Simplifying Fixpoint Computations in Verification of Real-Time Systems

Second Workshop on Real-Time Tools
Copenhagen, Denmark, August 1, 2002

Jesper Blak Møller
[jm@it.edu]

Department of Innovation
The IT University of Copenhagen
[www.it.edu/people/jm]

Outline of Talk

1. Modeling of real-time systems
   - δ-programs
2. Verification of real-time systems
   - Reachability analysis using fixpoint computations
   - Clock resets and state changes as substitutions
   - Local-time semantics where clocks advance asynchronously
3. Experimental results
   - Fischer's protocol
   - Alpha and beta examples
4. Conclusion

Outline of Talk

1. **Modeling of real-time systems**
2. Verification of real-time systems
3. Experimental results
4. Conclusion

δ-programs (I)

- Simple notation for modeling real-time systems similar to timed guarded commands [Henzinger et al. 94]
- Allowing nondeterministic, independent-choice assignments of real variables
- Expressive enough to encode popular models of systems with time such as timed automata and timed Petri nets

δ-programs (II)

- δ-program (V, C) : A set of commands C of the form

$$\delta\vec{v}.\phi$$

$\vec{v}$: Vector of Boolean and real-valued variables V, and
ϕ : Expression of the form

$$\phi ::= b \mid x \sim d \mid x - y \sim d \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists b.\phi \mid \exists x.\phi$$

- Semantics:

$$\frac{s[\vec{v'} := \vec{r}] \models \phi}{s \xrightarrow{\delta\vec{v}.\phi} s[\vec{v} := \vec{r}]}$$

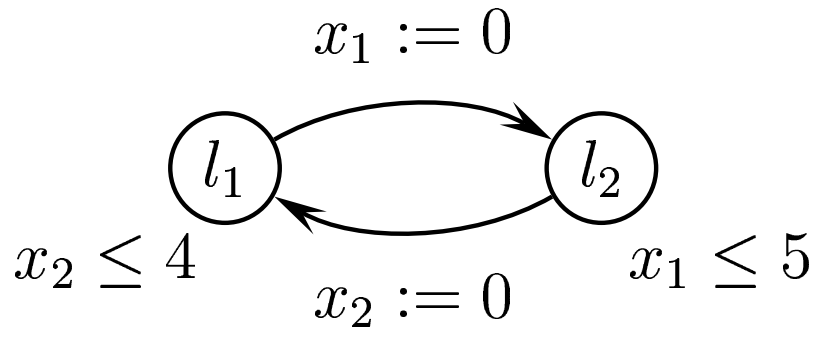
s : A state (an interpretation of the variables)
$\vec{r}$: Vector of Boolean and real-valued values

δ-programs (III)

- Example: δ-program over $V = \{l_1, l_2, x_1, x_2, z\}$:

$$\delta(l_1, l_2, x_1).(l_1 \wedge \neg l_1' \wedge l_2' \wedge x_1' = z)$$
$$\delta(l_1, l_2, x_2).(l_2 \wedge \neg l_2' \wedge l_1' \wedge x_2' = z)$$
$$\delta(z).\big(z' \leq z \wedge \forall z''(z' \leq z'' \leq z \Rightarrow (l_2 \Rightarrow x_1 - z'' \leq 5)) \wedge \forall z''(z' \leq z'' \leq z \Rightarrow (l_1 \Rightarrow x_2 - z'' \leq 4))\big)$$

Corresponding timed automaton:



Outline of Talk

1. Modeling of real-time systems
2. **Verification of real-time systems**
3. Experimental results
4. Conclusion

Reachability analysis (I)

- Let $P = (V, C)$ be a δ program with initial state ϕ_0
- A set of states of P can be represented by a formula ϕ
- Define two operators:
  - The set of states reachable from ϕ_0 :

$$\mathbf{post}(\phi_0) = \bigvee_{\delta\vec{v}.\phi \in C} \left(\exists \vec{v}.(\phi \wedge \phi_0)\right)[\vec{v}/\vec{v'}]$$

  - The set of states that can reach ϕ_0 :

$$\mathbf{pre}(\phi_0) = \bigvee_{\delta\vec{v}.\phi \in C} \exists \vec{v'}.(\phi \wedge \phi_0[\vec{v'}/\vec{v}])$$

