

Constraints in analysis of synchronous programs

Synchron' 2006

Christophe Maudas
Christophe.Maudas@univ-nantes.fr

LINA/Université de Nantes

Overview

- References : some works that had influenced mine.
- Experiments : those that work and others
- Tutorial : how to exploit constraints, to reformulate known techniques in analysis of synchronous programs.

References

- 20 years ago :
 - W. Buttner and H. Simonis, Embedding boolean expressions into logic programming
 - M.M. Corsini and A. Rauzy, Symbolic model-checking and constraint logic programming : a cross-fertilisation
- 10 years ago :
 - G. Delzanno and A. Podelski, Model Checking in CLP
 - W. Chan and R. Anderson and P. Beame and D. Notkin, Combining constraints solving and symbolic model-checking for a class of systems with non-linear constraints
 - F. Benhamou and W. Older, Applying interval arithmetic to real, integer and boolean constraints

References

- Recent works : 1st workshop on Constraints in Software Testing, Verification and Analysis (CSTVA'06)
 - The V3F Project, B.Blanc, Fabrice Bouquet, Arnaud Gottlieb, Bertrand Jeannet, Thierry Jeron, Bruno Legeard, Bruno Marre, Claude Michel, Michel Rueher
 - Generating random values using Binary Decision Diagrams and Convex Polyhedra Erwan Jahier, Pascal Raymond
 - ...

Experiments

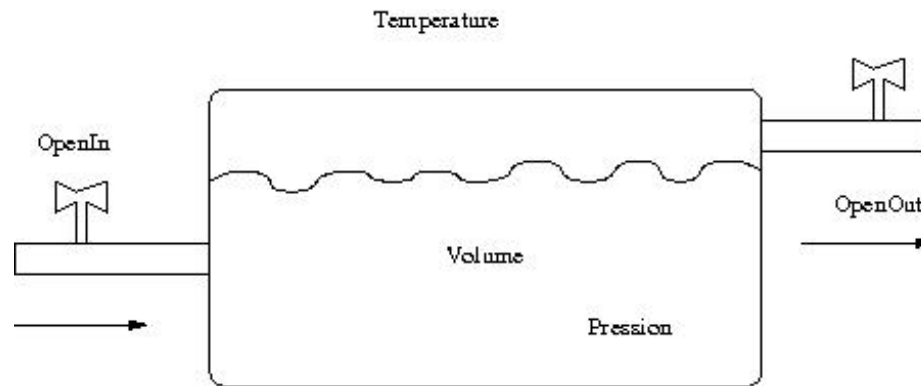
- Symbolic execution on interpreted automata (YoYo, 1996)
 - Simulation, verification, invariant synthesis
 - Description language : very poor
 - Execution model : only forward, no fixpoint
- Timed Default Concurrent Constraints
 - A language unifying concepts of synchronous languages
 - A prototype interpreter restricted to closed programs
 - A tentative to patch it for verification purposes (aborted)
- Deductive Model Checking
 - A method to compute set of states from a transition system encoded as constrained facts
 - A tentative to re-use it for TDCC (aborted)

Tutorial

5 Good reasons to use constraints :

- Ability to specify and execute incomplete designs
- Unicity of formalism for system under test, its environment and properties to be verified
- Bidirectionnality : pre and post are computed in a similar way
- Ability to represent set of states (characteristic function) or set of trajectories
- Availability of many solvers.

A Controller for a steam boiler



Computing Post predicate transformer as constraint solving

`post(In, Out, P, V, Nin, Nout, Np, Nv) .`

`In = 0, Nin = 0, Out = 0, Nout = 1, {P=Np}, {Nv=V}, {V=<0}, {Np>=6} ? ;`
`In = 0, Nin = 0, Out = 0, Nout = 1, {P= -1+Np}, {V=5+Nv}, {Nv=<19}, {Np>=6}, {Nv>= -4} ? ;`
`In = 1, Nin = 0, Out = 0, Nout = 1, {P=Np}, {Nv=V}, {V=<0}, {Np>=6} ? ;`
`In = 1, Nin = 0, Out = 1, Nout = 1, {P=Np}, {Nv=V}, {V=<0}, {Np>=6} ? ;`
`In = 1, Nin = 0, Out = 0, Nout = 1, {P= -1+Np}, {V=5+Nv}, {Nv=<19}, {Nv>= -4}, {Np>=6} ? ;`
`In = 1, Nin = 0, Out = 1, Nout = 1, {P= -1+Np}, {V=5+Nv}, {Nv=<19}, {Nv>= -4}, {Np>=6} ? ;`
`In = 1, Nin = 1, Out = 1, Nout = 1, {P=Np}, {Nv=V}, {Np}<5}, {V=<0} ?`
`...`

Computing image of a defined state

```
post(0,1,7,10,Nopenin,Nopenout,Npressure,Nvolume) .
```

```
Nopenin = 0,Nvolume = 5,Nopenout = 1,Npressure = 8 ? ;
```

```
Nopenin = 0,Nvolume = 15,Nopenout = 1,Npressure = 6 ? ;
```

