Service Interaction Contracts as Security Policies

Davide Basile

Computer Science Department
University of Pisa

Outline

Topic: SOC paradigm, Service Orchestrations

- Behavioral Contracts: Server-Client interactions
- Security: Resource Access Control

Goal: a framework that naturally deals with compliance of behavioral contracts, access control and multi-party.
Novelty: model checking techniques for ensure compliance of behavioral contracts.

Interactions

Behavioral Contracts as processes of CCS , built with three operators:

- prefix $a.\sigma$
- internal choice $\sigma_1 \oplus \sigma_2$
- external choice $\sigma_1 + \sigma_2$

Compliance

A service complies (written $\dashv$) with a Client if successfully terminates the interactions.

Example:

- $a \oplus \overline{b} \dashv \overline{a} + b$
- $a \oplus \overline{b} \dashv \overline{a} + b + c$
- $a \dashv \overline{a} + b$
- $a \not\dashv \overline{a} \oplus b$

Other features: subcontract relation

Resource Access Control

- Services and clients described in a functional language λ^{req} .
- History Expression H represents the abstract behavior of the service.
- There is a *Type and Effect System* that compute the abstract behavior of a λ^{req} expression.

Call-by-Contract

- Access event α abstracts from a security critical operation, and is logged into a history η.
- The security policies φ are regular properties of execution histories, i.e. "never perform write actions after read actions".
- Services are invoked by request with policy φ.

Call-by-Contract

- An orchestration machinery binds each request to a service which satisfies the requested type and the safety properties.
- Model-Checking techniques are developed for ensures that the above constraints are fulfilled.

Our work: intended benefit

- We extends History Expressions to include channel communications and internal/external choice for combining the notions of security access and progress of interactions, so merging and enriching the above surveyed approaches.
- We prove that compliance between client and server is a safety property.
- We use standard techniques of model checking for controlling compliance of behavioural contracts. Also we manage both multi-party contracts and sessions.

Working Example

$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\ \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$

$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$

$H_4 = \mathtt{open}_{1,\varphi_1}\ \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\ \overline{a}.g\ \mathtt{close}_3$

$\varphi_2 =$ "never β after α"

$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$

$\{\ell_1 : H_1, \ell_2 : H_2, \ell_3 : H_3\}^? \cup \{\}^! \triangleright$

$\ell_4 : H_4 \| \ell_5 : H_5$

Working Example

$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\ \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$

$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$

$H_4 = \mathtt{open}_{1,\varphi_1}\ \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\ \overline{a}.g\ \mathtt{close}_3$

$\varphi_2 =$ "never β after α"

$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$

$\{\ell_2 : H_2, \ell_3 : H_3\}^? \cup \{\ell_1 : H_1\}^! \triangleright$

$[\ell_4 : \overline{a}.(b + c)\ \mathtt{close}_1, \ell_1 : \varphi_1(\!|H_1|\!)] \ \|\ \ell_5 : H_5$

Working Example

$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\ \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$

$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$

$H_4 = \mathtt{open}_{1,\varphi_1}\ \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\ \overline{a}.g\ \mathtt{close}_3$

$\varphi_2 =$ "never β after α"

$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$

$\{\ell_2 : H_2\}^? \cup \{\ell_1 : H_1, \ell_3 : H_3\}^! \triangleright$

$[\ell_4 : \ldots, \ell_1 : \ldots] \| [\ell_5 : \overline{a}.g\ \mathtt{close}_3, \ell_3 : \varphi_3(\!|H_3|\!)]$

Working Example

$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\ \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$

$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$

$H_4 = \mathtt{open}_{1,\varphi_1}\ \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\ \overline{a}.g\ \mathtt{close}_3$

$\varphi_2 =$ "never β after α"

$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$

$\{\ell_2 : H_2, \ell_3 : H_3\}^? \cup \{\ell_1 : H_1\}^! \triangleright$

$[\ell_4 : (b + c)\ \mathtt{close}_1, \ell_1 : \mathtt{open}_{2,\varphi_2} \ldots]$

Working Example

$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\ \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$

$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$

$H_4 = \mathtt{open}_{1,\varphi_1}\ \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\ \overline{a}.g\ \mathtt{close}_3$

$\varphi_2 =$ "never β after α"

$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$

$\{\ell_3 : H_3\}^? \cup \{\ell_1 : H_1, \ell_2 : H_2\}^! \triangleright$

$[\ell_4 : \ldots, [\ell_1 : \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d, \ell_2 : \varphi_2(\!|H_2|\!)]]$

Working Example

$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\ \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$

$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$

$H_4 = \mathtt{open}_{1,\varphi_1}\ \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\ \overline{a}.g\ \mathtt{close}_3$

$\varphi_2 =$ "never β after α"

