Experience with Vereofy

Natallia Kokash

SYANCO meeting, Dresden, Germany
Introduction

- Application
  - Why Reo and Vereofy?
  - Case studies
- Analysis with Reo/CA and Vereofy
  - Control flow
    - Advanced Control Flow
  - Data flow
- Desired features for Vereofy
- Conclusions
Application

- COMPAS project
  - Compliance-driven Models, Languages, and Architectures for Services

- Compliance requirements
  - Constraints or assertions that results from the interpretation of the compliance sources (e.g., SOX, HIPAA, MIFID, IFRS)
  - Temporal (control flow) – what has to be done within the process
  - Monitoring, payment, privacy, QoS, information retention, security, transaction...
Why Reo and Vereofy?

- Formal modeling and verification of compliance requirements on business processes and web service compositions
  - Processes and service compositions are modeled in Reo
    - Can be obtained from high-level models (UML, BPMN) or BPEL code
    - Graphical notation + high expressivity + extensibility + compositionality
  - Constraint automata is a suitable formal model
    - Compliance constraints result in data or time-based constraints on transitions

- Vereofy is the main tool for analyzing
Formalization of compliance requirements

- From section in a legal document to a formal rule – University of Tilburg
  - $R_1$: “The bank does not officially sign the form unless the credit worthiness check has been performed”

- LTL, CTL, Deontic Logic, FCL, etc.

- Extensions
  - E.g., exceptions – “$R_1$ is not necessary for trusted clients”
Case study (1): THALES – Loan request process
Case study (2):
TARC-PL – Advanced mobile services
Case study (2) – part 2
Compliance requirements

- Loan Request Scenario:
  - Security:
    - Strong authorization
    - Risk analysis
    - Segregation of duties
  - Transactional properties:
    - Any loan request can be cancelled

- WatchMe:
  - Adherence to user contracts and license conditions
    - QoS, timeouts, etc.
Control flow verification

- Typical errors:
  - **Deadlock** is a situation when a service or a process stays idle waiting for resources permanently blocked by another party or messages that will never arrive.
  - **Livelock (starvation of progress)** is a situation when a process or a service is not blocked but doesn't make any progress nevertheless. A livelock may appear, for example, in a process with a loop structure without an appropriate end condition.
  - **Problems with synchronization** conflicts can be introduced by joining fork concurrent paths with a merge structure which in some task-flow specification languages (e.g.,
Control flow analysis with Reo and Vereofy

BPMN2Reo converter
Control flow verification

- **Vereofy:**
  - ASL: $\text{AG}[\text{EX}[true]]$ – check for deadlocks
  - LTL: $\text{G}(\text{request} \rightarrow \text{F} (\text{reject} \cup \text{sendFormOut}))$ – check that admissible states are reached

- **State of the art of early 90th**
Advanced control flow analysis

- Long-Running Business Transactions

Compensatable service

Transactional Service Composition
Advanced control flow analysis

- **Properties to check:**
  - The committed state or the canceled state of each started compensation pair $C_i$ will be eventually achieved:
    \[ \forall i, 1 \leq i < n, \quad G(C_i.start \rightarrow F(C_i.committed \vee C_i.cancelled)) \]
  - Arrival of a cancelation message implies that all performed activities will be eventually compensated:
    \[ G(M \rightarrow F \land \bigwedge_{i=1}^{n} C_i.cancelled) \]
  - For a reverse order cancelation, we should show that for each activity the compensation of its preceding activity starts at the next step after the current activity:
    \[ \forall i, 1 \leq i < n, \quad G((C_i.start \rightarrow X(\neg C_{i+1}.start \land M)) \rightarrow F C_1.cancelled \land \bigwedge_{j=2}^{i} (C_j.cancelled \rightarrow X C_{j-1}.cancel).) \]
Advanced control flow analysis

- **Timed LRT**
  - Activity or a sub-process which must be interrupted and compensated for if not completed within a specified time-out.
Advanced control flow analysis

- Modeling processes with fault, termination and compensation handling in Reo
  - Priorities are needed
  - Time
- Verification
  - Several connectors (also components or sub-connectors)
  - Specification format: iteration
  - Verification of timed constraint automata
Data flow analysis

24/08/2009
Data flow analysis
“Check” process fragment

- Input parameters:
  - Activation condition
    - **Data:** b: Boolean
    - **Filter condition:** b==true, b==false
  - Check condition
    - **Data:** x, y: Real; (e.g., credit amount, maximal amount)
    - **Filter condition:** x < y

- Filter constraints supported by Vereofy:
  - =c, >c, <c, ≠c, ≤c, ≥c, where c is a constant

- Problems:
  - Several variables are needed
  - More powerful constraint language is needed
  - Properties that include conditions:
    - $G \neg [(b \land \neg (x < y)) \rightarrow F \text{ violation}]$
Desired features: operations with data

- Multiple data items (variables)
  - x: Integer; y: Real; b: Boolean;

- Operations on data: create, destroy, copy, initialize, assign, etc.
  - b = (x==y);

- Ability to deal with different data types
  - Components (readers/writers) may generate/consume items of particular data type
  - Support for abstract data types

- Constraint definition language:
  - Filter constraints may include different variables, e.g., x+y<10

User-defined functions for transformers
Desired features: operations with constraint automata

- Ability to edit constraint automata
- CA minimization modulo bisimulation equivalence
- CA equivalence checking
- Support for join nodes
  - \( dY = \text{tuple}(dX, dY); \)
- Support for transformer and timer channels
- Verification across several connectors
- Time, context dependency and priority
  - \( A|B + A \Rightarrow \text{if not } A|B \text{ then } A \)
  - Intentional automata?
  - Timed constraint automata
  - Probabilistic (timed) constraint automata
Desired features: property specification language

- Unify LTL and ASL formats
  - State names “A” vs. A
  - In ASL syntax many port names cause parsing errors (e.g., `isCheckOk?`, `1`, `2`...)

- Make exception messages while parsing formulae more user-friendly

- Ability to refer to
  - FIFO1 channels using port names (not just `fifo1[0]`, `fifo1[1]`)
  - Filter conditions
  - States

- Global QoS constraints
  - For all paths $p$, $\text{executionTime}(p) < t$

- Ability to use constraints on variables and functions in formulae:
  - E.g., for all $y$, $x$: Int[$0..100$] such that $x == y^2$ state A is reachable...
Other tools

- mCRL2 toolset
  - Advantages:
    - Powerful support for data (most of the examples above can be modeled)
    - Richer property specification format
  - Disadvantages:
    - Hard to predict results and extract counterexamples
    - For infinite domains model checker often does not terminate (problems with formulae rewriting?)
    - Inability to define some useful data domains
    - Intentional model: state explosion problem!
Conclusions

- Successfully applied to control flow verification
- Almost no support for data flow analysis
- Almost no support for timed control flow and QoS analysis