Symbolic Verification in Synchronous Constraint Programming (Work in progress)

Synchronous workshop 2003

Christophe Mauras
Christophe.Mauras@univ-nantes.fr

Irin/Université de Nantes

Overview

– From Symbolic Simulation to Constraint Programming
– Synchronous Constraint Programming
– From TDCC to verification in CLP

From Symbolic Simulation to Constraint Programming

– Early works : Chip (H. Simonis, 1987), Toupie (A. Rauzy, 1994)
– Symbolic simulation of synchronous programs (Marseille 1995)
– Synchronous programs verification vs static analysis in CLP (JFPLC 1999)

Symbolic simulation with YoYo

```
state    x,y : int;
initial  x=0 and y=0;
relation
if x=0 then x'=x+1 and y'=y+1
       else if y=0 then x-1<= x' and x'<=x and y'=y
                   else x<= x' and x'<=x+1 and
                        y-1<= y' and y'<=y+1 and
                        (x'>=x+1 <=> y'>=y+1)
```

Result :

```
t=0 : x=0 and y=0
t=1 : x=1 and y=1
t=2 : 2x<=y+2 and x>=1 and x>=y
t=3 : 2x<=y+3 and x>=y and y>=0
t=4 : 2x<=y+4 and x>=y and y>=0
t=5 : 2x<=y+5 and x>=y and y>=0
```

Translation in CLP(Q)

```
initial(X,Y):-{X=0,Y=0}.
transition(X,Y,NX,NY):-{X=0,-X+NX-1=0,-Y+NY-1=0}.
transition(X,Y,NX,NY):-{Y=0,-X+NX+1>=0,X-NX>=0,-Y+NY=0,X=\=0}.
transition(X,Y,NX,NY):-{-X+NX>=0,X-NX+1>=0,-Y+NY+1>=0,Y-NY+1>=0,
                        -X+NX-1>=0,-Y+NY-1>=0,X=\=0,Y=\=0}.
transition(X,Y,NX,NY):-{-X+NX>=0,X-NX+1>=0,-Y+NY+1>=0,Y-NY+1>=0,
                        X-NX>=0,Y-NY>=0,X=\=0,Y=\=0}.
reach(X,Y,T):-{T=0},initial(X,Y).
reach(X,Y,T):-{PT-T+1=0},reach(PX,PY,PT),transition(PX,PY,X,Y).
out(X,Y):-reach(X,Y,T).
```

Static Analysis in CLP

– STAN, M. Handjieva, LIX, 1996
– Unfolding, forward/backward analysis, polyhedra, widening
– Result : `out(X,Y) :- X-Y>=0, Y>=0, reach(X,Y,T)`

Execution in CLP

```
:- reach(X,Y,T).
T = 0,X = 0,Y = 0 ? ;
T = 1,X = 1,Y = 1 ? ;
T = 2,X = 2,Y = 2 ? ;
T = 2,X = 1,{Y=<1},{Y>=0} ? ;
T = 3,X = 3,Y = 3 ? ;
T = 3,X = 2,{Y=<2},{Y>=1} ? ;
T = 3,Y = 0,{X=<1},{X>=0} ? ;
T = 3,X = 2,{Y=<2},{Y=\=1},{Y>=1} ? ;
...
```

Synchronous Constraint Programming

– Timed Default Concurrent Constraint (TDCC) Saraswat 1996
– Gentzen constraint system $\longrightarrow$
  Boolean expression $\wedge$ Linear constraints
– Add pre in constraints and duplicate variables

Timed Default Concurrent Constraint Programming

Vijay Saraswat, Vineet Gupta et Radha Jagadeesan

References :

– CC, a Generic Framework for Domain Specific Languages.
– TCC : Programming in Timed Concurrent Constraint Languages, 94.
– Foundations of Timed Concurrent Constraint Programming, LICS 94.
– TDCC : Timed Default Concurrent Constraint Programming, J. Symb. Comp, 96.
– HCC : The Hybrid CC Programming Language-User Manual, 96.
– Programming in Hybrid Constraint Languages, 95.
– Hybrid CC, Hybrid Automata and Program Verification, 97.

Timed Default CC

Agents :

– { C }

– **if** C **then** A

– **if** C **else** A

– **new** X **in** A

– A, B

– **hence** A

Definable combinators :

– **next** A

– **always** A

– **do** A **watching** C

– **time** A **on** C

Synchronous Constraint Programming

- TDCC with mixed (integer/boolean) constraints (cf Jeannet's regions)
- It's a constraint system : closed under $\wedge$ and $\exists$
- Example : $\{(x1 <= 2) \wedge (x1 + x2 >= 3) \wedge (b1 \vee b2 \wedge b3)\}$
- Duplicate variables : $\{x1 = pre(x1) + 1\}$

Example

```
{x=0, y=0},
hence (if pre x=0 then {x=pre x+1, y=pre y+1},
       if pre x=0 else if pre y=0 then
                  {pre x-1<= x, x<=pre x, y=pre y},
       if pre x=0 else if pre y=0 else
          ({pre x<= x,x<=pre x+1,pre y-1<= y,y<=pre y+1},
           if x>=pre x+1 then {y>=pre y+1},
           if x>=pre x+1 else {y<=pre y} )
```

Interpreter for Synchronous Constraints

– Merge TDCC interpreter with CLP(B) and CLP(Q)
– Both are available in Sictus Prolog
– Work in progress ...

From TDCC to verification in CLP

– Model Checking in CLP (A. Podelski, G. Delzanno 99)
– Compile TDCC in constraint automata (if deterministic)
– Compute reachable state space by computing fixpoint of "direct consequence operator".

Model checking in CLP

```
init <- C(State), p(State)
p(State) <- C(State, State'), p(State')
```

- Predecessor $\leftrightarrow$ direct consequence operator T_p
- Add property $notf$ to the program (observator)
- Backward fixpoint computation

Simulation with synchronous constraints

```
p(State) <- C(state), init.
p(State') <- C(State, State'), p(State)
```

- Constraint automata is written in opposite order
- Store transitions as constrained facts
- Use T_p to compute post
- Compute forward reachable states from init

Example

```
r(p(X,Y),init,{X=0,Y=0},1).
r(p(X1,Y1),p(X,Y), {X=0,X1=X+1,Y1=Y+1},2).
r(p(X1,Y1),p(X,Y), {Y=0,X1=X,Y1=Y,X=\=0},3).
r(p(X1,Y1),p(X,Y), {Y=0,X1=X-1,Y1=Y,X=\=0},4).
r(p(X1,Y1),p(X,Y), {X1=X+1,Y1=Y+1,X=\=0,Y=\=0},5).
r(p(X1,Y1),p(X,Y), {X1=X,Y1=Y-1,X=\=0,Y=\=0},6).
```

Result :

```
s(0, p(A,B), {B=0,A=0}, 1, (0,0)).
s(1, p(1.0,1.0), {}, 2, (2,1)).
s(2, p(2.0,2.0), {}, 3, (5,2)).
s(2, p(1.0,0.0), {}, 4, (6,2)).
s(3, p(3.0,3.0), {}, 5, (5,3)).
s(3, p(2.0,1.0), {}, 6, (6,3)).
s(4, p(4.0,4.0), {}, 7, (5,5)).
s(4, p(3.0,2.0), {}, 8, (5,6)).
s(4, p(2.0,0.0), {}, 9, (6,6)).
```

Open Problems

– Generalize to automata where each state contains a CC program ?
– Is it correct to compute an over-approximation (like convex hull) with defaults ?
– Halbwachs's widening or Podelski's pseudo-widening ?

Conclusion

Much work remains to be done...