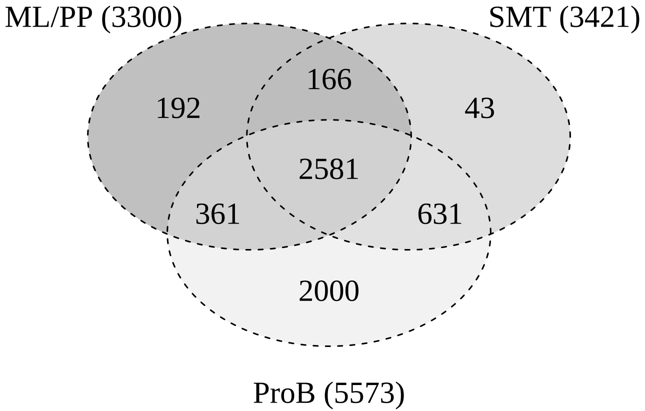
Comparison

predicate	with CHR	without CHR
$x > 3$	1.2	1.08
$x > y \wedge y > x$	0.92	timeout
$x = 3 \wedge x > y \wedge y = 4$	0.98	1.07
$x = 3 \wedge x > y$	0.86	1.0
$x = 3 \wedge x < y$	1.1	1.11
$w > x \wedge x > y \wedge y > z \wedge w = 1 \wedge z = 1$	0.98	0.95
$w > x \wedge x > y \wedge y > z \wedge z > w$	0.88	timeout
$x + 2 > y + 1 \wedge y > x$	timeout	timeout
$x > y \wedge y > x + 1$	0.9	timeout

Application - Proof

- Given Event-B sequent $H_i(x_1, \ldots, x_k) \models G(x_1, \ldots, x_k)$
- Build $\exists x_1, \ldots, x_k : (H_1(x_1, \ldots, x_k) \wedge \ldots \wedge H_n(x_1, \ldots, x_k)) \Rightarrow \neg G(x_1, \ldots, x_k)$
- Search for counter-example
- Exhaustive search $\Rightarrow$ proof

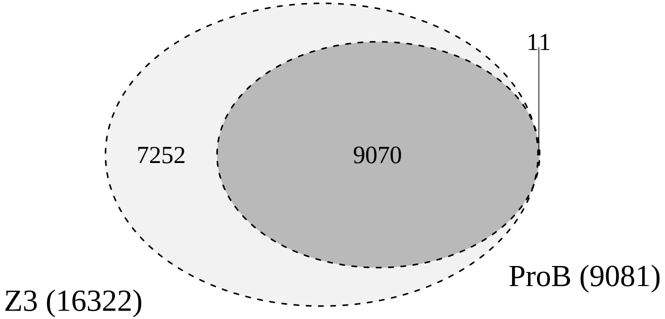
Results



Application - SMT Solving

- SMT-LIB standard fully supported
- Translation from SMT-LIB to B
- Competitive on integer arithmetic
- Not competitive for bitvectors, arrays and sets

Results on Integer Arithmetic



SMT-COMP 2016 - NIA - Main Track

Winners:

Sequential Performances	Parallel Performances
ProB	ProB

Result table[1]

Solver	Sequential performance		
	Error Score	Correctly Solved Score	avg. CPU time
CVC4	0.000	5.000	0.019
ProB	0.000	8.000	0.888
vampire_smt_4.1	0.000	6.000	335.043
vampire_smt_4.1_parallel	0.000	7.000	537.677
z3[n]	0.000	9.000	0.038

Thank you for your attention!
Are there any questions left?

krings@cs.uni-duesseldorf.de
www.stups.uni-duesseldorf.de