

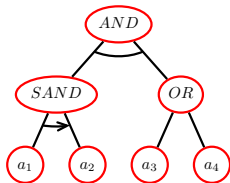
# Deciding the Emptiness of Attack trees

Maxime Audinot, Sophie Pinchinat, François Schwarzentruher,  
Florence Wacheux

Univ. Rennes, IRISA, CNRS

GraMSec 2018

# Emptiness of attack trees: Relevance



Attack trees always seem non-empty:

$a_1 a_3 a_2$  is an attack

But is the attack  $a_1 a_3 a_2$  realizable?

Maybe executing action  $a_1$  consumes the resource required to execute  $a_3$ .

We need a model  $\mathcal{S}$  of the system.



**Non-emptiness of an attack tree w.r.t. a system  $\mathcal{S}$**

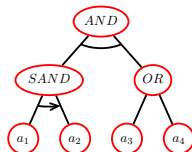
Is there an attack described by the tree that is realizable in  $\mathcal{S}$ ?

# Outline

- 1 Formal Setting
- 2 The non-emptiness problem
- 3 The non-emptiness problem for AND-free attack trees
- 4 Conclusion

# Attack trees

Action-based trees  
(classical)

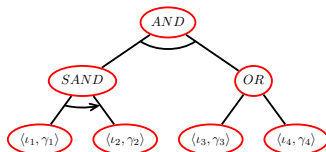


State-based trees

- more recent
- homogeneous formalism
- a system-based semantics [APK17]

## Definition (Attack tree)

$\tau ::= \langle \iota, \gamma \rangle \mid \text{OR}(\tau, \tau) \mid \text{SAND}(\tau, \tau) \mid \text{AND}(\tau, \tau) \mid \text{AND}(\tau, \tau, \tau) \mid \text{AND}(\tau, \tau, \tau, \tau) \mid \dots$   
 where  $\iota, \gamma \in \text{Prop}$ .



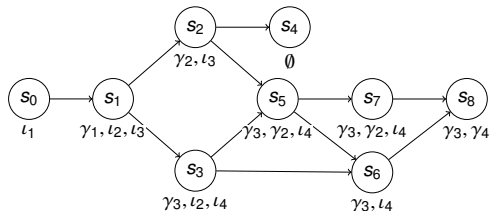
# Model of the system

## Definition (Labeled transition system)

A labeled transition system over  $Prop$  is a tuple  $\mathcal{S} = (S, \rightarrow, \lambda)$ , where

- $S$  finite set of states
  - $\lambda : S \rightarrow 2^{Prop}$  labeling function
  - $\rightarrow \subseteq S \times S$  transition relation
- Write  $s \models p$  whenever  $p \in \lambda(s)$ .

$Prop = \{\iota_1, \iota_2, \iota_3, \iota_4, \gamma_1, \gamma_2, \gamma_3, \gamma_4\}$



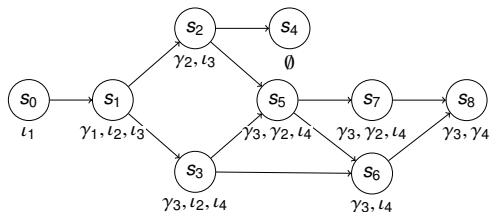
$\lambda(s_0) = \{\iota_1\}$ ,  $\lambda(s_1) = \{\gamma_1, \iota_2, \iota_3\}$ ,  $\lambda(s_2) = \{\gamma_2, \iota_3\} \dots$

# Paths in a label transition system

## Definition (Paths)

A path in  $\mathcal{S}$  is a sequence  $\pi = s_0 \dots s_n$  of consecutive states in  $\mathcal{S}$ .  
 $\pi.first := s_0$  and  $\pi.last := s_n$

Write  $Paths(\mathcal{S})$  for the set of paths in  $\mathcal{S}$ .



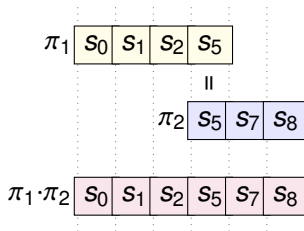
$$\pi = s_0 s_1 s_3 s_5 s_6 s_8$$

$$\pi.first = s_0$$

$$\pi.last = s_8$$

# Paths concatenation

Concatenation of paths  $\pi_1$  and  $\pi_2$  possible whenever  $\pi_1.\text{last} = \pi_2.\text{first}$

