Verification of Composed Array-based Systems with Applications to Security-Aware Workflows

Clara Bertolissi[1,2] and ***Silvio Ranise***[2]

[1]LIF-CNRS, UMR 7279 & AMU, Marseille, France

[2]FBK, Trento

FroCoS (Nancy) - September 18-20, 2013

Context

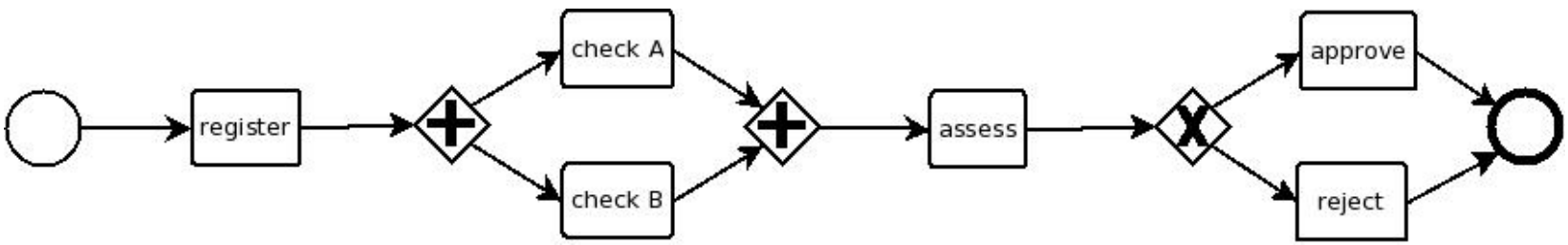
- Workflow management used in several applications
  - E-business
  - E-health
  - E-government
  - Scientific computing
  - ...
- Workflow management specification
  - What are the tasks*?*
  - What is the order of execution of the tasks*?*
  - Which data are manipulated by each task*?*
  - Who performs the tasks*?*

Simple example: insurance claim (control-flow)

What are the tasks?

- register insurance claim
- check A of insurance policy
- check B of damage reported
- assess the results of checks A and B
- approve the payment of damage
- reject the payment of damage

What is the order of execution of the tasks?



Simple example: insurance claim (access control)

Who performs the tasks?

- Three roles: Customer Service, Specialist A, Specialist B
- Six users: Anna, Adam, Benn, Beate, Carol, Chris

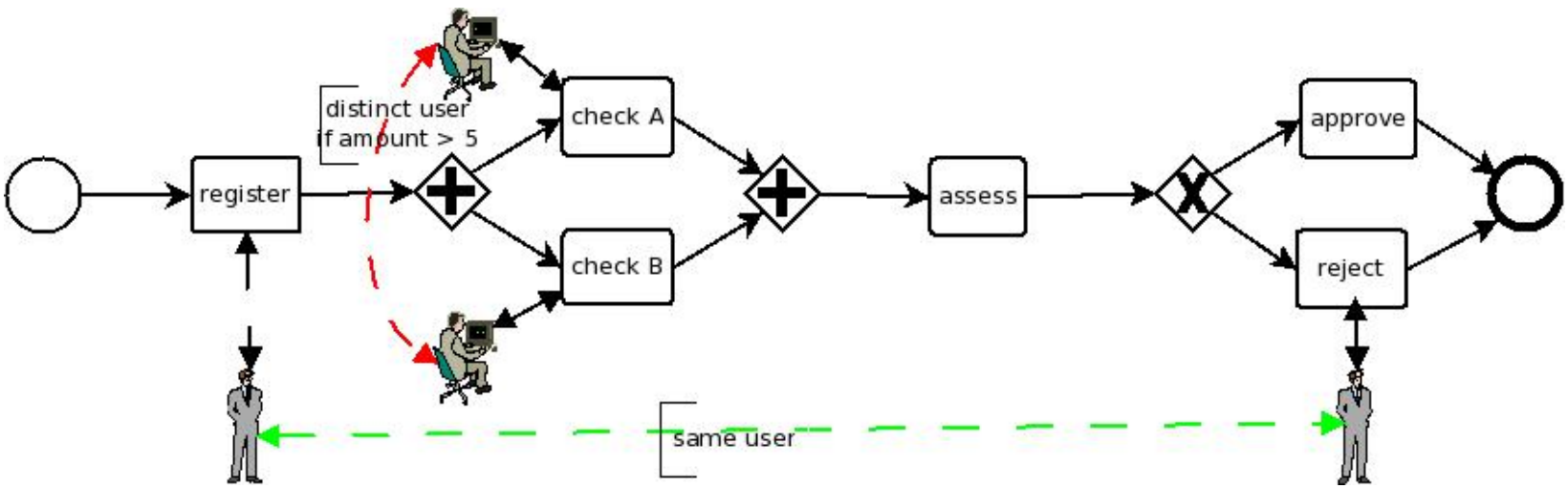
User	Role
Anna	Customer Service
Adam	Customer Service
Benn	Specialist A
Beate	Specialist A
Beate	Specialist B
Carol	Specialist B
Chris	Specialist B

