A comparison of boolean and rational solvers for interpreted automata verification

Synchronous workshop 2005

Christophe Mauras
Christophe.Mauras@univ-nantes.fr

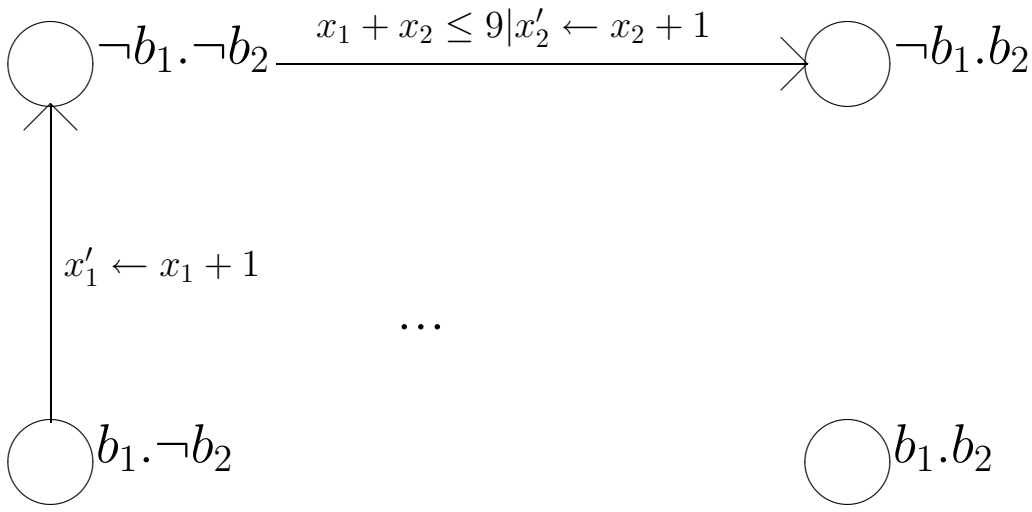
LINA/Université de Nantes

Overview

- Interpreted Automata
- Binary Decision Diagrams / CLP(B)
- Polyhedra / CLP(Q)
- Benchmarks
- State space search in CLP(B), CLP(Q)
- Computing all solutions, metaprogramming and dynamic constraints.
- Experimental Results
- Conclusion

Interpreted automata

Infinite state systems with boolean state variables and integer state variables. Transitions may be guarded by linear constraints.



For verification, we need to compute forward or backward reachable states. We focuses on coding and computing set of states.

Representation of interpreted automata

Initial States may be described by their characteristic function. Transitions may by described by relations about states (integer and boolean variables) and next states (primed variables).

```
initial
    not b1 and not b2 and x1>=0 and x2>=0
    and x1<=5 and x2<=5
relation
  ((b1' and x1'=x1+1) or (not b1' and x1'=x1-1))and
  ((b2' and x2'=x2+1) or (not b2' and x2'=x2-1))and
  ((x1+x2>=10) => (not b1' and not b2')) and
  (not b1' or not b2')
```

CLP(B) and Binary Decision Diagrams

The clp(B) system provided by this library module is an instance of the general Constraint Logic Programming scheme introduced in [Jaffar and Michaylov 87]. It is a solver for constraints over the Boolean domain, i.e. the values 0 and 1. This domain is particularly useful for modeling digital circuits, and the constraint solver can be used for verification, design, optimization etc. of such circuits.

8 bits adder with CLP(B)

```
onebitadder(Ain,Bin,Cin,Xout,Cout) :-
     sat(Xout =:= card([1,3],[Ain,Bin,Cin])),
     sat(Cout =:= card([2-3],[Ain,Bin,Cin])).

bits8adder([A0,A1,A2,A3,A4,A5,A6,A7],
           [B0,B1,B2,B3,B4,B5,B6,B7],
           [X0,X1,X2,X3,X4,X5,X6,X7],C0,C8) :-
            onebitadder(A0,B0,C0,X0,C1),
            onebitadder(A1,B1,C1,X1,C2),
            ...
            onebitadder(A7,B7,C7,X7,C8).
```