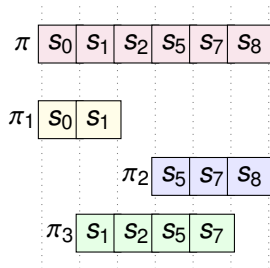


## Notation

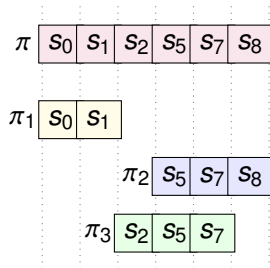
For  $\Pi_1, \Pi_2 \subseteq \text{Paths}(S)$ , define  $\Pi_1 \cdot \Pi_2 := \{\pi_1 \cdot \pi_2 \mid \pi_1 \in \Pi_1 \text{ and } \pi_2 \in \Pi_2\}$ .

# Path parallel composition

## Informal



(a) A parallel composition.



(b) Not a parallel composition.

## Notation

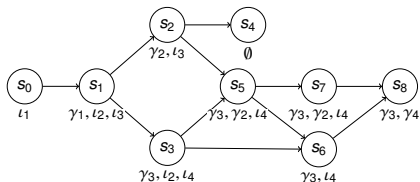
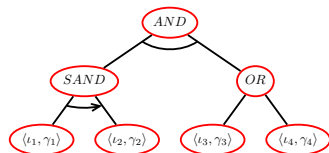
For  $\Pi_1, \dots, \Pi_n \subseteq \text{Paths}(\mathcal{S})$ , define  $\otimes_n(\Pi_1, \Pi_2, \dots, \Pi_n)$  by  $\{\pi \mid \pi \text{ is a parallel composition of some } \pi_1 \in \Pi_1, \dots, \pi_n \in \Pi_n\}$ .



# Path semantics of an attack tree [APK17]

## Definition ( $\llbracket \tau \rrbracket^{\mathcal{S}} \subseteq \text{Paths}(\mathcal{S})$ )

- $\llbracket \langle \iota, \gamma \rangle \rrbracket^{\mathcal{S}} = \{ \pi \in \text{Paths}(\mathcal{S}) \mid \pi.\text{first} \models \iota \text{ and } \pi.\text{last} \models \gamma \}$
- $\llbracket \text{OR}(\tau_1, \tau_2) \rrbracket^{\mathcal{S}} = \llbracket \tau_1 \rrbracket^{\mathcal{S}} \cup \llbracket \tau_2 \rrbracket^{\mathcal{S}}$
- $\llbracket \text{SAND}(\tau_1, \tau_2) \rrbracket^{\mathcal{S}} = \llbracket \tau_1 \rrbracket^{\mathcal{S}} \cdot \llbracket \tau_2 \rrbracket^{\mathcal{S}}$
- $\llbracket \text{AND}(\tau_1, \dots, \tau_n) \rrbracket^{\mathcal{S}} = \otimes_n (\llbracket \tau_1 \rrbracket^{\mathcal{S}}, \dots, \llbracket \tau_n \rrbracket^{\mathcal{S}})$



$$\llbracket \tau \rrbracket^{\mathcal{S}} = \{ s_0 s_1 s_2 s_5, s_0 s_1 s_2 s_6, s_0 s_1 s_3 s_6 s_8, \dots \}$$

# Outline

- 1 Formal Setting
- 2 The non-emptiness problem**
- 3 The non-emptiness problem for AND-free attack trees
- 4 Conclusion

# The decision problem NON-EMPTYNESS

## Definition (NON-EMPTYNESS)

Input: a system  $\mathcal{S}$  and an attack tree  $\tau$ .

Output:  $\llbracket \tau \rrbracket^{\mathcal{S}} \neq \emptyset?$

## Theorem

NON-EMPTYNESS is NP-complete.

Proof:

- NON-EMPTYNESS is NP-easy:
  - compute an abstraction of the path semantics
  - guess and check a paths corresponding with the abstract semantics
- NON-EMPTYNESS is NP-hard: Reduction of SAT (NP-complete by [Coo71])

# NON-EMPTINESS is NP-easy

## Abstracting the path semantics

Recall the path semantics:

$$\llbracket \langle \iota, \gamma \rangle \rrbracket^{\mathcal{S}} = \{ \pi \in \text{Paths}(\mathcal{S}) \mid \pi.\text{first} \models \iota \text{ and } \pi.\text{last} \models \gamma \}$$

The path semantics is not adequate for “computation”, as any cycle in  $\mathcal{S}$  yields infinitely many paths.

