UNIVERSITY OF TWENTE.

Efficient Algorithms for Quantitative Attack Tree Analysis

Carlos E. Budde[†] & Mariëlle Stoelinga[†*]

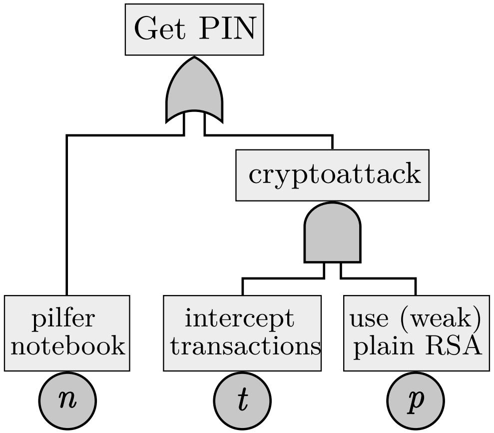
[†] Formal Methods & Tools, University of Twente, Enschede, The Netherlands

[*] Dept. of Software Science, Radboud University, Nijmegen, The Netherlands

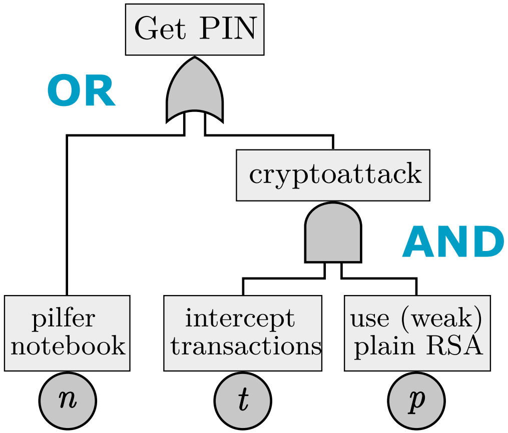
c.e.budde@utwente.nl
m.i.a.stoelinga@utwente.nl

CSF — June 23, 2021

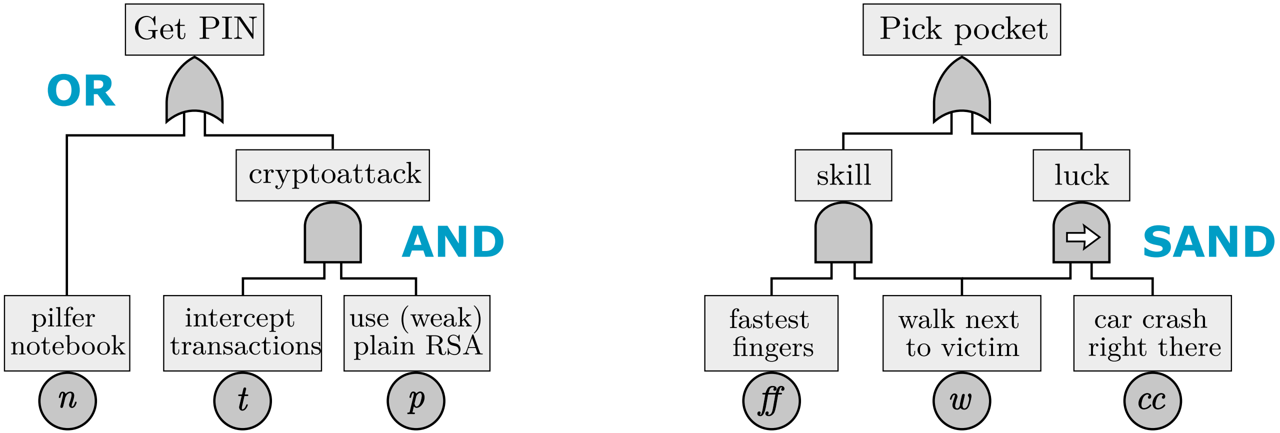
Attack Tree models



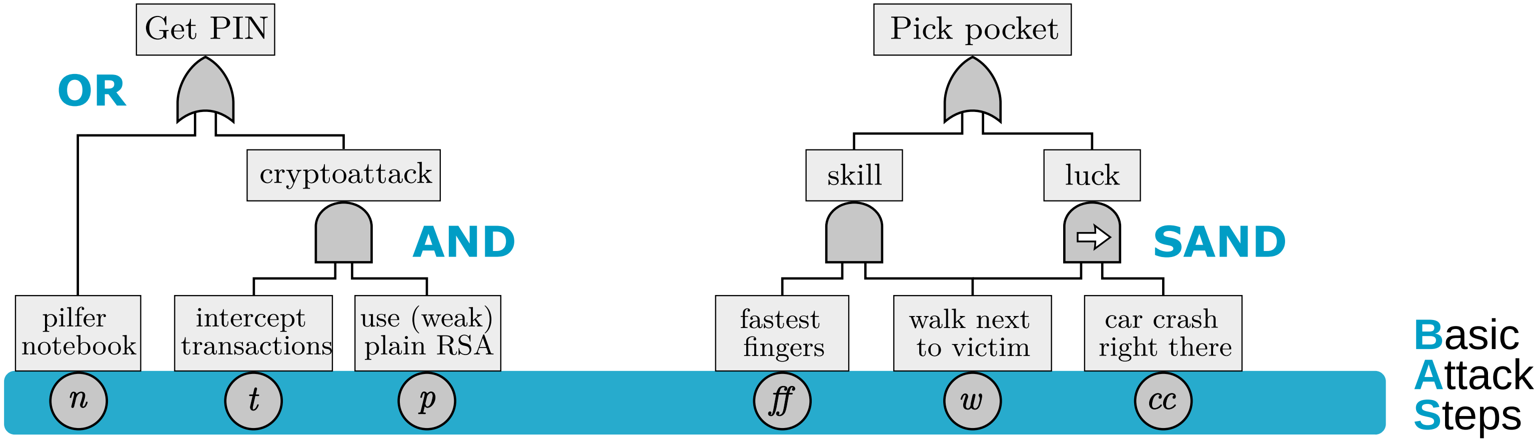
Attack Tree models

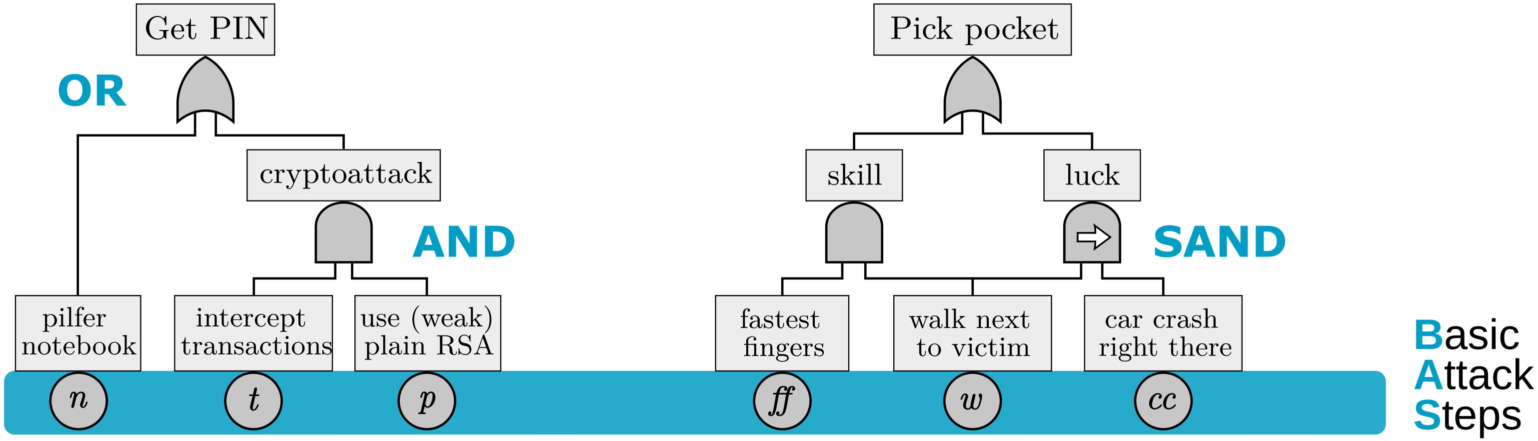


Attack Tree models



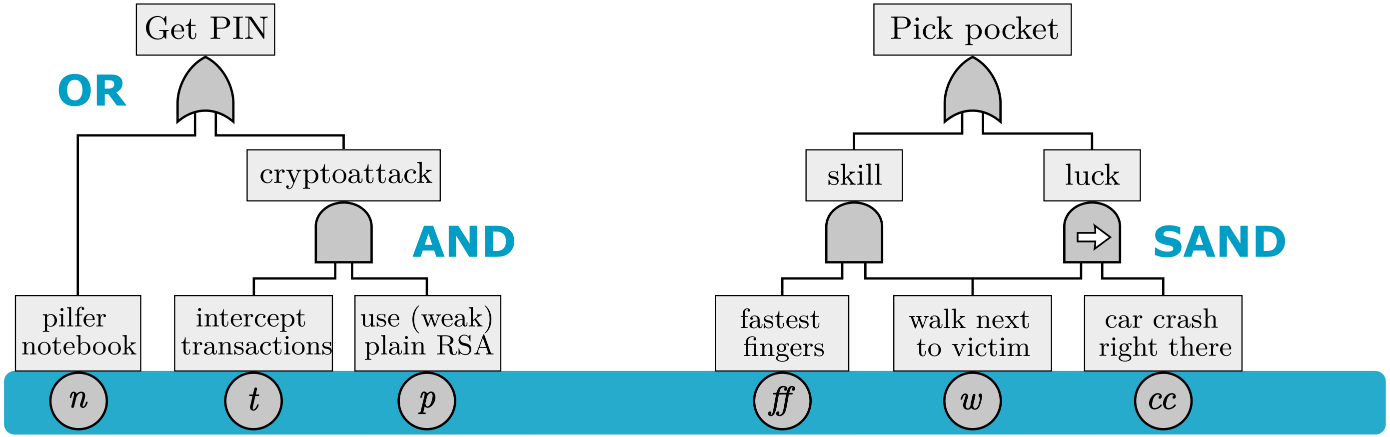
Attack Tree models





	n	t	p	ff	w	cc
Time	120	40	0	0.1	120	1
Cost	0	30	0	0	0	0
Prob.	0.07	0.01	0.95	0.001	0.6	0.05