8bits adder with BDD

```
onebitadder(Ain,Bin,Cin,Xout,Cout) :- bdd_xor(Ain,Bin,X1),
                bdd_xor(X1,Cin,Xout),  bdd_and(Ain,Bin,X2),
                bdd_and(Ain,Cin,X3),   bdd_and(Bin,Cin,X4),
                bdd_or(X2,X3,X5),      bdd_or(X4,X5,Cout).

bits8adder([A0,A1,A2,A3,A4,A5,A6,A7],
           [B0,B1,B2,B3,B4,B5,B6,B7],
           [X0,X1,X2,X3,X4,X5,X6,X7],C8) :-
            bdd_false(C0),
            bdd_ithvar(0,A0),bdd_ithvar(1,B0),
            bdd_ithvar(2,A1),bdd_ithvar(3,B1),
            ...
            bdd_ithvar(14,A7),bdd_ithvar(15,B7),
            onebitadder(A0,B0,C0,X0,C1),
            onebitadder(A1,B1,C1,X1,C2),
            ...
            onebitadder(A7,B7,C7,X7,C8).
```

Experimental results

CLP(B) versus BDD

n	Time (s)	Space
4	0	
5	1	
6	3	
7	9	
8	37	
9	148	
10		Out of memory

n	Time (s)	Nb Nodes
8	0.0	739
16	0.0	2231
32	0.0	8095
64	0.1	31423
128	0.2	124231
256	3.0	494215
512	48.6	1971463

Relational / Functional

CLP(B)

```
bits8adder1([a0,a1,a2,a3,a4,a5,a6,a7],
            [b0,b1,b2,b3,b4,b5,b6,b7],
             X,0,Cout).

bits8adder1([A0,A1,A2,A3,A4,A5,A6,A7],
            [B0,B1,B2,B3,B4,B5,B6,B7],
            [0,0,0,0,0,0,0,0],1,1).
```

BDD

Output are functions from inputs. To compute with relations, add the following :

```
bdd_ithvar(14,Cout), bdd_biimp(Cout,C8, R)
```

CLP(Q) and Polyhedra

CLP(Q)

Reference :

– Holzbaur C. : OFAI clp(q,r) Manual, Edition 1.3.3, Austrian Research Institute for Artificial Intelligence, Vienna, TR-95-09, 1995.

Polyhedra

References :

– Le Verge H. A note on Chernikova's algorithm, Irisa 1992.

– Wilde D. A library for doing polyhedral operations, Irisa 1993.

– Loechner V. A library for manipulation parametrized polyhedra, Univ. Strasbourg, 1999.

Constraints Enumeration/Polyhedra Union

A union of polyhedra is a set of points that belong to one of polyhedra : Implicit computation in CLP(Q).

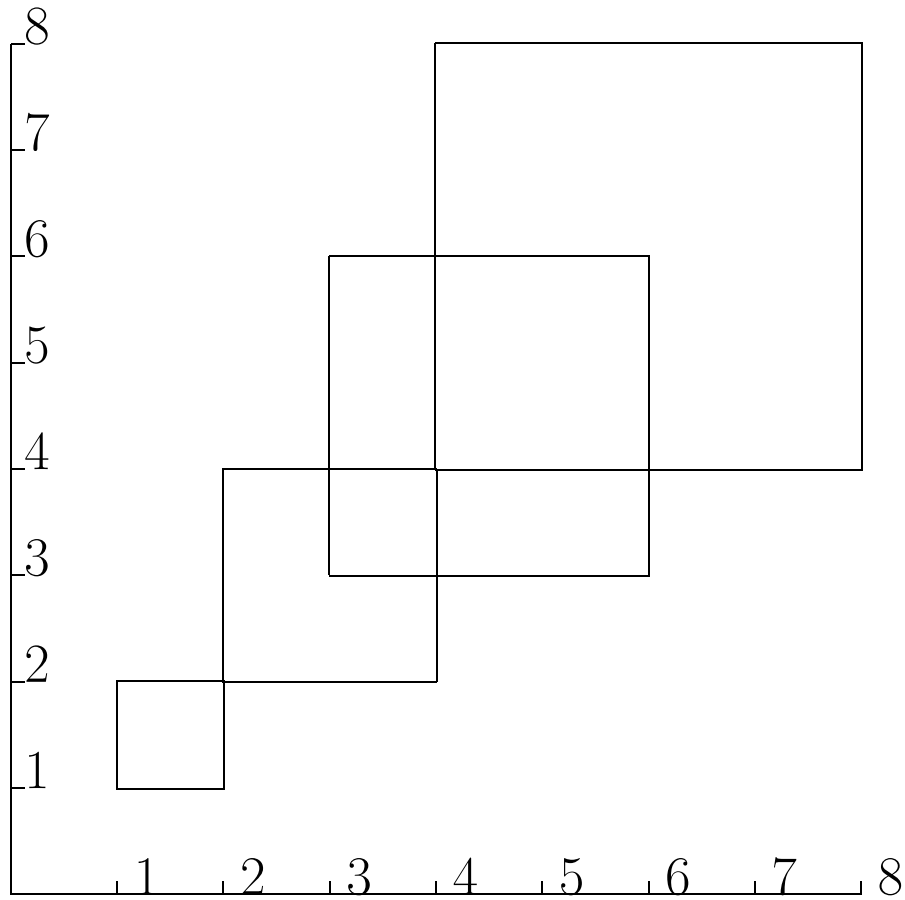
```
p(X,Y,N) :- {X>=N, X=<2*N, Y>=N, Y=<2*N}.

u(N,X,Y) :- {N>0,N1=N - 1},u(N1,X,Y).
u(N,X,Y) :- p(X,Y,N).
```