The *abstract semantics* retains only end-states of paths in  $\llbracket \langle \iota, \gamma \rangle \rrbracket^{\mathcal{S}}$ .

$$\llbracket \langle \iota, \gamma \rangle \rrbracket_{\text{abs}}^{\mathcal{S}} = \{ s_1 s_2 \mid s_1 \models \iota \text{ and } s_2 \models \gamma \}$$

# NON-EMPTYNESS is NP-easy

## Abstract semantics

### Definition ( $\llbracket \tau \rrbracket_{\text{abs}}^S \subseteq S^*$ )

- $\llbracket \langle \iota, \gamma \rangle \rrbracket_{\text{abs}}^S = \{s_1 s_2 \mid s_1 \models \iota \text{ and } s_2 \models \gamma\}$
- $\llbracket \text{OR}(\tau_1, \tau_2) \rrbracket_{\text{abs}}^S = \llbracket \tau_1 \rrbracket_{\text{abs}}^S \cup \llbracket \tau_2 \rrbracket_{\text{abs}}^S$
- $\llbracket \text{SAND}(\tau_1, \tau_2) \rrbracket_{\text{abs}}^S = \llbracket \tau_1 \rrbracket_{\text{abs}}^S \cdot \llbracket \tau_2 \rrbracket_{\text{abs}}^S$
- How about AND?

### Vocabulary

Call *word* any sequence  $u \in S^*$  (that may not be a path).

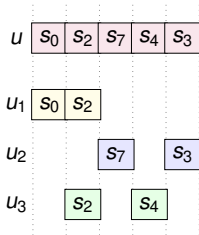
# NON-EMPTINESS is NP-easy

## Abstract semantics for AND

### Definition (Shuffle)

$u$  shuffles  $u_1, \dots, u_n$  whenever:

- (Linearization)  $u$  is composed of all the states occurring in  $u_1, \dots, u_n$  with preserved precedence.
- (Covering) Every sequence of two consecutive states in  $u$  is between the occurrence of  $u_j.first$  and  $u_j.last$  for some  $j$ .



### Definition

$$\llbracket \text{AND}(\tau_1, \dots, \tau_n) \rrbracket_{\text{abs}}^S = \{u \mid u \text{ shuffles some } u_1 \in \llbracket \tau_1 \rrbracket_{\text{abs}}^S, \dots, u_n \in \llbracket \tau_n \rrbracket_{\text{abs}}^S\}.$$

# Algorithm for Non-EMPTINESS

Guess  $u \in \llbracket \tau \rrbracket_{\text{abs}}^S$

**Input:** An attack tree  $\tau$  and a transition system  $S$

**Output:** A word  $u \in \llbracket \tau \rrbracket_{\text{abs}}^S$

switch  $\tau$  do

  case  $\langle \iota, \gamma \rangle$  do

    guess  $s_1, s_2 \in S$ ;

    check  $\iota \in \lambda(s_1)$  and  $\gamma \in \lambda(s_2)$ ;

    return  $s_1 s_2$ ;

  end

  case  $OR(\tau_1, \tau_2)$  do

    guess  $i \in \{1, 2\}$ ;

    return guessAbstractPath( $\tau_i, S$ );

  end

  case  $SAND(\tau_1, \tau_2)$  do

$u_1 :=$  guessAbstractPath( $\tau_1, S$ );

$u_2 :=$  guessAbstractPath( $\tau_2, S$ );

    check  $u_1.last = u_2.first$ ;

    return  $u_1 \cdot u_2$

  end

  case  $AND(\tau_1, \dots, \tau_n)$  do

$u_i :=$  guessAbstractPath( $\tau_i, S$ ) for each  $1 \leq i \leq n$ ;

    guess  $u$ , a linearization of  $u_1, \dots, u_n$ ;

    forall letters  $s$  of  $u$  except  $u.first$  and  $u.last$  do

      check there exists  $j, k \in [1, n]$  such that either  $s$  is strictly between  $u_j.first$   
and  $u_j.last$  in  $u$ , or  $s$  equals both  $u_j.first$  and  $u_k.last$

    end

    return  $u$ ;

  end

end

**Algorithm 1:** guessAbstractPath( $\tau, S$ ).

# Algorithm for NON-EMPTINESS

Check that  $u$  can instantiate some path

**Input:** An attack tree  $\tau$  and a transition system  $\mathcal{S}$   
**Output:** Accept whenever  $\llbracket \tau \rrbracket^{\mathcal{S}} \neq \emptyset$ .  
 $u := \text{guessAbstractPath}(\tau, \mathcal{S})$ ;  
**foreach**  $s_1, s_2$  successive in  $u$  **do**  
  | **check**  $\text{reach}_{\mathcal{S}}(s_1, s_2)$   
**end**  
**accept**

**Algorithm 2:**  $\text{emptiness}(\tau, \mathcal{S})$ .

## Theorem

NON-EMPTINESS is NP-complete.