Attack Tree models & metrics



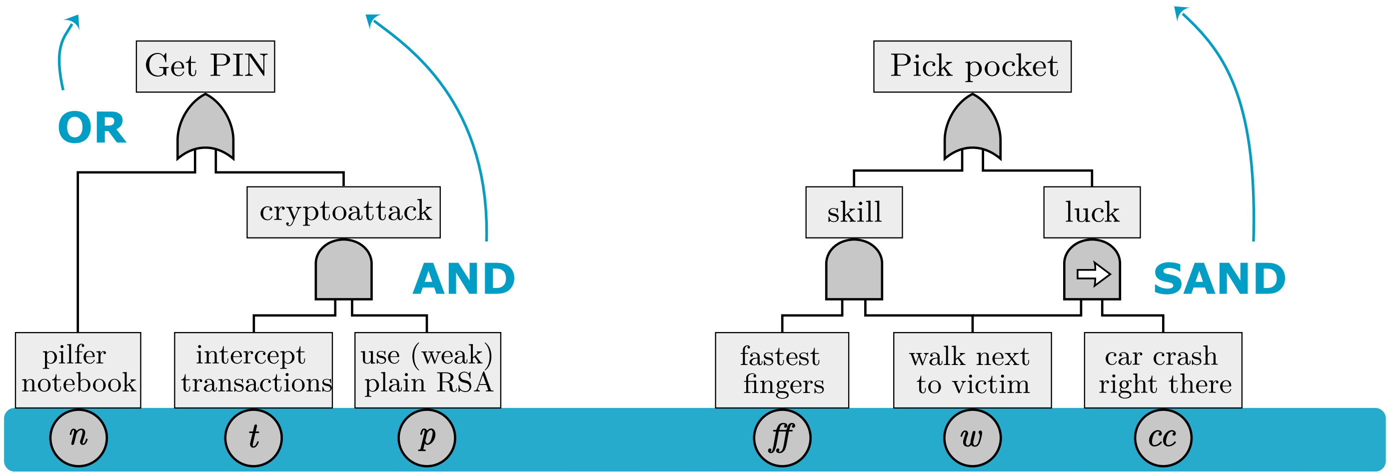
Basic
Attack
Steps

PIN								Pick
40	120	40	0	Time	0.1	120	1	120
0	0	30	0	Cost	0	0	0	0
0.07	0.07	0.01	0.95	Prob.	0.001	0.6	0.05	0.03

Attack Tree models & metrics

Static AT

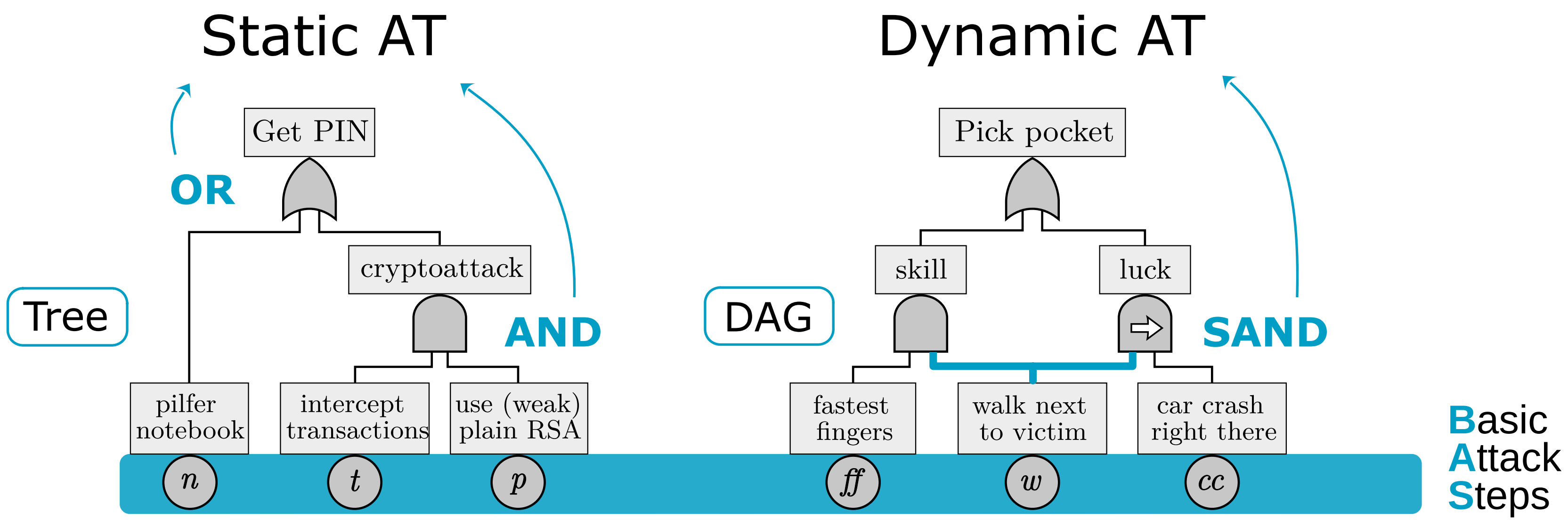
Dynamic AT



Basic **A**ttack **S**teps

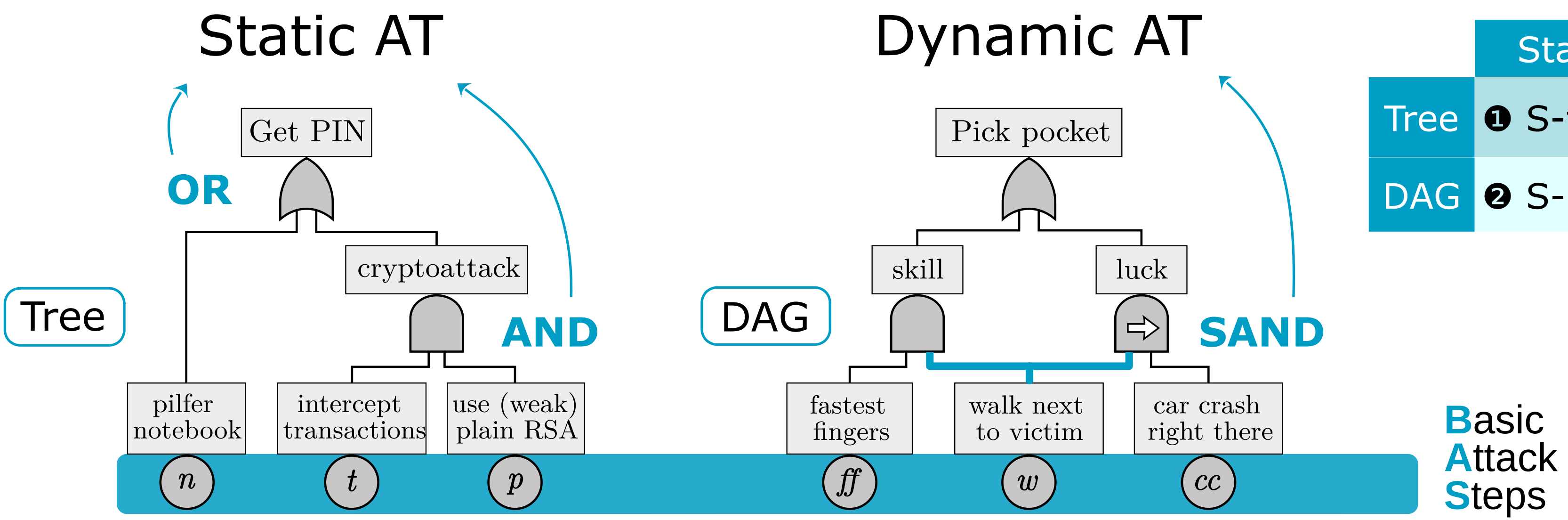
40	120	40	0	Time	0.1	120	1	120
0	0	30	0	Cost	0	0	0	0
0.07	0.07	0.01	0.95	Prob.	0.001	0.6	0.05	0.03
PIN								Pick

Attack Tree models & metrics



40	120	40	0	Time	0.1	120	1	120
0	0	30	0	Cost	0	0	0	0
0.07	0.07	0.01	0.95	Prob.	0.001	0.6	0.05	0.03
PIN								Pick

Attack Tree models & metrics



	Static	Dynamic
Tree	❶ S-tree	❸ D-tree
DAG	❷ S-DAG	❹ D-DAG

PIN								Pick
40	120	40	0	Time	0.1	120	1	120
0	0	30	0	Cost	0	0	0	0
0.07	0.07	0.01	0.95	Prob.	0.001	0.6	0.05	0.03

Algorithms to compute metrics

	❶	❷	❸	❹
Metric	Static tree	Static DAG	Dynamic tree	Dynamic DAG
min cost	BU [14, 15, 16]	MTBDD [17] $\mathcal{C}$-BU [18]	BU [4]	PTA [8]
min time	BU [14, 19]	Petri nets [12]	APH [9] BU [4]	PTA [8]
min skill	BU [14, 20]	$\mathcal{C}$-BU [18]	BU [4]	—
max damage	BU [14, 19, 20]	MTBDD [17] DPLL [7]	BU [4]	PTA [8]
probability	BU [6, 19]	BDD [21] DPLL [7]	APH [9]	I/O-IMC [5]
Pareto fronts	BU [22, 19]	$\mathcal{C}$-BU [11]	OPEN PROBLEM	PTA [8]
Any of the above	**Algo. 1:** $\mathtt{BU_{SAT}}$	**Algo. 2:** $\mathtt{BDD_{DAG}}$	**Algo. 5:** $\mathtt{BU_{DAT}}$	OPEN PROBLEM
***k*-top metrics**	BU-projection [14]	**Algo. 3:** `BDD shortest_paths`	OPEN PROBLEM	OPEN PROBLEM

	Static	Dynamic
Tree	❶ S-tree	❸ D-tree
DAG	❷ S-DAG	❹ D-DAG

BU: bottom-up on the AT structure. **APH**: acyclic phase-type (time distribution). **BDD**: binary decision diagram. **MTBDD**: multi-terminal BDD. **$\mathcal{C}$-BU**: repeated BU, identifying clones. **DPLL**: DPPL SAT-solving in the AT formula. **PTA**: priced time automata (semantics). **I/O-IMC**: input/output interactive Markov chains (semantics).

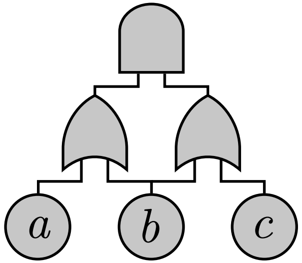
AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \big\{(v, u) \in N^2 \mid u \in ch(v)\big\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.



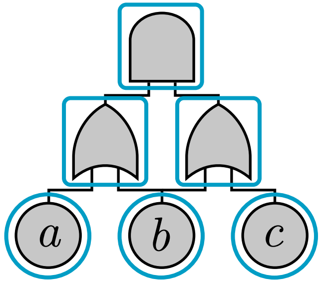
AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \big\{(v, u) \in N^2 \mid u \in ch(v)\big\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.



