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

- Get PIN
- Cryptoattack
  - Pilfer notebook
  - Intercept transactions
  - Use (weak) plain RSA
Attack Tree models

- Get PIN
  - OR
    - pilfer notebook
    - intercept transactions
    - use (weak) plain RSA
  - AND
Attack Tree models

- Get PIN
  - OR
    - cryptoattack
      - AND
        - pilfer notebook
        - intercept transactions
        - use (weak) plain RSA
  - n
  - t
  - p

- Pick pocket
  - skill
  - luck
    - AND
      - fastest fingers
      - walk next to victim
      - car crash right there
    - cc

- OR

- SAND
Attack Tree models

Get PIN
- cryptoattack
  - pilfer notebook
  - intercept transactions
  - use (weak) plain RSA

Pick pocket
- skill
- luck
  - fastest fingers
  - walk next to victim
  - car crash right there

OR

AND

SAND

Basic Attack Steps

Carlos E. Budde
Attack Tree models & metrics

### Basic Attack Steps

**OR**
- Get PIN
- **AND**
  - cryptoattack
    - pilfer notebook
    - intercept transactions
    - use (weak) plain RSA
- **SAND**
  - Pick pocket
    - skill
      - fastest fingers
    - luck
      - walk next to victim
      - car crash right there

#### Time & Cost

<table>
<thead>
<tr>
<th>n</th>
<th>t</th>
<th>p</th>
<th>ff</th>
<th>w</th>
<th>cc</th>
<th>Time</th>
<th>Cost</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>40</td>
<td>0</td>
<td>0.1</td>
<td>120</td>
<td>1</td>
<td>0.07</td>
<td>0</td>
<td>0.95</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0.95</td>
</tr>
<tr>
<td>0.07</td>
<td>0.01</td>
<td>0.95</td>
<td>0.001</td>
<td>0.6</td>
<td>0.05</td>
<td>0.001</td>
<td>0.6</td>
<td>0.05</td>
</tr>
</tbody>
</table>
Attack Tree models & metrics

OR

Get PIN

AND

cryptoattack

semi-attack steps

Pick pocket

SAND

Pick pocket

skill

luck

basic attack steps

pin

t

p

use (weak) plain RSA

pick fastest fingers

walk next to victim

car crash right there

pilfer notebook

intercept transactions

Time

Cost

Prob.

40

120

0

0.07

PIN

0.1

120

1

120

0

0.01

0.95

0.001

0.6

0.05

0.07

0.01

0.95

0.001

0.6

0.05

UNIVERSITY OF TWENTE.

Carlos E. Budde
Attack Tree models & metrics

Static AT
- Get PIN
- OR
  - cryptoattack
    - piller notebook
    - intercept transactions
    - use (weak) plain RSA
  - AND
    - pilfer notebook
    - intercept transactions
    - use (weak) plain RSA

Dynamic AT
- Pick pocket
  - OR
    - skill
    - luck
      - AND
        - fastest fingers
        - walk next to victim
        - car crash right there

<table>
<thead>
<tr>
<th>PIN</th>
<th>Time</th>
<th>Cost</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07</td>
<td>0.1</td>
<td>0</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Basic Attack Steps
- 40
- 120
- 0
- 0.07

- 40
- 120
- 0
- 0

- 0
- 30
- 0
- 0

- 0.07
- 0.01
- 0.95

- 0.07
- 0.01
- 0.95

- 0.001
- 0.6
- 0.05

- 0.001
- 0.6
- 0.05

- 120
- 0
- 0

- 0
- 0
- 0

- 0.03
- 0.03

Carlos E. Budde

UNIVERSITY OF TWENTE.
Carlos E. Budde

Attack Tree models & metrics

Static AT

Get PIN

OR

雌

And

GET PIN

cryptoattack

pilfer
notebook

intercept
transactions

use (weak)
plain RSA

Dynamic AT

Pick pocket

SAND

skill

luck

fastest
fingers

walk next
to victim

car crash
right there

Time

Cost

Prob.

0.1

0.001

0.01

120

0.6

0.07

120

0.07

0.95

0.01

0.6

0.05

40

0

0.07

UNIVERSITY OF TWENTE.

1/12
**Attack Tree models & metrics**

**Static AT**
- Get PIN
- OR
  - cryptoattack
    - pilfer notebook
    - intercept transactions
    - use (weak) plain RSA
- AND
  - Time
    - 120
  - Cost
    - 40
  - Prob.
    - 0.07

**Dynamic AT**
- Pick pocket
- skill
  - fastest fingers
  - walk next to victim
  - car crash right there
- luck
- SAND
- Time
  - 0.1
  - 120
  - 1
- Cost
  - 0
  - 0
  - 0
- Prob.
  - 0.001
  - 0.6
  - 0.05

**Basic Attack Steps**
- PIN
- 40
- 0
- 0.07

**Static vs Dynamic**
- Tree: S-tree, D-tree
- DAG: S-DAG, D-DAG

**Tree**
- OR
  - S-tree
  - D-tree

**DAG**
- AND
  - S-DAG
  - D-DAG

**Metrics**
- Time: 0.1, 120, 1
- Cost: 0, 0, 0
- Prob.: 0.001, 0.6, 0.05
## Algorithms to compute metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Static tree</th>
<th>Static DAG</th>
<th>Dynamic tree</th>
<th>Dynamic DAG</th>
</tr>
</thead>
</table>

**Pareto fronts**

- BU [22, 19]  
- C-BU [11]  
- OPEN PROBLEM  
- PTA [8]