Role	Task
Customer Service	register
Customer Service	assess
Customer Service	approve
Customer Service	reject
Specialist A	check A
Specialist B	check B

- Role-Based Access Control (RBAC)

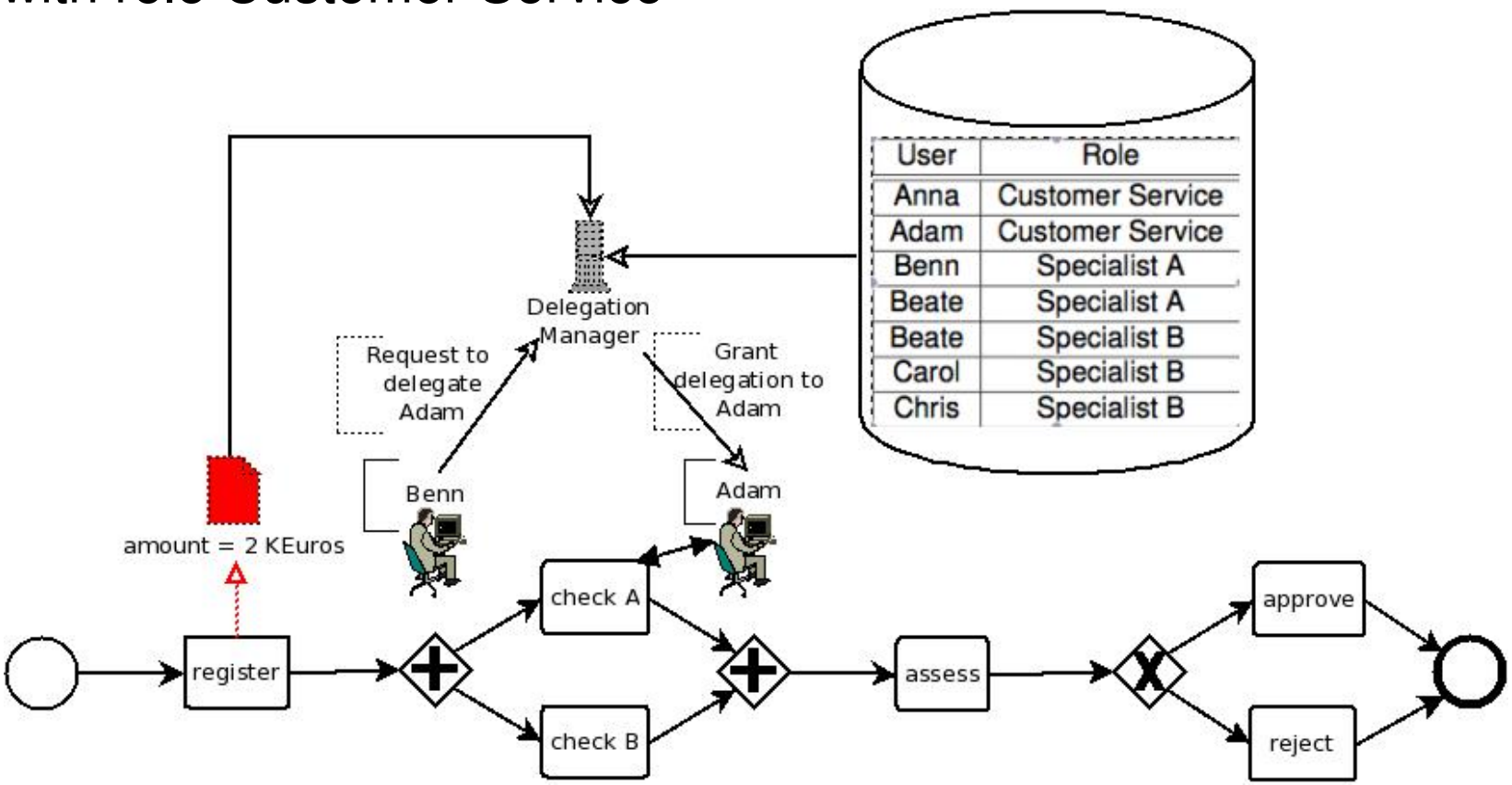
Insurance claim: more access control (I)