Useful for simulation

Computing image of a set of states

$\{P \geq 5\}, \{V < 12\},$
 $\text{post}(1, 0, P, V, \text{Nopenin}, \text{Nopenout}, \text{Npressure}, \text{Nvolume}).$

$\text{Nopenin} = 0, \text{Nopenout} = 1, \{\text{Nvolume} = V\}, \{P = \text{Npressure}\}, \{V < 0\}, \{\text{Npressure} \geq 6\} ? ;$

$\text{Nopenin} = 0, \text{Nopenout} = 1, \{\text{Nvolume} = -5 + V\}, \{P = -1 + \text{Npressure}\}, \{V < 12\}, \{\text{Npressure} \geq 6\}, \{V \geq 1\} ? ;$

$\text{Nopenin} = 1, \text{Nopenout} = 0, \{\text{Npressure} = -1 + P\}, \{\text{Nvolume} = 5 + V\}, \{V < 12\}, \{P \geq 5\} ? ;$

$P = 5, \text{Nopenin} = 1, \text{Nopenout} = 0, \text{Npressure} = 5, \{\text{Nvolume} = V\}, \{V < 0\} ? ;$

Useful for symbolic execution/observability in structural testing

Computing pre-image of a (set of) state(s)

```
post(Openin,Openout,Pressure,Volume,_,_,5,20).
```

```
Openin = 0,Volume = 15,Openout = 0,Pressure = 6 ? ;  
Openin = 0,Volume = 15,Openout = 1,Pressure = 6 ? ;  
Openin = 1,Volume = 15,Openout = 0,Pressure = 6 ? ;  
Openin = 1,Volume = 15,Openout = 1,Pressure = 6 ? ;  
Openin = 0,Volume = 25,Openout = 0,Pressure = 4 ? ;  
Openin = 1,Volume = 25,Openout = 0,Pressure = 4 ? ;  
Openin = 0,Volume = 25,Openout = 1,Pressure = 4 ? ;  
Openin = 1,Volume = 25,Openout = 1,Pressure = 4 ? ;
```

Useful for debugging/controllability in structural testing

Reachability analysis

```
initial(Openin,Openout,Pressure,Volume):-  
    sat(Openout), sat(~Openin), {Pressure-7=0}, {Volume-10=0}.  
reachable(Openin,Openout,Pressure,Volume,T):-  
    {T=0}, initial(Openin,Openout,Pressure,Volume).  
reachable(Openin,Openout,Pressure,Volume,T):-  
    {PT-T+1=0}, reachable(Popenin,Popenout,Ppressure,Pvolume,PT),  
    post(Popenin,Popenout,Ppressure,Pvolume,Openin,Openout,Pressure,Volume).
```

Aim : to compute $\text{Post}^*(\text{Init})$

Walking along reachable states

reachable(Openin, Openout, Pressure, Volume, T).

T = 0, Openin = 0, Volume = 10, Openout = 1, Pressure = 7 ? ;
T = 1, Openin = 0, Volume = 5, Openout = 1, Pressure = 8 ? ;
T = 1, Openin = 0, Volume = 15, Openout = 1, Pressure = 6 ? ;
T = 2, Openin = 0, Volume = 0, Openout = 1, Pressure = 9 ? ;
T = 2, Openin = 0, Volume = 10, Openout = 1, Pressure = 7 ? ;
T = 2, Openin = 0, Volume = 10, Openout = 1, Pressure = 7 ? ;
T = 2, Openin = 0, Volume = 20, Openout = 1, Pressure = 5 ? ;
T = 3, Openin = 0, Volume = 0, Openout = 1, Pressure = 9 ? ;
...

Need for a more compact representation.

Time abstraction

`reachable(Openin, Openout, Pressure, Volume, _).`

`Openin = 0, Volume = 10, Openout = 1, Pressure = 7 ? ;`

`Openin = 0, Volume = 5, Openout = 1, Pressure = 8 ? ;`

`Openin = 0, Volume = 15, Openout = 1, Pressure = 6 ? ;`

`Openin = 0, Volume = 0, Openout = 1, Pressure = 9 ? ;`

`Openin = 0, Volume = 10, Openout = 1, Pressure = 7 ? ;`

`Openin = 0, Volume = 10, Openout = 1, Pressure = 7 ? ;`

`Openin = 0, Volume = 20, Openout = 1, Pressure = 5 ? ;`

`Openin = 0, Volume = 0, Openout = 1, Pressure = 9 ?`

`...`

Need for widening operator.

Backward analysis

Is a given state reachable?

