Authorization Enforcement in Workflows: Maintaining Realizability Via Automated Reasoning

Jason Crampton, Michael Huth, and Jim Huan-Pu Kuo

1 July 2012, PAAR-2012 Workshop
Example workflow specification

Blue edges: temporal constraints. Binding of users to tasks constrained by equality $\equiv$, inequality $\neq$, and seniority $\prec$. 

Jason Crampton, Michael Huth, and Jim Huan-Pu Kuo
Authorization Enforcement in Workflows
Representative specification formalism

Specification of authorization schema $\mathcal{AS}$ comprised of:

- $(T, \leq)$ finite partial order of tasks: $t < t'$ means $t$ has to precede $t'$
- $U$ set of users
- $A \subseteq T \times U$ where $(t, u)$ in $A$ means: $u$ authorized to execute task $t$
- $C$ set of entailment constraints of form $(D, t \rightarrow t', \rho)$
  - $D \subseteq U$ and $\rho \subseteq U \times U$
  - meaning: if $u$ in $D$ and assigned to task $t$, then user $u'$ assigned to $t'$ is such that $(u, u')$ is in $\rho$
- e.g. $=$ as $\rho$ and $D$ as $U$ gives Binding of Duty
Unrealizable workflow example

- Alice hasRole FinAdmin, Bob hasRole FinClerk
- FinAdmin authorized to approve orders and payments
- FinClerk authorized to all other tasks
- Workflow below not realizable: Alice is most senior person
Unrealizable workflow: details

- If Alice creates order, no senior person can approve it.
- If Bob creates order, Alice needs to approve it (≺).
- But Alice also has to create payment because of ≠.
- But then there is no senior person to approve it.
Workflow realizability as invariant

- for realizable schema $\mathcal{A}_S$, let user $u$ request to do task $t$
- if $u$ is not authorized (when $(t, u)$ not in $A$), deny request
- what to do when $(t, u)$ is in $A$?
- granting this may make residual workflow unrealizable!
Authorization Enforcement Function (AEF)

\[ \text{AEF: accessRequest} \times \text{state} \to \text{decision} \times \text{state} \]

- AEF needs to enforce policy \( A \), constraints \( C \), and maintain realizability as invariant
- state is list of pairs \( ((u, t), d) \): decision \( d \) on request \( (u, t) \)
- stateful schema \( \mathcal{AS}[\sigma] \) like \( \mathcal{AS} \) except that \( A \) changes to

\[
\{(t, u) \in A \mid \forall ((t, u'), \text{grant}) \in \sigma : t.A = \{u'\}\}
\]

where we assume \( \sigma \) grants \( t \) at most once, so:
- if \( t \) granted in \( \sigma \), only granted user authorized for \( t \)
- if \( t \) not granted in \( \sigma \), authorization for \( t \) as in \( A \)
AEF with decision procedure (DP)

\[(\text{dec, state}) \ AEF-\text{DP}(\text{schema AS, state } s, \text{ accReq } (t, u))\]

\{ // @ denotes list concatenation
  if \(((t, u) \text{ in A } \&\& \text{isRealble(AS}[s @ (((u, t),\text{grant})])]))\) \{
    \text{return (grant, } s @ (((u, t),\text{grant}))\};
  \} \text{ else } \{
    \text{return (deny, } s @ (((u, t),\text{deny}))\};
  \}
\}

- returns decision and new state
- needs decision procedure \text{isRealble} for workflow realizability
- \text{AEF-\text{DP}} maintains policy \text{A}, constraints \text{C}, and realizability
- but has to call \text{isRealble} for each access request!
AEF with solution sets (SS)

\[ T = \bigcup_{i=0}^{n} T_i \quad \text{and} \quad U \supseteq \tilde{U} = \bigcup_{i=0}^{n} U_i \]

- partition task set \( T \) and subset of users \( \tilde{U} \) such that
  - every task in \( T_i \) may be assigned to some user in \( U_i \)
  - assignments are guaranteed not to interfere with each other

- compact representation of subset of realizability solutions
- compute this with LTL(F) satisfiability checking (in NuSMV)
Example partition

\[ T_0 \cup U_0 = \{ \text{approveOrder, approvePayment, Alice} \} \]
\[ T_1 \cup U_1 = \{ \text{createOrder, countersignNote, Bob} \} \]
\[ T_2 \cup U_2 = \{ \text{createPayment, signreceivedNote, Carol} \} \]

