Constrained Multiset Rewriting

Parosh Abdulla[1] and Giorgio Delzanno[2]

[1] *Uppsala University*   [2] *University of Genova*

Padova, Aprile 2006

Background

- Practical examples of multithreaded programs and protocols for distributed systems often have
  - *unbounded data*: generation of fresh names, ...
  - *unbounded control*: spawning of new processes, ...
  - *unbounded data and control*: multithreaded software
  - *process mobility*: dynamic reconfiguration of the network programs,...

- Can we still apply *automated verification techniques* when their *state-space* becomes infinite in *one* or *more dimensions*?

Bounded control, unbounded data

Constraints to symbolically represent data

- Henzinger-Ho-Wong-Toi. *HyTech: a Model Checker for Hybrid Systems*, CAV'97 — Based on the Polyhedra library

- Bultan-Gerber-Pugh. *Symbolic Model Checking of Infinite State Systems Using Presburger Arithmetics*, CAV'97 — Based on the Omega Library

- ...

Unbounded control, bounded data

Constraints to symbolically represent sets of processes

- Bouajjani-Jonsson-Nilsson-Touili. *Regular Model Checking*, CAV 00

  BASED ON REGULAR LANGUAGES

- German-Sistla. *Reasoning about Systems with Many Processes*, JACM 92

  BASED ON PETRI NETS

- Esparza-Finkel-Mayr. *Verification of Broadcast Protocols*, LICS 99

  SYMBOLIC ANALYSIS FOR PETRI NETS

Unbounded data and parameterized control

- Abdulla-Jonsson. *Verifying Networks of Timed Processes*, TACAS'98
- Abdulla-Nylén. *Better is Better than Well: On Efficient Verification of Infinite-State Systems*, LICS'00

BASED ON SYMBOLIC MODEL CHECKING

- Arons-Pnueli-Ruah-Xu-Zuck. *Parameterized Verification with Automatically Computed Inductive Assertions*, CAV'01

BASED ON ABSTRACTIONS+DEDUCTIVE VERIFICATION

- Lazic-Newcomb-Roscoe *Polymorphic Systems with Arrays, 2-Counter Machines and Multiset Rewriting*, Infinity'04

BASED ON PARTIAL FUNCTIONS

Current Research Line

Overall goal

To develop *sound* and *fully-automatic* methods based on *constraint programming* technology for the verification of concurrent systems with

- *unbounded control*
- *unbounded data*
- *process mobility*

Practical applications

Consistency protocols for distributed systems with shared memory
Cache coherence protocols for multi-processors and multi-line caches
Security protocols
Abstractions of multithreaded programs
Mobile systems

Several Problems to Solve

- We need a specification language for *parameterized systems with unbounded local data*
- We need an *assertional language* to specify safety properties
- We need *sound* and *fully automatic* procedures to validate the specification against the desired property

Low Level Specification Language

CMRS: Constrained Multiset Rewriting

- *Multiset rewriting over first order atomic formulas* (MSR) can be used as a flexible specification language for concurrent systems
- MSR has been introduced to specify *security protocols*
  - Locality of process definitions and communication via rendez-vous
  - First order terms as color for processes
- The combination of MSR with a *constraint system* $\mathcal{C}$ can be used to *symbolically* represent systems with heterogeneous data structures

An example: A Mutual Exclusion Protocol

Initialization and halting phase

$$
\begin{array}{lcl}
[\ init\ ] & \rightarrow & [\ v_0(X), initP(Id)\ ] : true \\
[\ initP(Id)\ ] & \rightarrow & [\ idle(Id), initP(Next)\ ] : Next > Id \\
[\ idle(X)\ ] & \rightarrow & [\ \ ] : true
\end{array}
$$

Core Protocol

$$
\begin{array}{llcl}
1. & [\ idle(X), v_0(Y)\ ] & \to & [\ waiting(X), v_0(X)\ ] : true \\
2. & [\ v_0(X)\ ] & \to & [\ v_1(X)\ ] : true \\
3. & [\ waiting(X), v_1(Y)\ ] & \to & [\ idle(X), v_1(Y)\ ] : X \neq Y \\
4. & [\ waiting(X), v_1(X)\ ] & \to & [\ cs(X), v_1(X)\ ] : true \\
5. & [\ cs(X), v_1(Y)\ ] & \to & [\ idle(X), v_0(Y)\ ] : true
\end{array}
$$