Explicit computation with PolyLib :

```
p(P,N) :- A is -N, B is 2*N,
          C=[[1,1,0,A],[1,-1,0,B],[1,0,1,A],[1,0,-1,B]],
          poly_of_constraints(C,4,4,P).
u(P,N):- {N=0}, p(P,0).
u(U,N):-{N>0},N1 is N-1,p(P,N),u(U1,N1),poly_union(U1,P,U)
```

Polyhedra Union



Execution and Constraints model

Prolog execution provides 2 ways to enumerate solutions :
– time enumeration : `u(3,X,Y).`
– space enumeration : `setof((X,Y), u(3,X,Y),R).`

CLP(Q) constraints model :
– the set of linear constraints on Q is a constraint domain closed under conjunction, existential quantification, and has a empty test.

Entailment/Inclusion : Execution Time

$$\forall i \leq N, \{x \geq i, x \leq 2*i, y \geq i, y \leq 2*i\} \subset \{2*x \geq y, 2*y \geq x\}$$

N	CLP(Q) (s)	PolyLib (s)
1000	1.03	2.04
2000	2.07	4.11
3000	3.08	6.15
4000	4.12	8.20
5000	5.13	10.26
6000	6.13	12.26
7000	7.18	14.33
8000	8.21	16.38
9000	9.22	18.43
10000	10.24	20.49

Convex Hull

```
% Holzbaur C.
conv_hull(Points,Xs):- lin_comb(Points,Lambdas,Zero,Xs),
                       zero(Zero),polytope(Lambdas).

polytope(Xs) :- positive_sum(Xs, 1).
...
```

N	CLP(Q) (s)	N	PolyLib (s)
10	0.10	100	0.47
20	0.34	200	1.07
30	0.73	300	1.90
40	1.28	400	2.84
50	1.97	500	4.13

Benchmarks

```
initial
    not b1 and not b2 ... and not bn
    and x1>=0 and x2>=0 ... and xn>=0
    and x1<=5 and x2<=5 ... and xn<=0
relation
  ((b1' and x1'=x1+1) or (not b1' and x1'=x1-1))and
  ((b2' and x2'=x2+1) or (not b2' and x2'=x2-1))and
  ...
  ((bn' and xn'=xn+1) or (not bn'and xn'=xn-1))and
  ((x1+x2+...+xn>=10) => (not b1'and not b2'...))
  and (not b1' or not b2'... or not bn')
```

State space search with CLP(B) and CLP(Q)

Direct translation of transition relation in clauses using disjunctive normal form. Execution gives an enumeration of reachable states :

```
| ?- reach(B1,B2,X1,X2,T).
T = 0,B1 = 0,B2 = 0,{X1=<5},{X2=<5},{X1>=0},{X2>=0} ? ;
T = 1,B1 = 1,B2 = 0,{X1=<6},{X2=<4},{X1+X2=<9},{X1>=1},{X2>= -1} ? ;
T = 1,B1 = 0,B2 = 1,{X1=<4},{X2=<6},{X1+X2=<9},{X1>= -1},{X2>=1} ? ;
T = 1,B1 = 0,B2 = 0,{X1=<4},{X2=<4},{X1>= -1},{X2>= -1} ? ;
T = 2,B1 = 1,B2 = 0,{X1=<7},{X2=<3},{X1+X2=<9},{X1>=2},{X2>= -2} ? ;
T = 2,B1 = 0,B2 = 1,{X1=<5},{X2=<5},{X1+X2=<9},{X1>=0},{X2>=0} ? ;
T = 2,B1 = 0,B2 = 0,{X1=<5},{X2=<3},{X1+X2=<7},{X1>=0},{X2>= -2} ? ;
T = 2,B1 = 1,B2 = 0,{X1=<5},{X2=<5},{X1+X2=<9},{X1>=0},{X2>=0} ? ;
T = 2,B1 = 0,B2 = 1,{X1=<3},{X2=<7},{X1+X2=<9},{X1>= -2},{X2>=2} ? ;
T = 2,B1 = 0,B2 = 0,{X1=<3},{X2=<5},{X1+X2=<7},{X1>= -2},{X2>=0} ? ;
T = 2,B1 = 1,B2 = 0,{X1=<5},{X2=<3},{X1>=0},{X2>= -2} ? ;
T = 2,B1 = 0,B2 = 1,{X1=<3},{X2=<5},{X1>= -2},{X2>=0} ? ;
T = 2,B1 = 0,B2 = 0,{X1=<3},{X2=<3},{X1>= -2},{X2>= -2} ? ;
```