$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$

$\{\ell_3 : H_3, \ell_2 : H_2\}^? \cup \{\ell_1 : H_1\}^! \triangleright$

$[\ell_4 : (b + c)\ \mathtt{close}_1, \ell_1 : (\overline{b} \oplus \overline{c}) \cdot d]$

Working Example

$$H_1 = a \cdot (\mathtt{open}_{2,\varphi_2}\, \overline{d}.(e + f)\ \mathtt{close}_2) \cdot (\overline{b} \oplus \overline{c}) \cdot d$$
$$H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha \qquad H_3 = a.\overline{g}$$
$$H_4 = \mathtt{open}_{1,\varphi_1}\, \overline{a}.(b + c)\ \mathtt{close}_1 \qquad H_5 = \mathtt{open}_{3,\varphi_3}\, \overline{a}.g\ \mathtt{close}_3$$
$$\varphi_2 = \text{“never } \beta \text{ after } \alpha\text{”}$$
$$\pi = \bigcup_{i \in \{1,2,3\}} (r_i, \ell_i)$$

$$\{\ell_1 : H_1, \ell_2 : H_2, \ell_3 : H_3\}^? \cup \{\}^! \rhd \varepsilon$$

Planning and Verifying

- for each request find compliant services
- check if the selected service respects φ
- check if the selected plan is valid (no conflicts between requests)

Model Checking Compliance

Given $\texttt{open}_{r,\varphi} H_1 \texttt{close}_r$ and H_2

- Projections $H^\flat$: remove policies access events and nested requests.
- FSA : $H_1^\flat \otimes H_2^\flat$
- For each state check:
  - client has not terminate: $H_1 \neq \varepsilon$
  - $\forall$ output action $\exists$ input action
- Compliance is a safety property

$\texttt{open}_{r,\varphi} H_1 \texttt{close}_r \vdash H_2$ iff $L(H_1^\flat \otimes H_2^\flat) = \emptyset$.

$H \models \varphi$ iff $[\![H]\!] \cap L(\mathcal{A}_\varphi) = \emptyset$.

Verifying Compliance and Safety

- compliance:
  $H_1 = \mathtt{open}_{2,\varphi_2}\, \overline{d}.(e + f)\ \mathtt{close}_2 \qquad H_2 = \beta \cdot d.(\overline{e} \oplus \overline{f}) \cdot \alpha$
  $H_2^\flat = d.(\overline{e} \oplus \overline{f})$
  $\mathcal{A} = \overline{d}.(e + f) \otimes d.(\overline{e} \oplus \overline{f})$
  $\mathsf{S} = \{\langle \overline{d}.(e + f), d.(\overline{e} \oplus \overline{f})\rangle, \langle e + f, \overline{e} \oplus \overline{f}\rangle, \langle \varepsilon, \varepsilon\rangle\}$
  $F = \emptyset$

- safety:
  $\varphi_2 =$ "never β after α" $\qquad H_2 = \beta \cdot \alpha$
  $[\![\varphi_2]\!] = \{\Sigma^* \alpha \Sigma^* \beta \Sigma^*\} \qquad [\![\beta \cdot \alpha]\!] = \{\beta \cdot \alpha\}$
  $[\![\varphi_2]\!] \cap [\![\beta \cdot \alpha]\!] = \emptyset$

- $(r_2, \ell_2) \in \pi$

Planning

The third last step ensures that a service never gets stuck while is waiting to opening a session

Example:
$\{\ell_2 : \overline{a}.(b + c + d + e)\}^? \triangleright$
$\ell_1 : \mathtt{open}_{1,\_}\, a.\mathtt{open}_{2,\_}\, a.(\overline{b} \oplus \overline{c})\mathtt{close}_2(\overline{b} \oplus \overline{c} \oplus \overline{d})\mathtt{close}_1$

$a.(\overline{b} \oplus \overline{c} \oplus \overline{d}) \vdash \overline{a}.(b + c + d + e)$
$a.(\overline{b} \oplus \overline{c}) \vdash \overline{a}.(b + c + d + e)$

$\pi = \{(r_1, \ell_2), (r_2, \ell_2)\}$

$\{\}^? \cup \{\ell_2 : H_2\}^! \triangleright [\ell_1 : \mathtt{open}_{2,\_} \cdots, \ell_2 : b + c + d + e] \nrightarrow$

Local Planning

Local Plan: a plan valid for a local service.

Steps for generate a local plan for a service:

- pick the outermost open/close term;
- select a service which is compliant with that request;
- remove the selected service from the set of compliant services of the nested open/close subterms.

Several local plans are possible for a single service.

Global Plan

Global plan: a plan valid for the whole network.

Steps for generate a global plan:

- merge the local plans
- techniques of model-checking for validating the global plan generated

Conclusion

- Formalism for expressing:
  - services
  - interactions between services and clients
  - multi-party sessions
- Security:
  - policy for resource access and control
  - progress of interactions

Future work: implementing the algorithms outlined above in one of the existing model-checker.