Configuration and Run

A Configuration is a multiset $\mathcal{M}$ of *ground* atomic formulas

One Step Rewriting

$$[\underline{idle(2)}, idle(1), waiting(0), \underline{v_0(1)}] \quad \Rightarrow$$
$$[\underline{\underline{waiting(2)}}, idle(1), waiting(0), \underline{\underline{v_0(2)}}]$$

using the instance of the first rule

$$[\ idle(2), v_0(1)\ ] \quad \to \quad [\ waiting(2), v_0(2)\ ]$$

Reachability $\mathcal{M}$ is reachable if $init \stackrel{*}{\Rightarrow} \mathcal{M}$

Mutual Exclusion There cannot be two processes with local state cs

Properties and Assertional Language

Parameterized Verification of Safety

- Let S be the set of *good configurations*. The corresponding *safety property* holds if for any $\mathcal{M}$

$$if\ init \stackrel{*}{\Rightarrow} \mathcal{M}\ \ then\ \ \mathcal{M} \in S$$

- Dually, let U be the set of *bad configurations*, then the property holds if

$$init \notin Pre^*(U)$$

where $Pre^*(U) \ = \ \{ \ \mathcal{M} \mid \mathcal{M} \stackrel{*}{\Rightarrow} \mathcal{M}', \ \mathcal{M}' \in U \ \}$

- We have to explore a potentially infinite number of configurations

Symbolic Representation of Configurations

- In our example the set of *unsafe states* can be represented as the *constrained configuration*:

$$[cs(i_1), cs(i_2)] \ : \ i_1 \geq 0, i_2 \geq 0$$

- if we consider its *upward-closed denotations*

$$[\![\mathbf{U}]\!] \ = \{ \ [cs(i), cs(j)] \oplus \mathcal{M}, \ \ for \ any \ i, j \in Nat, \ \ for \ any \ conf. \ \mathcal{M} \ \}$$