Deductive Model Checking [Podelski 1999]

DMC uses :

- meta-programming to extract constraints from internal CLP database. Predicates : `dump` and `call_residue`
- new operations (as widening) implemented on constraints as lists
- dynamic constraints to register states as "constrained facts" in prolog database.

Using DMC as forward checker

Translating interpreted automata in constrained facts allow to compute with DMC reachable states :

```
s(0, p(s_s,A,B,C,D), {D-5=<0,C-5=<0,D>=0,C>=0,B=0,A=0}, 1, (0,0)).
s(1, p(s_s,1.0,0.0,A,B), {A=<6.0,B=<4.0,A+B=<9.0,A>=1.0,B>= -1.0}, 2, (1,1)).
s(1, p(s_s,0.0,1.0,A,B), {A=<4.0,B=<6.0,A+B=<9.0,A>= -1.0,B>=1.0}, 3, (1,1)).
s(1, p(s_s,0.0,0.0,A,B), {A=<4.0,B=<4.0,A>= -1.0,B>= -1.0}, 4, (1,1)).
s(2, p(s_s,1.0,0.0,A,B), {A=<7.0,B=<3.0,A+B=<9.0,A>=2.0,B>= -2.0}, 5, (1,2)).
s(2, p(s_s,1.0,0.0,A,B), {A=<5.0,B=<5.0,A+B=<9.0,A>=0.0,B>=-0.0}, 6, (1,3)).
s(2, p(s_s,1.0,0.0,A,B), {A=<5.0,B=<3.0,A>=-0.0,B>= -2.0}, 7, (1,4)).
s(2, p(s_s,0.0,1.0,A,B), {A=<5.0,B=<5.0,A+B=<9.0,A>=0.0,B>=-0.0}, 8, (1,2)).
s(2, p(s_s,0.0,1.0,A,B), {A=<3.0,B=<7.0,A+B=<9.0,A>= -2.0,B>=2.0}, 9, (1,3)).
s(2, p(s_s,0.0,1.0,A,B), {A=<3.0,B=<5.0,A>= -2.0,B>=-0.0}, 10, (1,4)).
s(2, p(s_s,0.0,0.0,A,B), {A=<5.0,B=<3.0,A+B=<7.0,A>=0.0,B>= -2.0}, 11, (1,2)).
s(2, p(s_s,0.0,0.0,A,B), {A=<3.0,B=<5.0,A+B=<7.0,A>= -2.0,B>=-0.0}, 12, (1,3)).
s(2, p(s_s,0.0,0.0,A,B), {A=<3.0,B=<3.0,A>= -2.0,B>= -2.0}, 13, (1,4)).
...
```

Symbolic simulation

Using binary decision diagrams with linear constraints, interpreted automata may be simulated symbolically [mauras 96].

Results for previous example :

```
Time 0 : not b1 and not b2 and x1<=5 and x1>=0 and x2<=5 and x2>=0
Time 1 : if b1
then not b2 and x1+x2<=9 and x1<=6 and x1>=1 and x2<=4 and 0<=x2+1
else if b2
     then x1+x2<=9 and x1<=4 and 0<=x1+1 and x2<=6 and x2>=1
     else x1<=4 and 0<=x1+1 and x2<=4 and 0<=x2+1
Time 2 : if b1
then not b2 and x1+x2<=9 and x1<=7 and x1>=0 and x2<=5 and 0<=x2+2
else if b2
     then x1+x2<=9 and x1<=5 and 0<=x1+2 and x2<=7 and x2>=0
     else x1+x2<=7 and x1<=5 and 0<=x1+2 and x2<=5 and 0<=x2+2
```

Conclusion

For synchronous programming and verification :

– CLP(Q) is better than PolyLib for entailment test and empty test, (Fourier-Motzkin Elimination/Chernikova)

– Expressivity of CLP description allows programmation, verification and test with same description.

For constraint logic programming

– Need to improve BDD efficiency in CLP(B)

– Encoding booleans as numbers is not efficient

– Mixing binary decision diagrams with constraints may provide a new domain for CLP.