## NON-EMPTINESS is NP-hard

NP-hardness arise already for very simple attack trees of the form  
 $\text{AND}(\langle \iota_1, \gamma_1 \rangle, \dots, \langle \iota_n, \gamma_n \rangle)$ :

### Proposition (From [APK17])

Given a system  $S$  and  $\iota_1, \gamma_1, \dots, \iota_n, \gamma_n \in \text{Prop}$ , it is NP-hard to decide  
 $\llbracket \text{AND}(\langle \iota_1, \gamma_1 \rangle, \dots, \langle \iota_n, \gamma_n \rangle) \rrbracket^S \neq \emptyset$ .

Proof: By reduction of SAT (NP-complete by [Coo71]).

### SAT problem

Input:  $C_1, \dots, C_k$  clauses over Boolean variables  $p, q, r, \dots$

Output: is there a valuation of  $p, q, r, \dots$  that satisfies  $C_1, \dots, C_k$ ?

We define a polynomial translation from SAT inputs to NON-EMPTINESS inputs.

# NP-hardness of $\llbracket \text{AND}(\langle \iota_1, \gamma_1 \rangle, \dots, \langle \iota_n, \gamma_n \rangle) \rrbracket^S \neq \emptyset$ ?

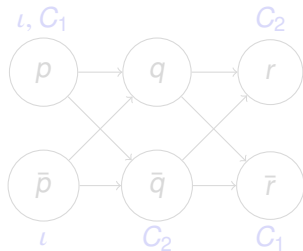
## SAT problem

Input:  $C_1, \dots, C_k$  clauses over Boolean variables  $p, q, r, \dots$

Output: is there a valuation of  $p, q, r, \dots$  that satisfies  $C_1, \dots, C_k$ ?

Reduction of SAT: From input  $C_1, \dots, C_k$  over  $p, q, r, \dots$  of SAT, define system  $\mathcal{S}$  (of polynomial size) over  $\text{Prop} = \{\iota, C_1, \dots, C_k\}$  s.t.

$$C_1, \dots, C_k \in \text{SAT} \text{ iff } \llbracket \text{AND}(\langle \iota, C_1 \rangle, \dots, \langle \iota, C_k \rangle) \rrbracket^S \neq \emptyset$$



$$C_1 = p \vee \bar{r}$$

$$C_2 = \bar{q} \vee r$$

$$\llbracket \text{AND}(\langle \iota, C_1 \rangle, \langle \iota, C_2 \rangle) \rrbracket^S = \{pqr, p\bar{q}, \dots\}$$

# NP-hardness of $\llbracket \text{AND}(\langle \iota_1, \gamma_1 \rangle, \dots, \langle \iota_n, \gamma_n \rangle) \rrbracket^S \neq \emptyset$ ?

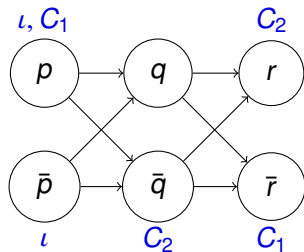
## SAT problem

Input:  $C_1, \dots, C_k$  clauses over Boolean variables  $p, q, r, \dots$

Output: is there a valuation of  $p, q, r, \dots$  that satisfies  $C_1, \dots, C_k$ ?

Reduction of SAT: From input  $C_1, \dots, C_k$  over  $p, q, r, \dots$  of SAT, define system  $S$  (of polynomial size) over  $\text{Prop} = \{\iota, C_1, \dots, C_k\}$  s.t.

$$C_1, \dots, C_k \in \text{SAT} \text{ iff } \llbracket \text{AND}(\langle \iota, C_1 \rangle, \dots, \langle \iota, C_k \rangle) \rrbracket^S \neq \emptyset$$



$$C_1 = p \vee \bar{r}$$

$$C_2 = \bar{q} \vee r$$

$$\llbracket \text{AND}(\langle \iota, C_1 \rangle, \langle \iota, C_2 \rangle) \rrbracket^S = \{pqr, p\bar{q}, \dots\}$$

# Outline

- 1 Formal Setting
- 2 The non-emptiness problem
- 3 The non-emptiness problem for AND-free attack trees**
- 4 Conclusion