Reachability analysis (II)

- Forward reachability:

  ϕ is an invariant iff $\models \mu X[\phi_0 \vee \mathbf{post}(X)] \Rightarrow \phi$

- Backward reachability:

  ϕ is an invariant iff $\not\models \mu X[\neg\phi \vee \mathbf{pre}(X)] \wedge \phi_0$

- Fixpoint computation of $\mu X[f(X)]$:

  $X := \mathbf{false}$
  repeat
  $X' := X$
  $X := f(X)$
  until $X = X'$
  return X

Clock resets and state changes as substitutions

- Resetting a clock $x := 0$ corresponds to the command $\delta x.(x' = z)$
- $\mathbf{pre}\big(\phi, \delta x.(x' = z)\big) \quad \equiv \quad \exists x'(x' = z \wedge \phi[x'/x]) \quad \equiv \quad \phi[z/x]$
- Thus, resetting a clock in backward reachability can be performed as a substitution
- Similarly for discrete state changes
- SAT of quantifier-free ϕ is **NP**-complete
- SAT of quantified ϕ is **PSPACE**-complete

Local-time semantics

- Idea: Introduce a zero point z_i for each clock x_i, and let clocks advance asynchronously
- Gives an over-approximation of the reachable state space
- In some situations we might still be able to prove properties for a δ-program
- Example from before:

$$\delta(l_1, l_2, x_1).(l_1 \wedge \neg l_1' \wedge l_2' \wedge x_1' = z_1)$$
$$\delta(l_1, l_2, x_2).(l_2 \wedge \neg l_2' \wedge l_1' \wedge x_2' = z_2)$$
$$\delta(z_1, z_2).\big(z_1' \leq z_1 \wedge \forall z_1''(z_1' \leq z_1'' \leq z_1 \Rightarrow (l_2 \Rightarrow x_1 - z_1'' \leq 5)) \wedge$$
$$z_2' \leq z_2 \wedge \forall z_2''(z_2' \leq z_2'' \leq z_2 \Rightarrow (l_1 \Rightarrow x_2 - z_2'' \leq 4))\big)$$

Outline of Talk

1. Modeling of real-time systems
2. Verification of real-time systems
3. **Experimental results**
4. Conclusion

Experimental results

- Implemented a symbolic model checker in SML using DDDLIB for representing formulas ϕ
- Two examples:
  - Fischer's mutual exclusion protocol
  - Bozga et al.'s alpha and beta examples

Experimental results: Fischer's protocol

	Forward		Backward		Local-time backward	
N	CPU time	DDD size	CPU time	DDD size	CPU time	DDD size
2	1.1	36	1.0	16	1.0	16
4	1.2	405	1.1	259	1.1	139
6	10.6	4,478	1.3	979	1.2	275
8	2138.0	50,291	2.3	3,383	1.4	411
10			6.3	12,331	1.5	547
12			27.8	47,263	1.6	683
14			589.2	185,939	1.7	819
16			8521.5	739,399	2.2	955
32					6.3	2,043
64					27.0	4,219
128					161.5	8,571
256					1318.0	17,275
512					5602.0	34,683

Experimental results: Alpha and beta examples

	Local-time forward			
	Alpha		Beta	
N	CPU time	DDD size	CPU time	DDD size
16	0.1	50	0.1	34
32	0.2	98	0.2	66
64	0.4	194	0.6	130
128	1.9	386	2.8	258
256	9.8	770	16.3	514
512	49.1	1538	74.4	1026
1024	212.0	3074	317.1	2050

Outline of Talk

1. Modeling of real-time systems
2. Verification of real-time systems
3. Experimental results
4. **Conclusion**

Conclusion

- Forward reachability performs comparably with KRONOS and UPPAAL
- Backward reachability is faster and more space efficient than forward reachability
- Local-time backward reachability:
  - Possible to verify mutual exclusion of Fischer's protocol with up to 512 processes with linear runtimes
  - Possible to compute the exact reachable state spaces for alpha and beta in linear time
- There is no guarantee that the proposed local-time model will always work as well as in the studied examples
- An over-approximation might become too large and hence not useful