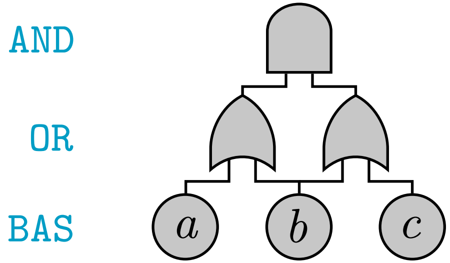
AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $\boxed{t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\} \text{ gives the } \textit{type} \text{ of each node;}}$
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \big\{(v,u) \in N^2 \mid u \in ch(v)\big\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.



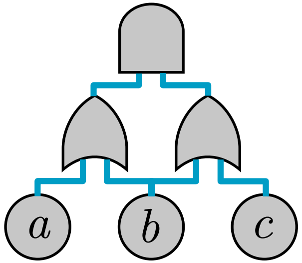
AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

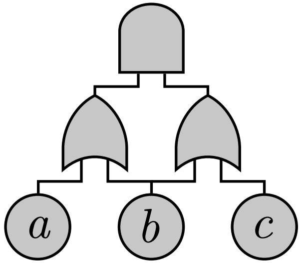
- (N, E) is a connected DAG, where $E = \{(v,u) \in N^2 \mid u \in ch(v)\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.



AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.



Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \big\{(v, u) \in N^2 \mid u \in ch(v)\big\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.

Definition (Metric)**.** Given an AT and a set V of values:

1. an *attribution* $\alpha\colon \mathrm{BAS} \to V$ assigns an *attribute value* $\alpha(a)$ to each basic attack step a;
2. an *attack metric* $\widehat{\alpha}\colon \mathscr{A}_T \to V$ assigns a value $\widehat{\alpha}(A)$ to an attack A; a *security metric* $\check{\alpha}\colon \mathcal{S}_T \to V$ assigns a value $\check{\alpha}(\mathcal{S})$ to a suite $\mathcal{S}$ of T.

We let $\check{\alpha}(T) = \check{\alpha}(\llbracket T \rrbracket)$: $\boxed{\text{the metric of an AT is given by its semantics.}}$

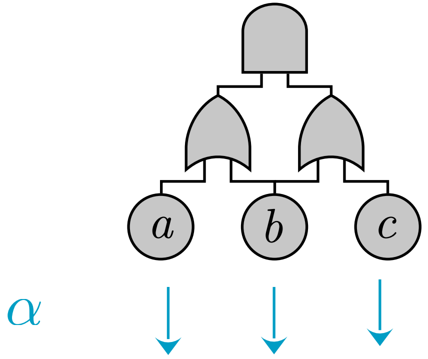
AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \{(v, u) \in N^2 \mid u \in ch(v)\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.



$V = \{\, 3\ ,\ 1\ ,\ 4 \,\}$

Definition (Metric)**.** Given an AT and a set V of values:

1. an $\boxed{\textit{attribution}\ \alpha\colon \mathrm{BAS} \to V}$ assigns an *attribute value* $\alpha(a)$ to each basic attack step a;
2. an *attack metric* $\widehat{\alpha}\colon \mathscr{A}_T \to V$ assigns a value $\widehat{\alpha}(A)$ to an attack A;
   a *security metric* $\breve{\alpha}\colon \mathcal{S}_T \to V$ assigns a value $\breve{\alpha}(\mathcal{S})$ to a suite $\mathcal{S}$ of T.

We let $\breve{\alpha}(T) = \breve{\alpha}([\![T]\!])$: the metric of an AT is given by its semantics.

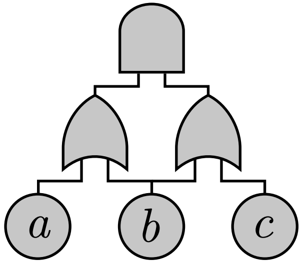
AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \{(v, u) \in N^2 \mid u \in ch(v)\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.



α

$V = \{\, 3 \ , \ 1 \ , \ 4 \,\}$

$\widehat{\alpha}(\text{“do } a \text{ and } c\text{”}) = 7$

Definition (Metric)**.** Given an AT and a set V of values:

1. an *attribution* $\alpha\colon \text{BAS} \to V$ assigns an *attribute value* $\alpha(a)$ to each basic attack step a;
2. an $\boxed{\textit{attack metric } \widehat{\alpha}\colon \mathscr{A}_T \to V}$ assigns a value $\widehat{\alpha}(A)$ to an attack A;
   a *security metric* $\breve{\alpha}\colon \mathcal{S}_T \to V$ assigns a value $\breve{\alpha}(\mathcal{S})$ to a suite $\mathcal{S}$ of T.

We let $\breve{\alpha}(T) = \breve{\alpha}([\![T]\!])$: the metric of an AT is given by its semantics.

AT syntax & metric on semantics

Definition (AT)**.** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

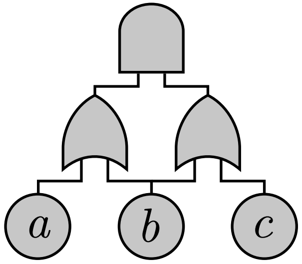
Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \{(v, u) \in N^2 \mid u \in ch(v)\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.

Definition (Metric)**.** Given an AT and a set V of values:

1. an *attribution* $\alpha\colon \mathrm{BAS} \to V$ assigns an *attribute value* $\alpha(a)$ to each basic attack step a;
2. an *attack metric* $\widehat{\alpha}\colon \mathscr{A}_T \to V$ assigns a value $\widehat{\alpha}(A)$ to an attack A;
   a $\boxed{\textit{security metric } \breve{\alpha}\colon \mathcal{S}_T \to V}$ assigns a value $\breve{\alpha}(\mathcal{S})$ to a suite $\mathcal{S}$ of T.

We let $\breve{\alpha}(T) = \breve{\alpha}(\llbracket T \rrbracket)$: the metric of an AT is given by its semantics.



α $\downarrow\ \downarrow\ \downarrow$

$V = \{\, 3 \;,\; 1 \;,\; 4 \,\}$

$\widehat{\alpha}(\text{“do } a \text{ and } c\text{”}) = 7$

$\breve{\alpha}(\text{“any } T \text{ attack”}) = 1$

AT syntax & metric on semantics

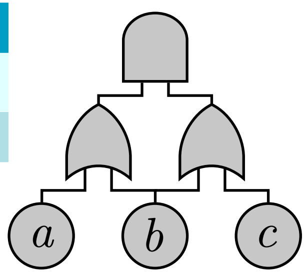
Definition (AT). An *attack tree* is a tuple $T = (N, t, ch)$ where:

- N is a finite set of *nodes*;
- $t\colon N \to \{\texttt{BAS}, \texttt{OR}, \texttt{AND}, \texttt{SAND}\}$ gives the *type* of each node;
- $ch\colon N \to N^*$ gives the sequence of *children* of a node.

Moreover, T satisfies the following constraints:

- (N, E) is a connected DAG, where $E = \{(v, u) \in N^2 \mid u \in ch(v)\}$;
- T has a unique root, denoted R_T: $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$;
- BAS_T nodes are the leaves of : $\forall v \in N.\ t(v) = \texttt{BAS} \Leftrightarrow ch(v) = \varepsilon$.

	Static	Dynamic
Tree	❶ S-tree	❸ D-tree
DAG	❷ S-DAG	❹ D-DAG



$$\alpha \quad \downarrow \quad \downarrow \quad \downarrow$$

$$V = \{\, 3 \;,\; 1 \;,\; 4 \,\}$$

$\widehat{\alpha}(\text{“do } a \text{ and } c\text{”}) = 7$

$\check{\alpha}(\text{“any } T \text{ attack”}) = 1$

Definition (Metric). Given an AT and a set V of values:

1. an *attribution* $\alpha\colon \text{BAS} \to V$ assigns an *attribute value* $\alpha(a)$ to each basic attack step a;
2. an *attack metric* $\widehat{\alpha}\colon \mathscr{A}_T \to V$ assigns a value $\widehat{\alpha}(A)$ to an attack A;
   a *security metric* $\check{\alpha}\colon \mathcal{S}_T \to V$ assigns a value $\check{\alpha}(\mathcal{S})$ to a suite $\mathcal{S}$ of T.

We let $\check{\alpha}(T) = \check{\alpha}([\![T]\!])$: the metric of an AT is given by its semantics.

Static AT semantics (no order in attacks)

- An **attack** is a set of BAS of the AT: $A \subseteq \mathrm{BAS}$
- An **attack suite** is a set of attacks: $\mathcal{S} \subseteq 2^{\mathrm{BAS}}$
- A **structure function** tells if an attack succeeds:

$$f_T(v, A) = \begin{cases} \top & \text{if } t(v) = \texttt{OR} \text{ and } \exists u \in ch(v).f_T(u, A) = \top, \\ \top & \text{if } t(v) = \texttt{AND} \text{ and } \forall u \in ch(v).f_T(u, A) = \top, \\ \top & \text{if } t(v) = \texttt{BAS} \text{ and } v \in A, \\ \bot & \text{otherwise.} \end{cases}$$

Static AT semantics (no order in attacks)

- An **attack** is a set of BAS of the AT: $A \subseteq \mathrm{BAS}$
- An **attack suite** is a set of attacks: $\mathcal{S} \subseteq 2^{\mathrm{BAS}}$
- A **structure function** tells if an attack succeeds:

$$f_T(v, A) = \begin{cases} \top & \text{if } t(v) = \texttt{OR} \ \text{ and } \exists u \in ch(v).f_T(u, A) = \top, \\ \top & \text{if } t(v) = \texttt{AND} \text{ and } \forall u \in ch(v).f_T(u, A) = \top, \\ \top & \text{if } t(v) = \texttt{BAS} \text{ and } v \in A, \\ \bot & \text{otherwise.} \end{cases}$$

- The **semantics of** T is the suite of all minimal successful attacks:

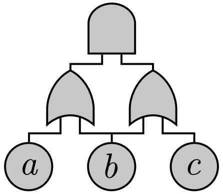
$$[\![T]\!] = \{A \subseteq \mathrm{BAS} \mid f_T(A) \wedge A \text{ is minimal}\}$$