- Authorization constraints: Separation/Bound of Duty (SoD/BoD)
  - SoD: If amount is larger than 5 KEuros, then the same user cannot execute both tasks check A and check B
  - BoD: Task reject have to be performed by the same user who performed the task register



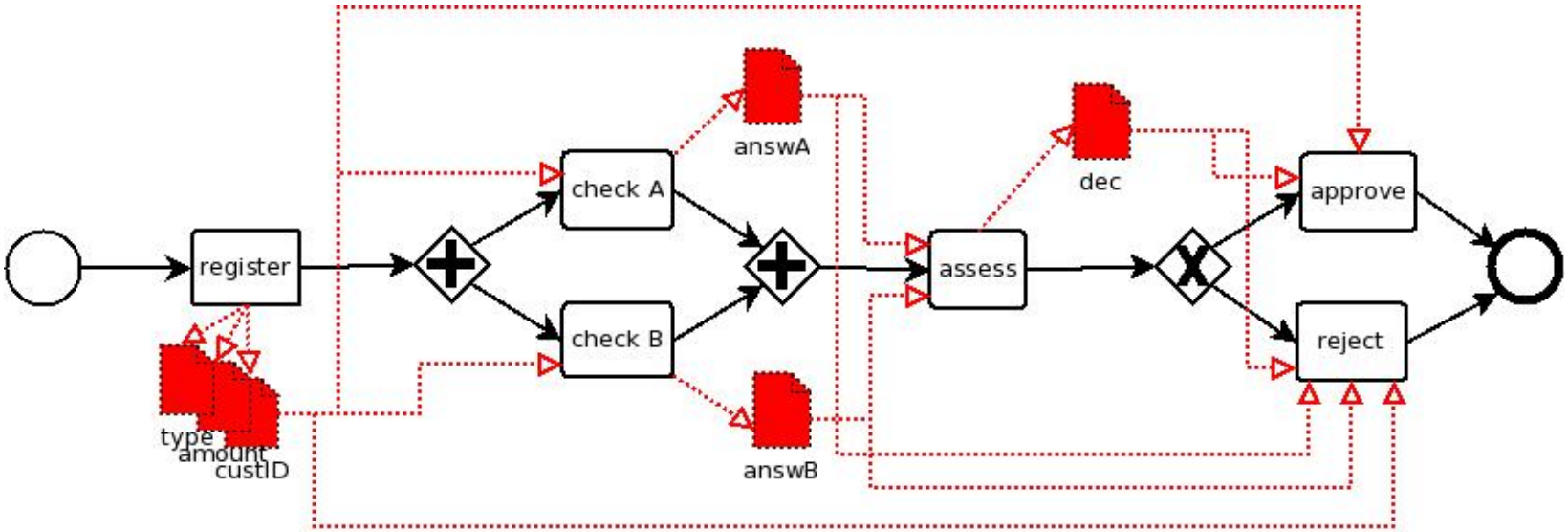
Insurance claim: more access control (II)

- Delegation of task execution
- Rule: if amount is less than 5 KEuros, then user with role Specialist A can delegate the right to execute task check A to user with role Customer Service



Simple example: insurance claim (data-flow)

Which data are manipulated by each task?



- custID: unique identifier for customer
- type: enumerated data-type for identifying type of damages
- amount: money requested for damage
- answA, answB: either “ok” or “nok”
- decision: either “grant” or “refuse”

An interesting question

Given a workflow specification, is a SoD/BoD constraint redundant?