`reachable(Openin,Openout,3,30,T) .`

`T = 4,Openin = 0,Openout = 1 ? ;`

`T = 6,Openin = 0,Openout = 1 ? ;`

`T = 6,Openin = 0,Openout = 1 ? ;`

`T = 6,Openin = 0,Openout = 1 ? ;`

`T = 6,Openin = 1,Openout = 0 ? ;`

`T = 6,Openin = 1,Openout = 0 ? ;`

`T = 6,Openin = 1,Openout = 0 ? ;`

`T = 8,Openin = 0,Openout = 1 ?`

`...`

Property checking

$\{P \geq 8\}, \text{reachable}(_, _, P, V, T) .$

$P = 8, T = 1, V = 5 ? ;$

$P = 9, T = 2, V = 0 ? ;$

$P = 9, T = 3, V = 0 ? ;$

$P = 8, T = 3, V = 5 ? ;$

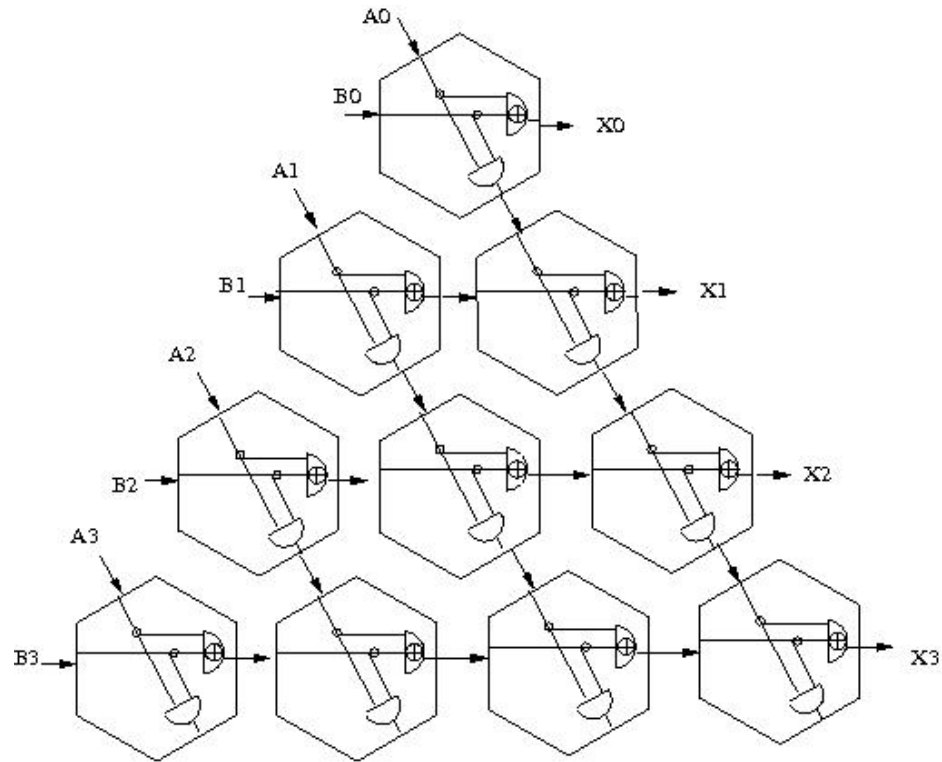
$P = 8, T = 3, V = 5 ? ;$

$P = 8, T = 3, V = 5 ? ;$

$P = 9, T = 4, V = 0 ?$

...

Diagonal carry adder



Equivalence checking in CLP(B)

```
andxor(A,B,X,C):-
    sat(X == A#B),sat(C == A*B).
adderandxor([A0,A1,A2,A3],[B0,B1,B2,B3],[X0,X1,X2,X3]):-
    andxor(A0,B0,X0,C00),
    andxor(A1,B1,X10,C10),andxor(X10,C00,X1,C11),
    andxor(A2,B2,X20,C20),andxor(X20,C10,X21,C21), andxor(X21,C11,X2,C22),
    andxor(A3,B3,X30,C30),andxor(X30,C20,X31,C31), andxor(X31,C21,X32,C32), andxor(X32,C22,X3,C33).

adder1bit(Ain,Bin,Cin,Xout,Cout) :-
    sat(Xout == Ain#Bin#Cin),sat(Cout == Ain*Bin+Ain*Cin+Bin*Cin).
adder4bits([A0,A1,A2,A3],[B0,B1,B2,B3],[X0,X1,X2,X3]):-
    adder1bit(A0,B0,0,X0,C1),
    adder1bit(A1,B1,C1,X1,C2),
    adder1bit(A2,B2,C2,X2,C3),
    adder1bit(A3,B3,C3,X3,C4).

test(A0,A1,A2,A3,B0,B1,B2,B3):-
    adder4bits([A0,A1,A2,A3],[B0,B1,B2,B3],[X0,X1,X2,X3]),
    adderandxor([A0,A1,A2,A3],[B0,B1,B2,B3],[Y0,Y1,Y2,Y3]),
    sat(X0==Y0),sat(X1==Y1),sat(X2==Y2),sat(X3==Y3).

| ?- test(A0,A1,A2,A3,B0,B1,B2,B3).
true
```

Conclusion

- Constraints programming is a good paradigm for analysis of synchronous programs
- A new solver is needed to represent globally set of states with boolean and numerical variables
- Concurrent constraints already need constraints for one execution path. What kind of constraints for a set of execution paths?