Theorem: computing $[\![T]\!]$ is an **NP-complete** problem

Static AT metrics

- An **attribute domain** is a tuple $D = (V, \triangledown, \vartriangle)$ where:
  - $\triangledown : V^2 \to V$ is a **disjunctive** operator
  - $\vartriangle : V^2 \to V$ is a **conjunctive** operator

associative
&
commutative



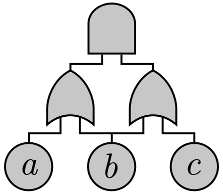
$$[\![T]\!] = \{\{b\}, \{a, c\}\}$$

Static AT metrics

- An **attribute domain** is a tuple $D = (V, \triangledown, \vartriangle)$ where:
  - $\triangledown : V^2 \to V$ is a **disjunctive** operator
  - $\vartriangle : V^2 \to V$ is a **conjunctive** operator

associative & commutative

Metric	V	$\triangledown$	$\vartriangle$
min cost	$\mathbb{N}_\infty$	min	$+$
min time	$\mathbb{N}_\infty$	min	$+$
min skill	$\mathbb{N}_\infty$	min	max
max challenge	$\mathbb{N}_\infty$	max	max
max damage	$\mathbb{N}_\infty$	max	$+$
discrete prob.	$[0,1]_\mathbb{Q}$	max	$*$
continu. prob.	$\mathbb{R} \to [0,1]_\mathbb{Q}$	max	$*$



$$[\![T]\!] = \{\{b\}, \{a, c\}\}$$

Static AT metrics

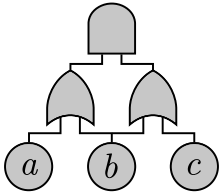
- An **attribute domain** is a tuple $D = (V, \triangledown, \vartriangle)$ where:
  - $\triangledown : V^2 \to V$ is a **disjunctive** operator
  - $\vartriangle : V^2 \to V$ is a **conjunctive** operator

  $\left.\begin{array}{l} \\ \\ \end{array}\right\}$ associative & commutative

Metric	V	$\triangledown$	$\vartriangle$
min cost	$\mathbb{N}_\infty$	min	$+$
min time	$\mathbb{N}_\infty$	min	$+$
min skill	$\mathbb{N}_\infty$	min	max
max challenge	$\mathbb{N}_\infty$	max	max
max damage	$\mathbb{N}_\infty$	max	$+$
discrete prob.	$[0,1]_\mathbb{Q}$	max	$*$
continu. prob.	$\mathbb{R} \to [0,1]_\mathbb{Q}$	max	$*$

- The **metric for a static AT** T, attribution α, and domain D is:

$$\breve{\alpha}(T) = \underbrace{\bigtriangledown_{A \in [\![T]\!]}}_{\breve{\alpha}} \underbrace{\bigtriangleup_{a \in A}}_{\widehat{\alpha}} \alpha(a)$$



$$[\![T]\!] = \{\{b\}, \{a, c\}\}$$

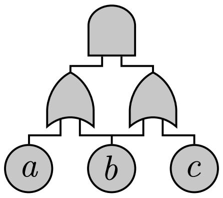
Static AT metrics

- An **attribute domain** is a tuple $D = (V, \triangledown, \vartriangle)$ where:
  - $\triangledown : V^2 \to V$ is a **disjunctive** operator
  - $\vartriangle : V^2 \to V$ is a **conjunctive** operator

  } associative & commutative

- The **metric for a static AT** T, attribution α, and domain D is:

$$\check{\alpha}(T) = \underbrace{\bigtriangledown_{A \in [\![T]\!]}}_{\check{\alpha}} \underbrace{\bigtriangleup_{a \in A}}_{\widehat{\alpha}} \alpha(a)$$



$$\downarrow \quad \downarrow \quad \downarrow \quad = \alpha$$

$$\{\, 3\ ,\ \ 1\ \ ,\ \ 4\ \} = V$$

$$[\![T]\!] = \{\{b\}, \{a, c\}\}$$

$$D = (V, \triangledown, \vartriangle) = (\mathbb{N}, \min, +)$$

min cost

METRIC	V	$\triangledown$	$\vartriangle$
min cost	$\mathbb{N}_\infty$	min	$+$
min time	$\mathbb{N}_\infty$	min	$+$
min skill	$\mathbb{N}_\infty$	min	max
max challenge	$\mathbb{N}_\infty$	max	max
max damage	$\mathbb{N}_\infty$	max	$+$
discrete prob.	$[0,1]_\mathbb{Q}$	max	$*$
continu. prob.	$\mathbb{R} \to [0,1]_\mathbb{Q}$	max	$*$

Static AT metrics

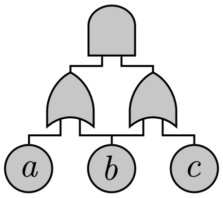
- An **attribute domain** is a tuple $D = (V, \triangledown, \vartriangle)$ where:
  - $\triangledown : V^2 \to V$ is a **disjunctive** operator
  - $\vartriangle : V^2 \to V$ is a **conjunctive** operator

  associative & commutative

- The **metric for a static AT** T, attribution α, and domain D is:

$$\check{\alpha}(T) = \underbrace{\mathop{\nabla}_{A \in [\![T]\!]}}_{\check{\alpha}} \underbrace{\mathop{\Delta}_{a \in A}}_{\widehat{\alpha}} \alpha(a)$$

Metric	V	$\triangledown$	$\vartriangle$
$\boxed{\text{min cost}}$	$\mathbb{N}_\infty$	min	$+$
min time	$\mathbb{N}_\infty$	min	$+$
min skill	$\mathbb{N}_\infty$	min	max
max challenge	$\mathbb{N}_\infty$	max	max
max damage	$\mathbb{N}_\infty$	max	$+$
discrete prob.	$[0,1]_\mathbb{Q}$	max	$*$
continu. prob.	$\mathbb{R} \to [0,1]_\mathbb{Q}$	max	$*$



$\downarrow \quad \downarrow \quad \downarrow \; = \alpha$

$\{\, 3 \,,\; 1 \,,\; 4 \,\} = V$

$$[\![T]\!] = \big\{\{b\}, \{a, c\}\big\}$$

$$D = (V, \triangledown, \vartriangle) = (\mathbb{N}, \min, +)$$

min cost

$$\begin{aligned}\check{\alpha}(T) &= \mathop{\nabla}_{A \in [\![T]\!]} \mathop{\Delta}_{a \in A} \alpha(a) \\ &= \big(\alpha(b)\big) \triangledown \big(\alpha(a) \vartriangle \alpha(c)\big) \\ &= (1) \min (3 + 4) \\ &= 1\end{aligned}$$

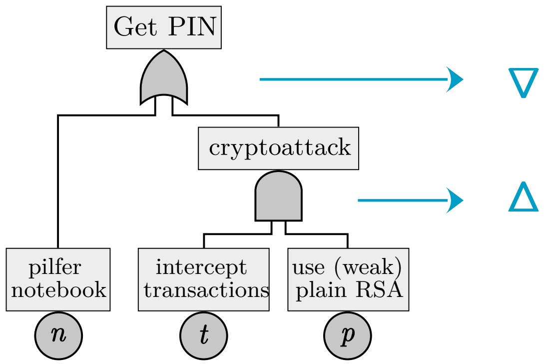
① T is a static tree

- Domain $D = (V, \triangledown, \vartriangle)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$

S. Mauw & M. Oostdijk: "*Foundations of Attack Trees*." ICISC 2006. DOI: 10.1007/11734727_17

① T is a static tree

- Domain $D = (V, \triangledown, \vartriangle)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$



S. Mauw & M. Oostdijk: "*Foundations of Attack Trees*." ICISC 2006. DOI: 10.1007/11734727_17

① T is a static tree

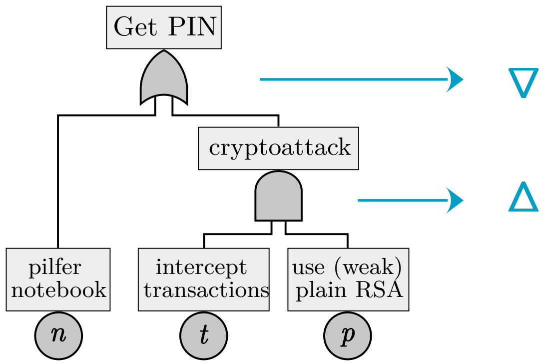
- Domain $D = (V, \triangledown, \vartriangle)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$

$\mathrm{BU}_{\mathtt{SAT}}$ algorithm, **linear in T**

Input: S-tree $T = (N, t, ch)$,
node $v \in N$,
attribution α,
semiring attribute domain
$D = (V, \triangledown, \vartriangle)$.

Output: Metric value $\breve{\alpha}(T) \in V$.

if $t(v) = \mathtt{OR}$ **then**
  return $\bigtriangledown_{u \in ch(v)} \mathrm{BU}_{\mathtt{SAT}}(T, u, \alpha, D)$
else if $t(v) = \mathtt{AND}$ **then**
  return $\bigtriangleup_{u \in ch(v)} \mathrm{BU}_{\mathtt{SAT}}(T, u, \alpha, D)$
else // $t(v) = \mathtt{BAS}$
  return $\alpha(v)$



S. Mauw & M. Oostdijk: "*Foundations of Attack Trees*." ICISC 2006. DOI: 10.1007/11734727_17

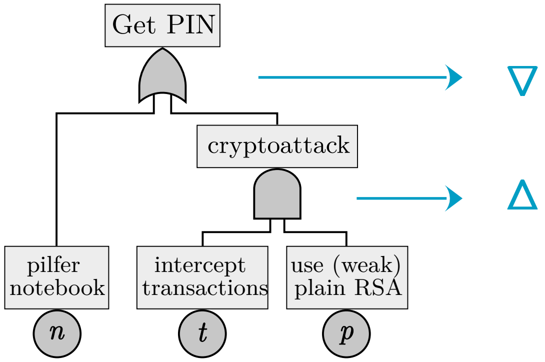
① T is a static tree

- Domain $D = (V, \triangledown, \vartriangle)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$

$\texttt{BU}_{\texttt{SAT}}$ algorithm, **linear in T**

Input: S-tree $T = (N, t, ch)$,
node $v \in N$,
attribution α,
semiring attribute domain
$D = (V, \triangledown, \vartriangle)$.
Output: Metric value $\check{\alpha}(T) \in V$.