Issues:

- control-flow semantics of workflows $=$ Safe Petri nets
- Finite but unknown number of
  - workflow instances
  - users
- Express
  - authorization constraints
  - data structures to model data-flow

Key observation

Workflow specifications are structured along two dimensions:

1. parametric in the number of workflow instances/users/...

   (i.e. finitely many but unknown number of workflow instances/...)

2. layered in components: one component per aspect, e.g.,
   - control-flow
   - authorization
   - data-flow

   components only reduce possible behaviors of workflows by adding constraints

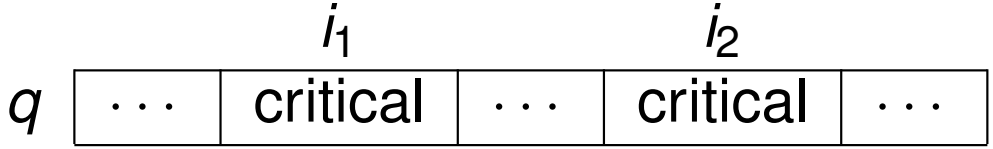
How to be Parametric (I)

- Use Array-based Systems [Ghilardi & R., LMCS'10]
- Symbolic transition systems where state variables are arrays, e.g.,

$$q : \text{Nat} \rightarrow \text{Locations}$$

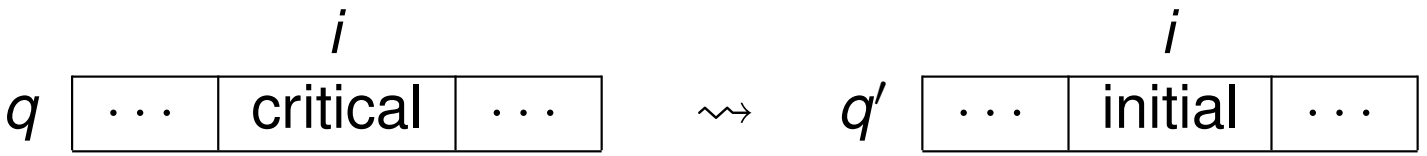
- Set of states = class of existentially quantified formulae, e.g.,

$$\underbrace{\exists i_1, i_2.(i_1 \neq i_2 \wedge q[i_1] = \text{critical} \wedge q[i_2] = \text{critical})}$$



- Transitions = class of formulae in functional form, e.g.,

$$\underbrace{\exists i.(q[i] = \text{critical} \wedge q' = \lambda j.(\text{if } j = i \text{ then initial else } q[j]))}$$



- Types = Theories, e.g., Nat as the theory of $\mathbb{N}$

How to be Parametric (IIa)

- Array-based system supports Infinite State Model checking by
  - symbolic backward reachability analysis [→ see next slide]
  - pre-image computation = logical manipulations of formulae
    Closure under pre-image computation
  - fix-point checking = SMT solving
    Decidability of fix-point checking and even termination

How to be Parametric: Backward Reachability (IIb)

function BReach(G)
  $i \longleftarrow 0$; $BR^0(\tau, G) \longleftarrow G$; $K^0 \longleftarrow G$
  if SMT($BR^0(\tau, K) \wedge I$) $=$ sat **then return** unsafe
  repeat
    $K^{i+1} \longleftarrow \mathrm{Pre}(\tau, K^i)$
    $BR^{i+1}(\tau, K) \longleftarrow BR^i(\tau, K) \vee K^{i+1}$
    if SMT($BR^{i+1}(\tau, K) \wedge I$) $=$ sat **then return** unsafe
    else $i \longleftarrow i + 1$
  until SMT($\neg(BR^{i+1}(\tau, K) \rightarrow BR^i(\tau, K))$ $=$ unsat
  return safe
end

G: goal formula usually obtained by complementing an invariant

How to be Parametric (IIc)

- Array-based system supports Infinite State Model checking by
  - symbolic backward reachability analysis
  - pre-image computation = logical manipulations of formulae
    Closure under pre-image computation
  - fix-point checking = SMT solving
    Decidability of fix-point checking and even termination

- Problem: while data stored in arrays can be of several types...