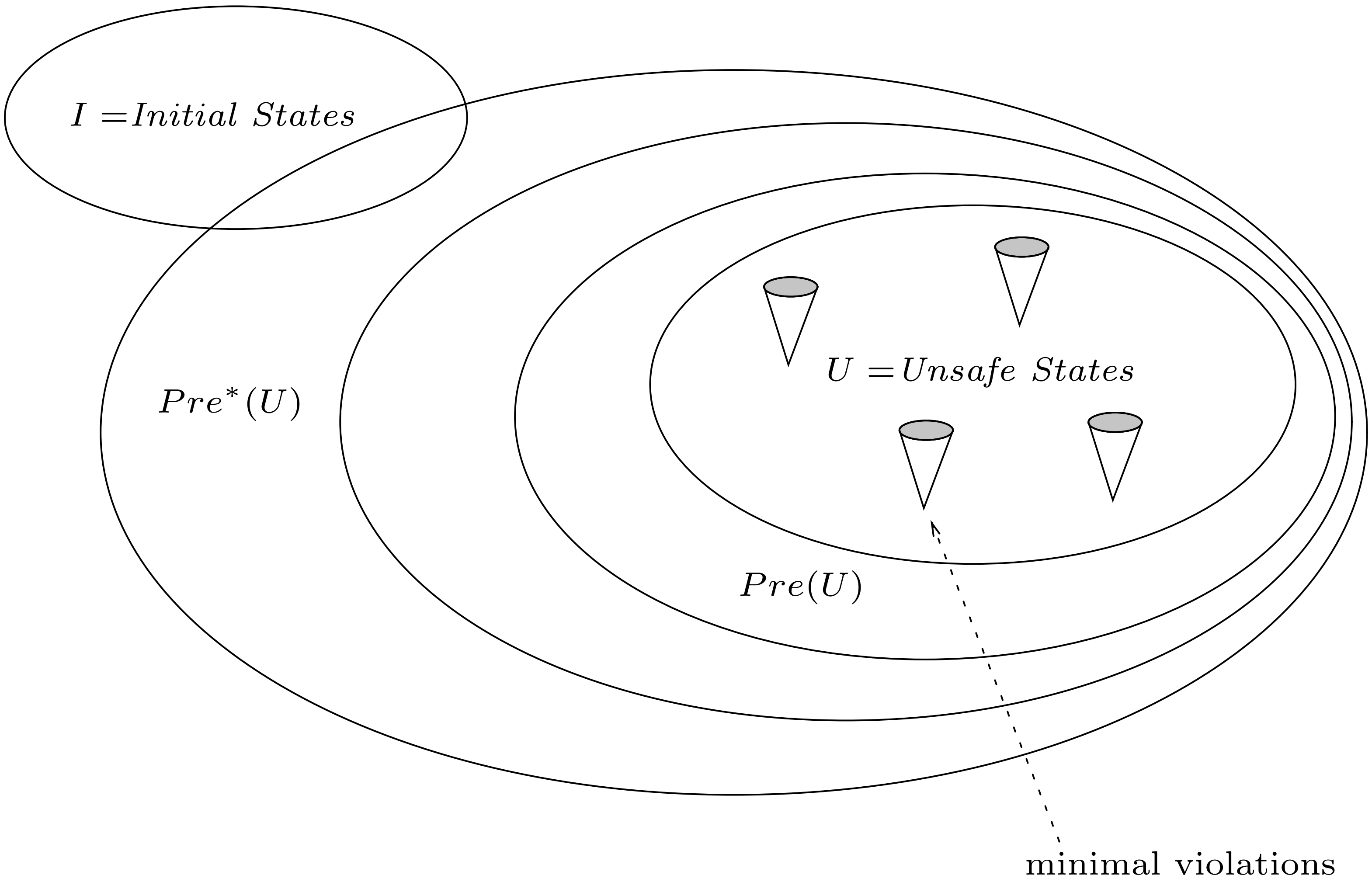
- defined in general as follows

$$[\![\mathcal{M} : \varphi]\!] \ = \{ \ \mathcal{N} \mid \sigma(\mathcal{M}) \preccurlyeq \mathcal{N}, \ \sigma \ solution \ of \ \varphi \ \}$$

- E.g. $[cs(1), cs(2)]$, $[cs(1), cs(3), cs(4)]$ belong to $[\![\mathbf{U}]\!]$.

Verification Procedures

Backward Reachability



Pre-image Computation

From

$$[p(u), \underline{\underline{p(v)}}] \ : \ true$$

using the rule

$$[\underline{\underline{w(x)}}, \underline{t(y)}] \ \longrightarrow \ [\underline{\underline{p(x')}}, \underline{t(y')}] : x = y, x' = x, y' = y$$

we get

$$[p(u), \underline{\underline{w(x)}}, \underline{t(y)}] : x = y$$

but also

$$[p(u), p(v), \underline{\underline{w(x)}}, \underline{t(y)}] : x = y$$

Entailment

- We define an ordering based on *AC unification* and on the *entailment* relation of the underlying constraints:
- For instance

$$[p(x,y), q(z), r(u)] : x > y, y = z$$

$$entails$$

$$[q(z'), p(x',y')] \; : x' > y'$$

- Infact,

  $[p(x,y), q(z)]$ and $[q(z'), p(x',y')]$ unify via $x = x', y = y', z = z'$

  $x' > y', x' = z'$ entails $x' > y'$.

Sufficient Conditions for Termination

- Symbolic backward reachability terminates when
  - Predicates are monadic ($p(x)$ ok, $p(x, y)$ not ok)
  - Constraints are *gap-order* (Revesz '93), i.e., they are conjunctions of atomic constraints of the form

    $x + c < y$

    $x = y$

    $x@c$

    where $@ \in \{\leq, \geq, <, >\}$, c is a *natural number*, and x, y are interpreted over *natural numbers*

- A example of rule in the fragment

$$[p(x), m(y)] \rightarrow [q(v), n(w)] : x + 1 < y, v > 2, w = x$$

- A example of symbolic configuration in the fragment

$$[p(x), p(v), q(y), r(y), s(z)] : x = v, x + 1 < y$$

Termination

- Termination can be proved using a non trivial application of the theory of well-quasi orderings
- Intuition: a configuration like

$$[p(x), p(v), q(y), r(y), s(z)] : x = v, x + 1 < y$$

can be represented a finite set of strings built on multisets of predicate symbols and integers (gaps)

$$[p, p]1[q, r]0[s] \quad [s]0[p, p]1[q, r] \quad [p, p, s]1[q, r] \quad \ldots$$

- We order sets of strings combining pointwise ordering for sets, embedding of strings and embedding of multisets
- The resulting order
  - can be used to check the containment of the denotations of two symbolic configurations (termination test for backward reachability)
  - is a well-quasi ordering (there cannot be infinite sequences of incomparable sets of strings)
- These properties guarantee termination of symbolic backward reachability

Conclusions

- *Push-button* verification method for infinite-state concurrent systems based on the paradigm of symbolic model checking and constraints
- Application to *nominal process calculi* with *unbounded control*, *fresh name generation*, and *name mobility* [TPLP 2006]
- Application to *verification* of *security protocols* [TACAS 2004]
- Possible application to pointer analysis?
- Specialized data structures are needed to scale up
- Abstractions/accelerations are needed for terminations (class of widening operators for security protocols?)