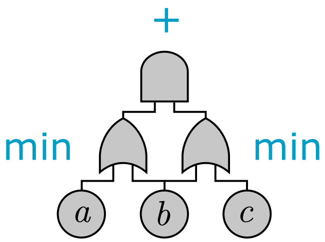
if $t(v) = \texttt{OR}$ **then**
  return $\bigtriangledown_{u \in ch(v)} \texttt{BU}_{\texttt{SAT}}(T, u, \alpha, D)$
else if $t(v) = \texttt{AND}$ **then**
  return $\bigtriangleup_{u \in ch(v)} \texttt{BU}_{\texttt{SAT}}(T, u, \alpha, D)$
else // $t(v) = \texttt{BAS}$
  return $\alpha(v)$



Theorem. *Let T be a static AT with tree structure, α an attribution on V, and $D = (V, \triangledown, \vartriangle)$ a semiring attribute domain. Then $\check{\alpha}(T) = \texttt{BU}_{\texttt{SAT}}(T, R_T, \alpha, D)$.*

S. Mauw & M. Oostdijk: "*Foundations of Attack Trees*." ICISC 2006. DOI: 10.1007/11734727_17

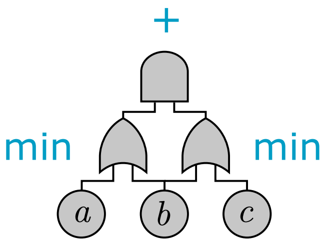
② T is a static DAG



$$\breve{\alpha}(T) = 1 \qquad \{\,3\ ,\ 1\ ,\ 4\,\}$$

$$\mathtt{BU}_{\mathtt{SAT}}(T) = (3 \min 1) + (1 \min 4) = 2$$

② T is a static DAG



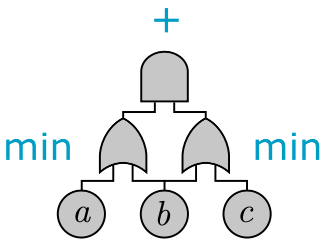
$$\{\,3\ ,\ 1\ ,\ 4\,\}$$

$$\breve{\alpha}(T) = 1$$

$$\mathrm{BU}_{\cancel{\mathtt{SAT}}}(T) = (3 \min 1) + (1 \min 4) = 2$$

② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (\underbrace{W, Lab}_{\mathrm{BAS}_T \cup \{1,0\}}, \underbrace{Low, High}_{f_T})$



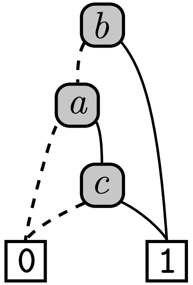
$\{\,3\ ,\ 1\ ,\ 4\,\}$

$\breve{\alpha}(T) = 1$

$\mathrm{BU}_{\cancel{\mathtt{SAT}}}(T) = (3 \min 1) + (1 \min 4) = 2$

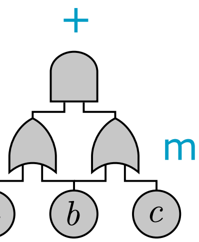
② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (\underbrace{W, Lab}_{\mathrm{BAS}_T \cup \{1,0\}}, \underbrace{Low, High}_{f_T})$



failure success

$b < a < c$



$\{3, 1, 4\}$

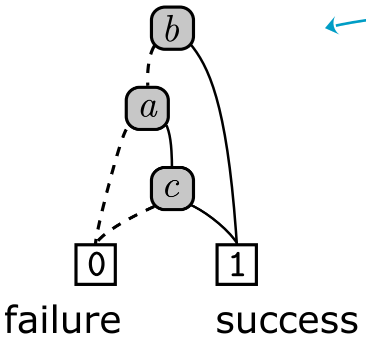
② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (\underbrace{W, Lab,}_{\mathrm{BAS}_T \cup \{1,0\}} \underbrace{Low, High}_{f_T})$

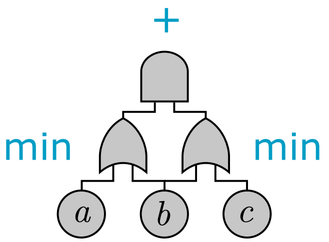
$\mathtt{BDD_{SAT}}$ algorithm, **linear in** B_T

Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.
Output: Metric value $\check{\alpha}(T) \in V$.

if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $Lab(w) = v \in \mathrm{BAS}$, or not
  return $\big(\alpha(Lab(w)) \vartriangle \cdots$
  $\cdots \mathtt{BDD_{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\triangledown$ $\mathtt{BDD_{SAT}}(B_T, Low(w), \alpha, D_*)$



$b < a < c$



$\{\,3\ ,\ 1\ ,\ 4\,\}$

② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (\underbrace{W, Lab}_{\mathrm{BAS}_T \cup \{1,0\}}, \underbrace{Low, High}_{f_T})$

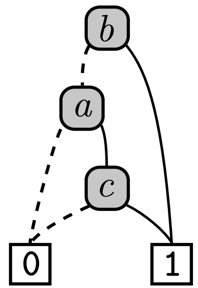
$\mathtt{BDD_{SAT}}$ algorithm, **linear in B_T**

Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.

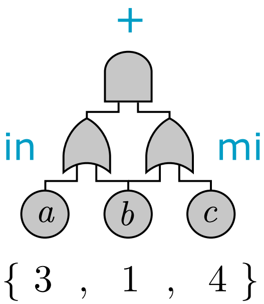
Output: Metric value $\check{\alpha}(T) \in V$.

if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $Lab(w) = v \in$ BAS, or not
  return $\boxed{\big(\alpha(Lab(w)) \vartriangle \cdots \; \cdots \mathtt{BDD_{SAT}}(B_T, High(w), \alpha, D_*)\big)}$
  $\triangledown$ $\mathtt{BDD_{SAT}}(B_T, Low(w), \alpha, D_*)$

do attack $Lab(w) = v \in \mathrm{BAS}$

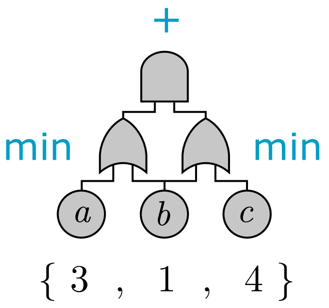


$b < a < c$



② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (\underbrace{W, Lab}_{\mathrm{BAS}_T \cup \{1,0\}}, \underbrace{Low, High}_{f_T})$

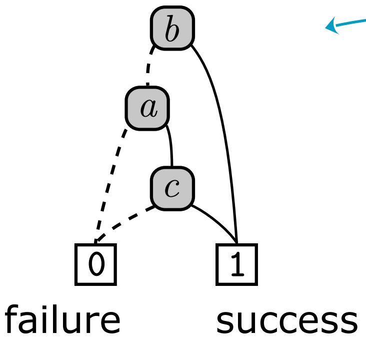


$\mathtt{BDD}_{\mathtt{SAT}}$ algorithm, **linear in** B_T

Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.

Output: Metric value $\check{\alpha}(T) \in V$.

if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $Lab(w) = v \in$ BAS, or not
  return $\big(\alpha(Lab(w)) \vartriangle \cdots$
  $\cdots \mathtt{BDD}_{\mathtt{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\boxed{\triangledown\ \mathtt{BDD}_{\mathtt{SAT}}(B_T, Low(w), \alpha, D_*)}$ do not attack v



② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (\underbrace{W, Lab}_{\mathrm{BAS}_T \cup \{1,0\}}, \underbrace{Low, High}_{f_T})$

$\mathtt{BDD_{SAT}}$ algorithm, **linear in** B_T

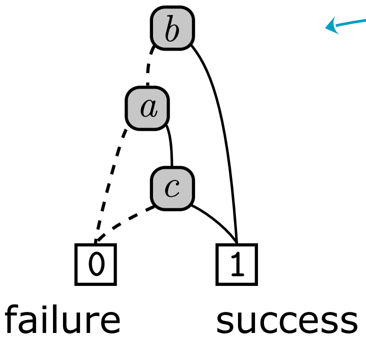
Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.

Output: Metric value $\breve{\alpha}(T) \in V$.

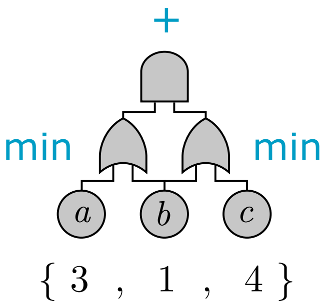
if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$

failure/success

else // either do $Lab(w) = v \in$ BAS, or not
  return $\big(\alpha(Lab(w)) \vartriangle \cdots$
  $\cdots \mathtt{BDD_{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\triangledown$ $\mathtt{BDD_{SAT}}(B_T, Low(w), \alpha, D_*)$

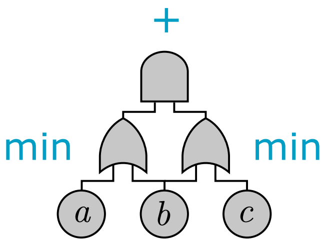


$b < a < c$



② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (W, Lab, Low, High)$



$b < a < c$

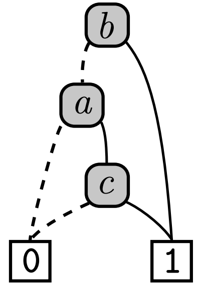
$\{\ 3\ ,\ 1\ ,\ 4\ \}$

$D = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle) = (\text{cost}, \min, +, \infty, 0)$

$\texttt{BDD}_{\texttt{SAT}}$ algorithm, **linear in** B_T

Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.
Output: Metric value $\check{\alpha}(T) \in V$.

if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $Lab(w) = v \in$ BAS, or not
  return $\big(\alpha(Lab(w)) \vartriangle \cdots$
  $\cdots \texttt{BDD}_{\texttt{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\triangledown\ \texttt{BDD}_{\texttt{SAT}}(B_T, Low(w), \alpha, D_*)$



$1_\triangledown = \infty$ $0 = 1_\vartriangle$

$\texttt{BDD}_{\texttt{SAT}}(w_b) =$

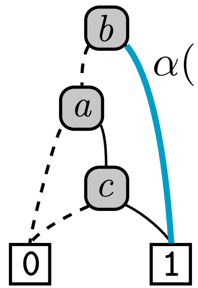
② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (W, \mathit{Lab}, \mathit{Low}, \mathit{High})$

BDD$_{\texttt{SAT}}$ algorithm, **linear in** B_T

Input: BDD $B_T = (W, \mathit{Low}, \mathit{High}, \mathit{Lab})$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.
Output: Metric value $\check{\alpha}(T) \in V$.