  ... indexes of arrays are of just one type*!*

  $\Longrightarrow$ difficult to accomodate workflow instances, users, etc
- Solution: permit multiple types for indexes

  easy extension $\Longrightarrow$ composed array-based systems

  BUT closure, fix-point decidability, and termination do not lift*!*

Layering enters

- Recall that for the class of workflow systems we are interested in, we have one component per aspect, e.g.,
  - control-flow: expressed in terms of workflow instances only
  - authorization: expressed in terms of users only
  - system = composition (conjunction) of the components

- Set of states = conjunction of existentially quantified formulae

$$a : I_a \to E_a \qquad b : I_b \to E_b$$

$$\exists \underbrace{i_1, ..., i_n}_{\text{type } I_a \text{ only}} . \varphi(i_1, ..., i_n, a) \wedge \exists \underbrace{k_1, ..., k_m}_{\text{type } I_b \text{ only}} . \psi(j_1, ..., j_m, b)$$

- Transition = conjunction of formulae in functional form

$$\exists \underbrace{i_1, ..., i_n}_{\text{type } I_a \text{ only}} . (C_a \wedge a' = \lambda k.(\cdot\ a\ \cdot)) \wedge \exists \underbrace{j_1, ..., j_m}_{\text{type } I_b \text{ only}} . (C_b \wedge b' = \lambda k.(\cdot\ b\ \cdot))$$

n-components Array-Based Systems

Lifting model checking of array-based systems to n-components array-based systems

- Pre-image computation is modular

$$Pre(S_a \wedge S_b, \tau_a \wedge \tau_b) = Pre(S_a, \tau_a) \wedge Pre(S_b, \tau_b)$$

- Fix-point computation is modular

$$(S_a \wedge S_b) \Rightarrow Reach \text{ is valid} \quad \text{iff} \quad \begin{array}{l} S_a \Rightarrow Reach|_a \text{ is valid and} \\ S_b \Rightarrow Reach|_b \text{ is also valid} \end{array}$$

where $Reach := \bigvee_{j \in J}(R_a^j \wedge R_b^j)$ and

$Reach|_x := \bigvee_{j \in J} R_x^j$ for $x \in \{a, b\}$

n-components Array-Based Systems: modularity of termination (I)

Recall the following for array-based systems:

- set of configurations = models of existential formulae
- partial order $\preceq$ = "embedding" between models
- if $\preceq$ well-quasi-order, then backward reachability terminates
- sufficient conditions on theories for $\preceq$ to be well-quasi-ordering:

Index	Element	$\preceq$ is a wqo by
Pure equality	Enumerated data-type	Dickson lemma
Total order	Enumerated data-type	Higman lemma
Pure equality	Rationals with $<$	Kruskal theorem

n-components Array-Based Systems: modularity of termination (II)

- Also recall that if $\preceq_a$ and $\preceq_b$ are well-quasi-orders, then also $\preceq_{a,b}$ is so, where

$$(a, b) \preceq_{a,b} (a', b') \quad \text{iff} \quad a \preceq_a a' \text{ and } b \preceq b'$$

$$\Longrightarrow \text{key to prove Dickson Lemma}$$

- Termination is modular: if well-quasi-order on each component of n-component array-based system, then backward reachability terminates

- Reuse of sufficient conditions for termination of array-based systems

Back to workflow verification

Two interesting classes of workflow systems

- Constrained workflow systems
  - Control-flow: Pure theory of equality (Workflow instances) + Enumerated Datatype (Tokens in places of Safe Petri net)
  - Authorization: Pure theory of equality (Users) + Enumerated Datatype (Booleans for role set)
- Constrained workflow systems with numerical data-flow
  - Control-flow and authorization as above
  - Data-flow: Pure theory of equality (Users) + Rationals with $<$ only (Integers with ordering relation only)
- Termination of backward reachability is guaranteed*!*
- It is always possible to answer the question: Is a SoD/BoD constraint redundant*?* by reduction to reachability problem

Summing up

- n-component array-based system: parametric + layered
- Components allow for re-using results for array-based systems
- Just of first step in the application to workflow verification...

- Future work: solving the Workflow Satisfiability Problem by a combination of infinite-state model checking and SMT solving