## The sub-polynomial AND-free case

### Definition ( $\text{NON-EMPTINESS}_{\text{ANDfree}}$ )

Input: a system  $\mathcal{S}$  and an AND-free attack tree  $\tau$

Output:  $[[\tau]]^{\mathcal{S}} \neq \emptyset?$

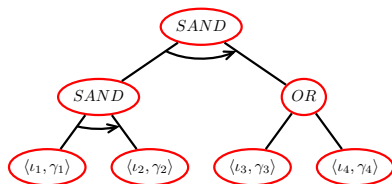
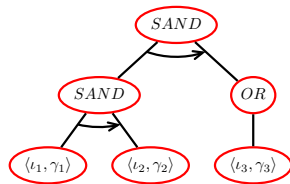
### Theorem

$\text{NON-EMPTINESS}_{\text{ANDfree}}$  is NLOGSPACE-complete.

Proof:

- $\text{NON-EMPTINESS}_{\text{ANDfree}}$  is NLOGSPACE-hard.  
Trivial logspace reduction from the  $s-t$ -connectivity in a graph (NLOGSPACE-complete by [Jon75]) to the non-emptiness of a leaf attack tree  $\langle \iota, \gamma \rangle$ .
- $\text{NON-EMPTINESS}_{\text{ANDfree}}$  is NLOGSPACE-easy

# NON-EMPTYNESS<sub>ANDfree</sub> is NLOGSPACE-easy

 $\tau$ An OR-resolution of  $\tau$ 

## Proposition

$\pi \in \llbracket \tau \rrbracket^S$  iff there exists an OR-resolution  $\tau'$  of  $\tau$  s.t.  $\pi \in \llbracket \tau' \rrbracket^S$

Guess simultaneously  $\tau'$  and  $\pi \in \llbracket \tau' \rrbracket^S$  during a depth-first search of  $\tau$ .  
Require to store (logarithmic memory):

- 1 node of  $\tau$
- 2 states of  $S$

# Algorithm for NON-EMPTINESS<sub>ANDfree</sub>

**Input:** An AND-free attack tree  $\tau$  and a transition system  $S$

**Output:** Accept whenever  $\llbracket \tau \rrbracket^S \neq \emptyset$ .

**guess**  $s \in S$ ;

$node := \text{root of } \tau$ ;

$lastOp := \text{down}$ ;

**repeat**

**if**  $node = \langle \iota, \gamma \rangle$  **then**

**check**  $s \models \iota$ ;

**loop**

**guess** whether we break the loop or not; if yes, **break** the loop;

**guess**  $s' \in S$  with  $s \rightarrow s'$ ;

$s := s'$

**endLoop**

**check**  $s \models \gamma$ ;

**end**

**if** ( $lastOp = \text{down}$ ) or ( $lastOp = \text{over}$ ) **then**

    Try to perform and update  $node$  with operation  $down$ ,  $over$ ,  $up$  in priority;

    Store in  $lastOp$  the last performed operation

**else**

    Try to perform and update  $node$  with operation  $over$ ,  $up$  in priority;

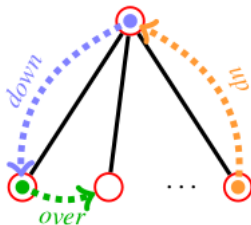
    Store in  $lastOp$  the last performed operation

**end**

**until** ( $node = \text{root of } \tau$ ) and ( $lastOp = \text{up}$ );

**accept**

**Algorithm 5:**  $\text{emptiness}_{NL_{ANDfree}}(\tau, S)$ .



# Outline

- 1 Formal Setting
- 2 The non-emptiness problem
- 3 The non-emptiness problem for AND-free attack trees
- 4 Conclusion



# Conclusion

- Achievements

- Deciding  $\llbracket \tau \rrbracket^S \neq \emptyset$  is NP-complete.
- Deciding  $\llbracket \tau \rrbracket^S \neq \emptyset$  for an AND-free attack tree is NLOGSPACE-complete.
- AND is a really complex operator!

- Future work

- Non-emptiness of action-based attack trees.  
Our results should still hold.
- Other decision problems, e.g.

$$\llbracket \tau_1 \rrbracket^S = \llbracket \tau_2 \rrbracket^S?$$



Maxime Audinot, Sophie Pinchinat, and Barbara Kordy.

Is my attack tree correct?

In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 83–102, Cham, 2017. Springer International Publishing.



Stephen A Cook.

The complexity of theorem-proving procedures.

In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.



Neil D Jones.

Space-bounded reducibility among combinatorial problems.

*Journal of Computer and System Sciences*, 11(1):68–85, 1975.