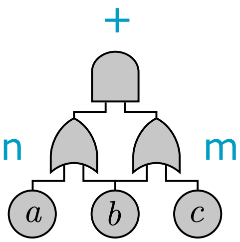
if $\mathit{Lab}(w) = 0$ **then**
  return $1_\triangledown$
else if $\mathit{Lab}(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $\mathit{Lab}(w) = v \in$ BAS, or not
  return $\big(\alpha(\mathit{Lab}(w)) \vartriangle \cdots$
  $\cdots \texttt{BDD}_{\texttt{SAT}}(B_T, \mathit{High}(w), \alpha, D_*)\big)$
  $\triangledown \ \texttt{BDD}_{\texttt{SAT}}(B_T, \mathit{Low}(w), \alpha, D_*)$



$\alpha(b) = 1$

$1_\triangledown = \infty \qquad 0 = 1_\vartriangle$

$b < a < c$



$\{\,3 \ , \ 1 \ , \ 4\,\}$

$$D = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle) = (\text{cost}, \min, +, \infty, 0)$$

$$\texttt{BDD}_{\texttt{SAT}}(w_b) = (1 + 0) \min \texttt{BDD}_{\texttt{SAT}}(w_a)$$

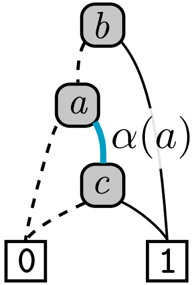
② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (W, Lab, Low, High)$

$\mathtt{BDD_{SAT}}$ algorithm, **linear in** B_T

Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.
Output: Metric value $\breve{\alpha}(T) \in V$.

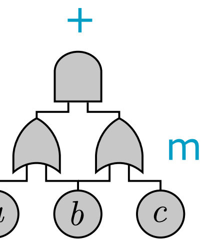
if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $Lab(w) = v \in$ BAS, or not
  return $\big(\alpha(Lab(w)) \vartriangle \cdots$
  $\cdots \mathtt{BDD_{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\triangledown\ \mathtt{BDD_{SAT}}(B_T, Low(w), \alpha, D_*)$



$\alpha(a) = 3$

$1_\triangledown = \infty$ $\quad 0 = 1_\vartriangle$

$b < a < c$



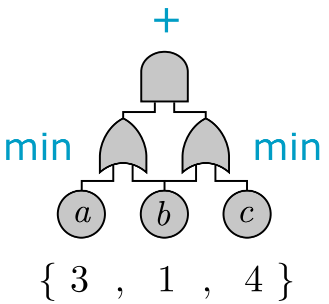
$\{\,3\ ,\ 1\ ,\ 4\,\}$

$$D = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle) = (\text{cost}, \min, +, \infty, 0)$$

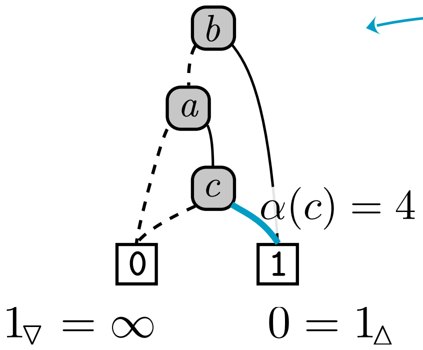
$$\begin{aligned} \mathtt{BDD_{SAT}}(w_b) &= (1+0) \min \mathtt{BDD_{SAT}}(w_a) \\ &= (1) \min \big((3 + \mathtt{BDD_{SAT}}(w_c)) \min \infty\big) \end{aligned}$$

② T is a static DAG

- Binary Decision Diagram (**BDD**) $B_T = (W, Lab, Low, High)$



$b < a < c$



$$D = (V, \nabla, \Delta, 1_\nabla, 1_\Delta) = (\text{cost}, \min, +, \infty, 0)$$

$$\begin{aligned}\texttt{BDD}_{\texttt{SAT}}(w_b) &= (1+0) \min \texttt{BDD}_{\texttt{SAT}}(w_a) \\ &= (1) \min \big((3 + \texttt{BDD}_{\texttt{SAT}}(w_c)) \min \infty\big) \\ &= (1) \min \big(3 + ((4+0) \min \infty)\big) \;=\; 1\end{aligned}$$

BDD$_{\texttt{SAT}}$ algorithm, **linear in B_T**

Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \nabla, \Delta, 1_\nabla, 1_\Delta)$.
Output: Metric value $\breve{\alpha}(T) \in V$.

if $Lab(w) = 0$ **then**
  return 1_∇
else if $Lab(w) = 1$ **then**
  return 1_Δ
else // either do $Lab(w) = v \in$ BAS, or not
  return $\big(\alpha(Lab(w)) \,\Delta \cdots$
  $\cdots \texttt{BDD}_{\texttt{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\nabla\; \texttt{BDD}_{\texttt{SAT}}(B_T, Low(w), \alpha, D_*)\big)$

② T is a static DAG

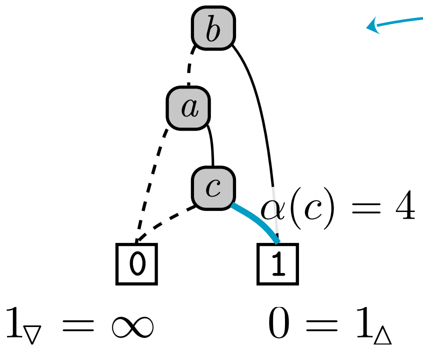
- Binary Decision Diagram (**BDD**) $B_T = (W, Lab, Low, High)$

$\mathtt{BDD_{SAT}}$ algorithm, linear in B_T

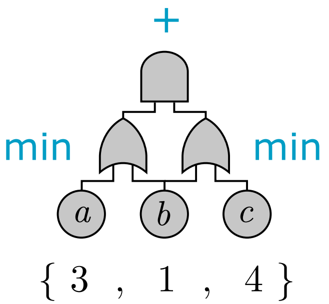
Input: BDD $B_T = (W, Low, High, Lab)$,
node $w \in W$,
attribution α,
semiring attribute domain
$D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle)$.

Output: Metric value $\breve{\alpha}(T) \in V$.

if $Lab(w) = 0$ **then**
  return $1_\triangledown$
else if $Lab(w) = 1$ **then**
  return $1_\vartriangle$
else // either do $Lab(w) = v \in$ BAS, or not
  return $\big(\alpha(Lab(w)) \vartriangle \cdots$
  $\cdots \mathtt{BDD_{SAT}}(B_T, High(w), \alpha, D_*)\big)$
  $\triangledown\ \mathtt{BDD_{SAT}}(B_T, Low(w), \alpha, D_*)$



$b < a < c$

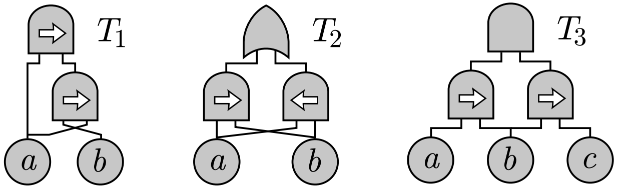


$$D = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle) = (\text{cost}, \min, +, \infty, 0)$$

$$\begin{aligned}\mathtt{BDD_{SAT}}(w_b) &= (1+0) \min \mathtt{BDD_{SAT}}(w_a) \\ &= (1) \min \big((3 + \mathtt{BDD_{SAT}}(w_c)) \min \infty\big) \\ &= (1) \min \big(3 + ((4+0) \min \infty)\big) \ = \ 1\end{aligned}$$

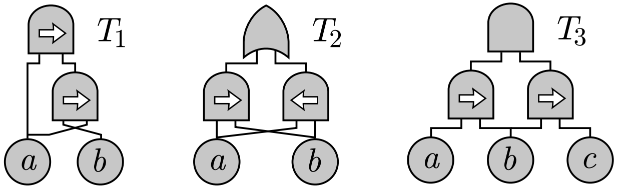
Theorem. *Let T be a static AT, B_T its BDD encoding, α an attribution on V, and $D_* = (V, \triangledown, \vartriangle, 1_\triangledown, 1_\vartriangle))$ a semiring attr. dom. with neutral elements resp. for $\triangledown$ and $\vartriangle$.*
Then $\breve{\alpha}(T) = \mathtt{BDD_{SAT}}(B_T, R_B, \alpha, D_*)$.

Dynamic AT semantics (order in attacks)



Dynamic AT semantics (order in attacks)

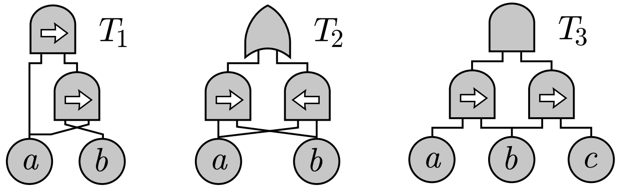
- An **attack** is a partially ordered set: $\langle \boldsymbol{A}, \prec \rangle$
  - $a \prec b$ iff $a \in \mathrm{BAS}$ must finish before b begins



$a \prec b \ \wedge \ b \prec a$ $\qquad$ $a \prec b \prec c$

Dynamic AT semantics (order in attacks)

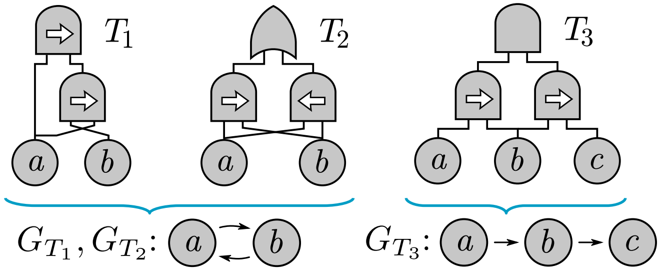
- An **attack** is a partially ordered set: $\langle A, \prec \rangle$
  - $a \prec b$ iff $a \in \mathrm{BAS}$ must finish before b begins



$a \prec b \ \wedge \ b \prec a$ $\qquad a \prec b \prec c$

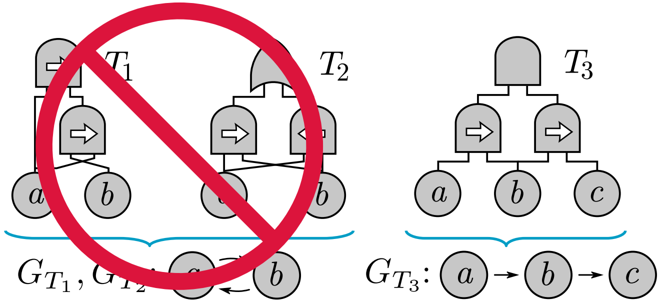
- The **ordering graph** $G_T = (\mathrm{BAS}_T, \rightarrow)$ of a dynamic AT has the edge $a \rightarrow b$ iff $\exists\ \mathrm{SAND}(v_1, \ldots, v_n)$ s.t. $a \in \mathrm{BAS}(v_i) \wedge b \in \mathrm{BAS}(v_{i+1})$
- A dynamic AT is **well-formed** if its ordering graph is acyclic

Dynamic AT semantics (order in attacks)