Carol and Bob are FinClerk, Alice is FinAdmin (more senior).
Pseudo-code for AEF-SS

\[
\text{dec, state) AEF-SS(schema AS, state (P,s), accReq (t,u))}
\]
\[
\begin{cases}
\quad \text{if (there is (Ti,Ui) in P: t in Ti && u in Ui)} \\
\quad \quad \text{return (grant, (P, s @ [((u,t),grant)]))};
\end{cases}
\]
\[
\begin{cases}
\quad \text{else} \\
\quad \quad \text{return (deny, (P, s @ [((u,t),deny)]))};
\end{cases}
\]
Implement $\text{AEF-DP}$ via BDDs

$$\eta_{\text{bind}} = \bigwedge_{t \in T} \bigvee_{u \in U_t} x(t,u) \quad (U_t \overset{\text{def}}{=} \{u \in U \mid (t, u) \in A\})$$

$$\eta_{C} = \bigwedge_{(D,t \rightarrow t', \rho) \in C} \eta(D,t \rightarrow t', \rho)$$

$$\eta(D,t \rightarrow t', \rho) = \bigwedge_{u \in D \cap U_t} \left( x(t,u) \rightarrow \bigwedge_{u' \in U_{t'} \setminus \rho \cdot u} \neg x(t',u') \right)$$

- BDDs may replace need for recomputing BDD from scratch in $\text{AEF-DP}$, synthesized via conjunction of formulas above
- $x(t,u)$ true: $u$ may do $t$; $x(t,u)$ false: $u$ must not execute $t$
Implementing $\text{AEF-SS}$ through model checking

- $\delta_{FT} \overset{\text{def}}{=} \bigwedge_{t \in T} F t$
- $\delta_{GU} \overset{\text{def}}{=} G(\bigvee_{u \in U} u)$
- $\delta_{FU} \overset{\text{def}}{=} \bigwedge_{u \in U} F u$
- $\delta_{A} \overset{\text{def}}{=} \bigwedge_{t \in T} G \left( t \rightarrow \neg(\bigvee_{(t,u) \not\in A} u) \right)$
- $\delta_{C} \overset{\text{def}}{=} \bigwedge_{(D,t \rightarrow t',\rho) \in C} \delta(D,t \rightarrow t',\rho)$
- $\delta_{(D,t \rightarrow t',\rho)} \overset{\text{def}}{=} \bigwedge_{u \in D} (F(t \land u)) \rightarrow G \left( t' \rightarrow \neg(\bigvee_{(u,u') \not\in \rho} u') \right)$
- $\delta_{AS} \overset{\text{def}}{=} \delta_{FT} \land \delta_{GU} \land \delta_{A} \land \delta_{C} \land \delta_{FU}$

- Bounded model checking of $\neg \delta_{AS}$ on fully connected graph
- Counterexample yields partition, a state captures a $T_i \cup U_j$. 

Jason Crampton, Michael Huth, and Jim Huan-Pu Kuo
Authorization Enforcement in Workflows
Experimental results

- Authorization density $ad$ is $|A| / (|U| \cdot |T|)$
- Constraint density $cd$ is $|C| / |U|$
- Studied runtimes of BDD and LTL(F) approaches for three types of authorization density (low, medium, high)
- For each such type varied configurations $uu-cd$ where $uu$ is $|U| = |T|$
BDDs vs. LTL model checking times ($a_d = 0.1$)
BDDs vs. LTL model checking times ($\alpha = 0.5$)
BDDs vs. LTL model checking times ($\alpha = 1.0$)

Running time (s) vs Configurations

- **BDD**
- **LTL**
Conclusion

- motivated need for workflow management systems to maintain “security-aware” realizability
- proposed two authorization enforcement functions that ensure such realizability
- one of them needs a decision procedure that has to be called for each request to execute a task
- circumventing this need through the synthesis and update of a BDD seemed to fail experimentally
- the other authorization enforcement function computes a solution set and seems to scale better
- we now want to improve that scalability and increase flexibility of the solution set (e.g. through delegation)
Thank You for Your Kind Attention

Questions?