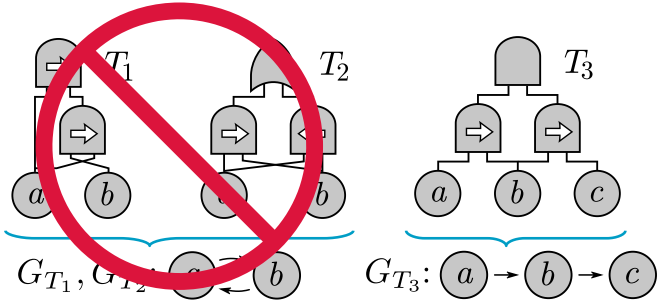
- An **attack** is a partially ordered set: $\langle A, \prec \rangle$
  - $a \prec b$ iff $a \in \mathrm{BAS}$ must finish before b begins
- The **ordering graph** $G_T = (\mathrm{BAS}_T, \to)$ of a dynamic AT has the edge $a \to b$ iff $\exists\ \mathrm{SAND}(v_1, \ldots, v_n)$ s.t. $a \in \mathrm{BAS}(v_i) \wedge b \in \mathrm{BAS}(v_{i+1})$
- A dynamic AT is **well-formed** if its ordering graph is acyclic

Dynamic AT semantics (order in attacks)



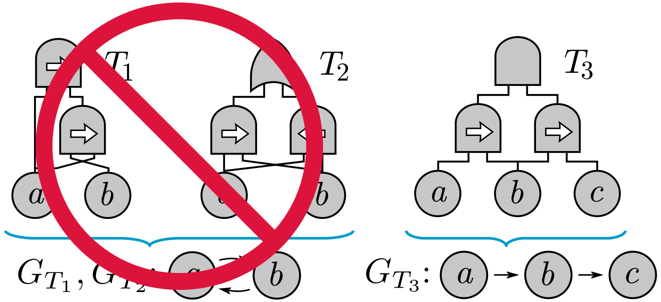
- An **attack** is a partially ordered set: $\langle A, \prec \rangle$
  - $a \prec b$ iff $a \in \mathrm{BAS}$ must finish before b begins
- The **ordering graph** $G_T = (\mathrm{BAS}_T, \rightarrow)$ of a dynamic AT has the edge $a \rightarrow b$ iff $\exists\ \mathrm{SAND}(v_1, \ldots, v_n)$ s.t. $a \in \mathrm{BAS}(v_i) \wedge b \in \mathrm{BAS}(v_{i+1})$
- A dynamic AT is **well-formed** if its ordering graph is acyclic

Dynamic AT semantics (order in attacks)



- An **attack** is a partially ordered set: $\langle \boldsymbol{A}, \prec \rangle$
  - $a \prec b$ iff $a \in \mathrm{BAS}$ must finish before b begins
- The **ordering graph** $G_T = (\mathrm{BAS}_T, \rightarrow)$ of a dynamic AT has the edge $a \rightarrow b$ iff $\exists\ \mathrm{SAND}(v_1, \ldots, v_n)$ s.t. $a \in \mathrm{BAS}(v_i) \wedge b \in \mathrm{BAS}(v_{i+1})$
- A dynamic AT is **well-formed** if its ordering graph is acyclic
- Attacks $\langle A, \prec \rangle$ defined for well-formed ATs only
  - $\prec$ is a restriction (to $A \subseteq \mathrm{BAS}$) of the edges of G_T

Dynamic AT semantics (order in attacks)



- An **attack** is a partially ordered set: $\langle A, \prec \rangle$
  - $a \prec b$ iff $a \in \mathrm{BAS}$ must finish before b begins
- The **ordering graph** $G_T = (\mathrm{BAS}_T, \to)$ of a dynamic AT has the edge $a \to b$ iff $\exists\, \mathrm{SAND}(v_1, \ldots, v_n)$ s.t. $a \in \mathrm{BAS}(v_i) \wedge b \in \mathrm{BAS}(v_{i+1})$
- A dynamic AT is **well-formed** if its ordering graph is acyclic

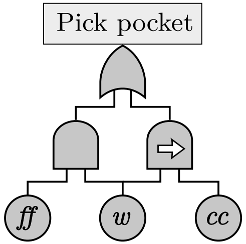
- Attacks $\langle A, \prec \rangle$ defined for well-formed ATs only
  - $\prec$ is a restriction (to $A \subseteq \mathrm{BAS}$) of the edges of G_T

- The **semantics of** T is the suite of all minimal successful attacks
  - $\langle A, \prec \rangle$ is minimal iff $A \subseteq \mathrm{BAS}$ and $\prec\, \subseteq \mathrm{BAS}^2$ are minimal

Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \triangledown, \vartriangle, \triangleright)$
  - $\triangleright : V^2 \to V$ is a **sequential** operator

    For the ordered steps. Also associative & commutative.



$$[\![T]\!] = \{ \langle \{ f\!f, w \}, \varnothing \rangle \\ , \langle \{ w, cc \}, \{ w \prec cc \} \rangle \}$$

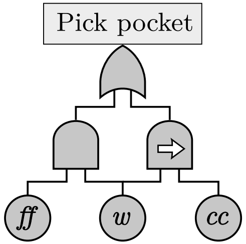
Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \triangledown, \vartriangle, \triangleright)$
  - $\triangleright : V^2 \to V$ is a **sequential** operator

    For the ordered steps. Also associative & commutative.

- The **metric for a dynamic AT** T, attribution α, and domain D is:

$$\check{\alpha}(T) = \underbrace{\bigtriangledown_{\langle A,\prec\rangle\in[\![T]\!]}}_{\check{\alpha}} \underbrace{\bigtriangleup_{C\in H_A^{\prec}}}_{\widehat{\alpha}} \underbrace{\rhd_{a\in C}}_{\vec{\alpha}} \alpha(a)$$



$$[\![T]\!] = \{\langle\{ff, w\}, \varnothing\rangle\\ ,\langle\{w, cc\}, \{w \prec cc\}\rangle\}$$

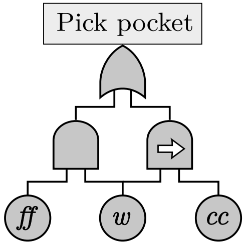
Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \triangledown, \vartriangle, \triangleright)$
  - $\triangleright : V^2 \to V$ is a **sequential** operator

    For the ordered steps. Also associative & commutative.

- The **metric for a dynamic AT** T, attribution α, and domain D is:

$$\check{\alpha}(T) = \underbrace{\bigtriangledown_{\langle A,\prec\rangle\in[\![T]\!]}}_{\check{\alpha}} \underbrace{\bigtriangleup_{C\in H_A^{\prec}}}_{\widehat{\alpha}} \underbrace{\rhd_{a\in C}}_{\vec{\alpha}} \alpha(a)$$



$\{\,3\ ,\ 15\ ,\ 1\,\}$

$$[\![T]\!] = \{\langle\{ff, w\}, \varnothing\rangle, \langle\{w, cc\}, \{w \prec cc\}\rangle\}$$

$$D = (V, \triangledown, \vartriangle, \triangleright) = (\mathbb{N}, \min, \max, +)$$

min time

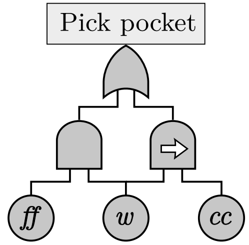
Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \triangledown, \vartriangle, \triangleright)$
  - $\triangleright : V^2 \to V$ is a **sequential** operator

    For the ordered steps. Also associative & commutative.

- The **metric for a dynamic AT** T, attribution α, and domain D is:

$$\check{\alpha}(T) = \underbrace{\mathop{\bigtriangledown}_{\langle A,\prec\rangle\in[\![T]\!]}}_{\check{\alpha}} \underbrace{\mathop{\bigtriangleup}_{C\in H_A^{\prec}}}_{\hat{\alpha}} \underbrace{\mathop{\rhd}_{a\in C}}_{\vec{\alpha}} \alpha(a)$$



$\{\,3\ ,\ 15\ ,\ 1\,\}$

$$[\![T]\!] = \{\langle\{ff, w\}, \varnothing\rangle\\ ,\langle\{w, cc\}, \{w \prec cc\}\rangle\}$$

$$D = (V, \triangledown, \vartriangle, \triangleright) = (\mathbb{N}, \min, \max, +)$$

min time

$$\begin{aligned}\check{\alpha}(T) &= \mathop{\bigtriangledown}_{\langle A,\prec\rangle\in[\![T]\!]} \mathop{\bigtriangleup}_{C\in H_A^{\prec}} \mathop{\rhd}_{a\in C} \alpha(a)\\ &= \big(\alpha(ff) \vartriangle \alpha(w)\big) \triangledown \big(\alpha(w) \triangleright \alpha(cc)\big)\\ &= (3 \max 15) \min (15 + 1)\\ &= 15\end{aligned}$$

③ T is a dynamic tree

- $D = (V, \triangledown, \vartriangle, \triangleright)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$, and $\triangleright$ over $\triangledown$ and $\vartriangle$

③ T is a dynamic tree

- $D = (V, \triangledown, \vartriangle, \vartriangleright)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$, and $\vartriangleright$ over $\triangledown$ and $\vartriangle$

$\mathtt{BU}_{\mathtt{DAT}}$ algorithm, **linear in T**

Input: S-tree $T = (N, t, ch)$,
node $v \in N$,
attribution α,
semiring dynamic attr. dom.
$D = (V, \triangledown, \vartriangle, \vartriangleright)$.

Output: Metric value $\breve{\alpha}(T) \in V$.

if $t(v) = \mathtt{OR}$ **then**
    return $\bigtriangledown_{u \in ch(v)} \mathtt{BU}_{\mathtt{DAT}}(T, u, \alpha, D)$
else if $t(v) = \mathtt{AND}$ **then**
    return $\bigtriangleup_{u \in ch(v)} \mathtt{BU}_{\mathtt{DAT}}(T, u, \alpha, D)$
else if $t(v) = \mathtt{SAND}$ **then**
    return $\rhd_{u \in ch(v)} \mathtt{BU}_{\mathtt{DAT}}(T, u, \alpha, D)$
else // $t(v) = \mathtt{BAS}$
    return $\alpha(v)$

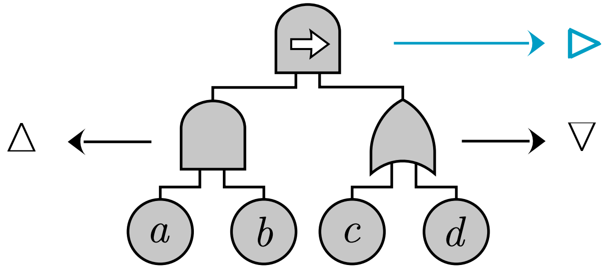
③ T is a dynamic tree

- $D = (V, \triangledown, \vartriangle, \triangleright)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$, and $\triangleright$ over $\triangledown$ and $\vartriangle$

$\texttt{BU}_{\texttt{DAT}}$ algorithm, **linear in T**

Input: S-tree $T = (N, t, ch)$,
node $v \in N$,
attribution α,
semiring dynamic attr. dom.
$D = (V, \triangledown, \vartriangle, \triangleright)$.
Output: Metric value $\breve{\alpha}(T) \in V$.

if $t(v) = \texttt{OR}$ **then**
  return $\bigtriangledown_{u \in ch(v)} \texttt{BU}_{\texttt{DAT}}(T, u, \alpha, D)$
else if $t(v) = \texttt{AND}$ **then**
  return $\bigtriangleup_{u \in ch(v)} \texttt{BU}_{\texttt{DAT}}(T, u, \alpha, D)$
else if $t(v) = \texttt{SAND}$ **then**
  return $\rhd_{u \in ch(v)} \texttt{BU}_{\texttt{DAT}}(T, u, \alpha, D)$
else // $t(v) = \texttt{BAS}$
  return $\alpha(v)$



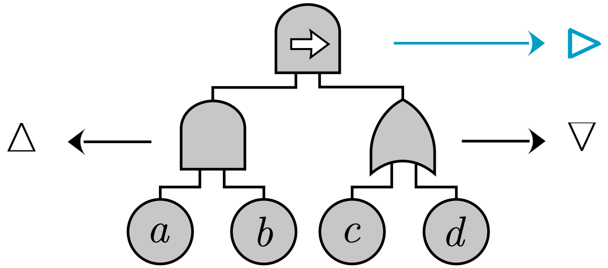
③ T is a dynamic tree

- $D = (V, \triangledown, \vartriangle, \triangleright)$ is a **semiring** if $\vartriangle$ distributes over $\triangledown$, and $\triangleright$ over $\triangledown$ and $\vartriangle$

$\mathtt{BU_{DAT}}$ algorithm, **linear in T**

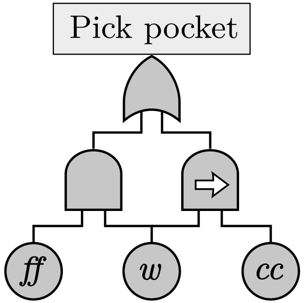
Input: S-tree $T = (N, t, ch)$,
node $v \in N$,
attribution α,
semiring dynamic attr. dom.
$D = (V, \triangledown, \vartriangle, \triangleright)$.
Output: Metric value $\breve{\alpha}(T) \in V$.

if $t(v) = \mathtt{OR}$ **then**
  return $\bigtriangledown_{u \in ch(v)} \mathtt{BU_{DAT}}(T, u, \alpha, D)$
else if $t(v) = \mathtt{AND}$ **then**
  return $\bigtriangleup_{u \in ch(v)} \mathtt{BU_{DAT}}(T, u, \alpha, D)$
else if $t(v) = \mathtt{SAND}$ **then**
  return $\rhd_{u \in ch(v)} \mathtt{BU_{DAT}}(T, u, \alpha, D)$
else // $t(v) = \mathtt{BAS}$
  return $\alpha(v)$

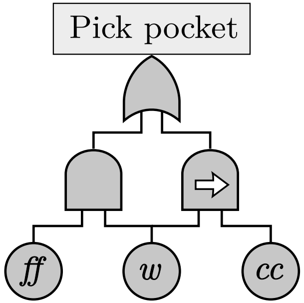


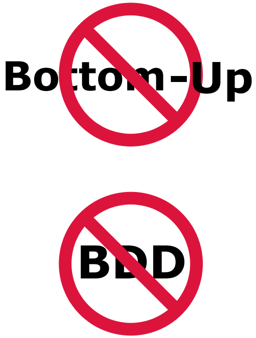
Theorem. *Let T be a dynamic AT with tree structure, α an attribution on V, and $D = (V, \triangledown, \vartriangle, \triangleright)$ a semiring dyn. attr. dom. Then $\breve{\alpha}(T) = \mathtt{BU_{DAT}}(T, R_T, \alpha, D)$.*

④ T is a dynamic DAG

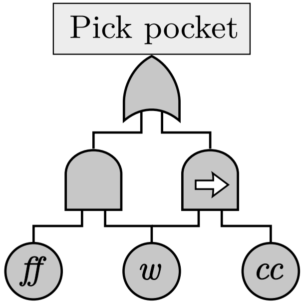


④ T is a dynamic DAG





④ T is a dynamic DAG



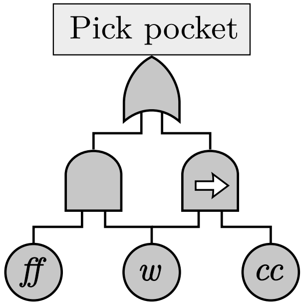
~~Bottom-Up~~

~~BDD~~

$$\breve{\alpha}(T) \;=\; \mathop{\bigtriangledown}_{\langle A,\prec\rangle\in[\![T]\!]} \; \mathop{\bigtriangleup}_{C\in H_A^{\prec}} \; \mathop{\rhd}_{a\in C} \; \alpha(a)$$

$[\![T]\!]$ can be computed from the ordering graph G_T
and the semantics of the static transform T'

④ T is a dynamic DAG



~~Bottom-Up~~

~~BDD~~

$$\breve{\alpha}(T) \;=\; \mathop{\bigtriangledown}_{\langle A,\prec\rangle\in[\![T]\!]} \; \mathop{\bigtriangleup}_{C\in H_A^{\prec}} \; \mathop{\triangleright}_{a\in C} \; \alpha(a)$$

$[\![T]\!]$ can be computed from the ordering graph G_T and the semantics of the static transform T'

Else: extend **sequential BDDs** for attack metrics?

- An S-BDD considers all combinations of descendants of SAND gates
- Combinatorial explosion on top of exponential explosion :’(

H. Yu & X. Wu: “*A method for transformation from dynamic fault tree to binary decision diagram*.” (2020) Part O: Journal of Risk and Reliability.
DOI: 10.1177/1748006X20974187

Summary of contributions

	Static	Dynamic
Tree	❶ S-tree	❸ D-tree
DAG	❷ S-DAG	❹ D-DAG

Metric	❶ Static tree	❷ Static DAG	❸ Dynamic tree	❹ Dynamic DAG
min cost	BU [14, 15, 16]	MTBDD [17] $\mathcal{C}$-BU [18]	BU [4]	PTA [8]
min time	BU [14, 19]	Petri nets [12]	APH [9] BU [4]	PTA [8]
min skill	BU [14, 20]	$\mathcal{C}$-BU [18]	BU [4]	—
max probability	BU [6, 19]	BDD [21] DPLL [7]	APH [9]	I/O-IMC [5]
Any of the above	**Algo. 1**: $\texttt{BU}_{\texttt{SAT}}$	**Algo. 2**: $\texttt{BDD}_{\texttt{DAG}}$	**Algo. 5**: $\texttt{BU}_{\texttt{DAT}}$	**OPEN PROBLEM**
k-top metrics	BU-projection [14]	**Algo. 3**: `BDD shortest_paths`	**OPEN PROBLEM**	**OPEN PROBLEM**

Summary of contributions

	Static	Dynamic
Tree	❶ S-tree	❸ D-tree
DAG	❷ S-DAG	❹ D-DAG

	❶	❷	❸	❹
Metric	Static tree	Static DAG	Dynamic tree	Dynamic DAG
min cost	BU [14, 15, 16]	MTBDD [17] $\mathcal{C}$-BU [18]	BU [4]	PTA [8]
min time	BU [14, 19]	Petri nets [12]	APH [9] BU [4]	PTA [8]
min skill	BU [14, 20]	$\mathcal{C}$-BU [18]	BU [4]	—
max probability	BU [6, 19]	BDD [21] DPLL [7]	APH [9]	I/O-IMC [5]
Any of the above	**Algo. 1:** $\mathtt{BU}_{\mathtt{SAT}}$	**Algo. 2:** $\mathtt{BDD}_{\mathtt{DAG}}$	**Algo. 5:** $\mathtt{BU}_{\mathtt{DAT}}$	**OPEN PROBLEM**
k-top metrics	BU-projection [14]	**Algo. 3:** `BDD shortest_paths`	**OPEN PROBLEM**	**OPEN PROBLEM**

- **NP-hard**: compute minimal successful attack in static ATs is NP-hard
- $\mathtt{BDD}_{\mathtt{DAG}}$: BDD algorithm to compute metrics for static-DAG ATs
- $\mathtt{BDD}_{\mathtt{shortest\text{-}path}}$: algorithm to compute k-top best attacks

Static

Summary of contributions

	Static	Dynamic
Tree	❶ S-tree	❸ D-tree
DAG	❷ S-DAG	❹ D-DAG

	❶	❷		❸		❹
Metric	Static tree	Static DAG		Dynamic tree		Dynamic DAG
min cost	BU [14, 15, 16]	MTBDD [17]	$\mathcal{C}$-BU [18]	BU [4]		PTA [8]
min time	BU [14, 19]	Petri nets [12]		APH [9]	BU [4]	PTA [8]
min skill	BU [14, 20]	$\mathcal{C}$-BU [18]		BU [4]		—
max probability	BU [6, 19]	BDD [21]	DPLL [7]	APH [9]		I/O-IMC [5]
Any of the above	**Algo. 1:** $\mathtt{BU_{SAT}}$	**Algo. 2:** $\mathtt{BDD_{DAG}}$		**Algo. 5:** $\mathtt{BU_{DAT}}$		OPEN PROBLEM
***k*-top metrics**	BU-projection [14]	**Algo. 3:** `BDD shortest_paths`		OPEN PROBLEM		OPEN PROBLEM

- **NP-hard**: compute minimal successful attack in static ATs is NP-hard
- $\mathtt{BDD_{DAG}}$: BDD algorithm to compute metrics for static-DAG ATs
- $\mathtt{BDD_{shortest\text{-}path}}$: algorithm to compute k-top best attacks
- **Poset semantics** (and well-formedness) for dynamic ATs
- $\mathtt{BU_{DAT}}$: Bottom-Up algorithm to compute metrics for dynamic-tree ATs
- Directions to analyse dynamic-DAG ATs (open problem)

Dyn.

UNIVERSITY OF TWENTE.

Efficient Algorithms for Quantitative Attack Tree Analysis

Carlos E. Budde[†] & Mariëlle Stoelinga[†*]

[†] Formal Methods & Tools, University of Twente, Enschede, The Netherlands

[*] Dept. of Software Science, Radboud University, Nijmegen, The Netherlands

c.e.budde@utwente.nl

m.i.a.stoelinga@utwente.nl

CSF — June 23, 2021