**Any of the above**

- **Algo. 1**: BU_{PAT}  
- **Algo. 2**: BDD_{DAG}  
- **Algo. 3**: BDD shortest_paths  
- **Algo. 5**: BU_{PAT}  
- OPEN PROBLEM  
- OPEN PROBLEM

**k-top metrics**

- BU-projection [14]  
- OPEN PROBLEM  
- OPEN PROBLEM  
- OPEN PROBLEM

**BU**: bottom-up on the AT structure. **APH**: acyclic phase-type (time distribution). **BDD**: binary decision diagram. **MTBDD**: multi-terminal BDD. **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).
**Definition** (AT). An *attack tree* is a tuple $T = (N, t, ch)$ where:

- $N$ is a finite set of *nodes*;
- $t: N \rightarrow \{\text{BAS, OR, AND, SAND}\}$ gives the *type* of each node;
- $ch: N \rightarrow N^*$ 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: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$. 
**Definition (AT).** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- $N$ is a finite set of *nodes*;
- $t: N 	o \{\text{BAS, OR, AND, SAND}\}$ gives the *type* of each node;
- $ch: N \to N^*$ 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: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$. 

$$T = \text{AND}(\text{OR}(a, b), \text{OR}(b, c))$$

$\text{cost}(a) = \text{cost}(c) = 2, \text{cost}(b) = 1$

$$\text{mincost}(T) = 1 \cdot \min(2, 1) + \min(1, 2) = 2$$

$\neq \frac{3}{12}$
**Definition (AT).** An *attack tree* is a tuple $T = (N, t, ch)$ where:

- $N$ is a finite set of *nodes*;
- $t: N \rightarrow \{\text{BAS, OR, AND, SAND}\}$ gives the *type* of each node;
- $ch: N \rightarrow N^*$ 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 $R_T$: $\exists! R_T \in N. \forall v \in N. R_T \notin ch(v)$;
- $\text{BAS}_T$ nodes are the leaves of: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$. 

$$T = \text{AND}(\text{OR}(a,b),\text{OR}(b,c))$$

\[\text{cost}(a) = 2, \quad \text{cost}(b) = 1, \quad \text{cost}(c) = 2.\]
**Definition (AT).** An *attack tree* is a tuple $T = (N, t, ch)$ where:

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

Moreover, $T$ satisfies the following constraints:

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

\[ T = \text{AND}(\text{OR}(a, b), \text{OR}(b, c)) \]
\[ \text{cost}(a) = \text{cost}(c) = 2, \quad \text{cost}(b) = 1 \]
\[ \text{mincost}(T) = 1 \min(2, 1) + \min(1, 2) = 2 \neq \frac{3}{12} \]
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;
- $t: N \rightarrow \{\text{BAS, OR, AND, SAND}\}$ gives the type of each node;
- $ch: N \rightarrow N^*$ 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: $\forall v \in N. \ t(v) = \text{BAS} \iff ch(v) = \epsilon$.

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

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

We let $\bar{\alpha}(T) = \bar{\alpha}([T]):$ 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*;
- $t: N \rightarrow \{\text{BAS, OR, AND, SAND}\}$ gives the *type* of each node;
- $ch: N \rightarrow N^*$ 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: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$.

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

1. an *attribution* $\alpha: \text{BAS} \rightarrow V$ assigns an *attribute value* $\alpha(a)$ to each basic attack step $a$;

2. an *attack metric* $\tilde{\alpha}: \mathcal{A}_T \rightarrow V$ assigns a value $\tilde{\alpha}(A)$ to an attack $A$;
   
   a *security metric* $\tilde{\alpha}: \mathcal{S}_T \rightarrow V$ assigns a value $\tilde{\alpha}(S)$ to a suite $S$ of $T$.

We let $\tilde{\alpha}(T) = \tilde{\alpha}(\llbracket T \rrbracket)$: the metric of an AT is given by its semantics.
**Definition** (AT). An *attack tree* is a tuple $T = (N, t, ch)$ where:
- $N$ is a finite set of nodes;
- $t: N \rightarrow \{\text{BAS, OR, AND, SAND}\}$ gives the type of each node;
- $ch: N \rightarrow N^*$ 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: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$.

**Definition** (Metric). Given an AT and a set $V$ of values:
1. an *attribution* $\alpha: \text{BAS} \rightarrow V$ assigns an attribute value $\alpha(a)$ to each basic attack step $a$;
2. an *attack metric* $\hat{\alpha}: \mathcal{A}_T \rightarrow V$ assigns a value $\hat{\alpha}(A)$ to an attack $A$; a security metric $\alpha: \mathcal{S}_T \rightarrow V$ assigns a value $\alpha(S)$ to a suite $S$ of $T$.

We let $\overline{\alpha}(T) = \overline{\alpha}(\llbracket T \rrbracket)$: the metric of an AT is given by its semantics.
Definition (AT). An attack tree is a tuple $T = (N, t, ch)$ where:

- $N$ is a finite set of nodes;
- $t : N \to \{\text{BAS, OR, AND, SAND}\}$ gives the type of each node;
- $ch : N \to N^*$ 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: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$.

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

1. an attribution $\alpha : \text{BAS} \to V$ assigns an attribute value $\alpha(a)$ to each basic attack step $a$;
2. an attack metric $\hat{\alpha} : S_T \to V$ assigns a value $\hat{\alpha}(A)$ to an attack $A$;
   a security metric $\tilde{\alpha} : S_T \to V$ assigns a value $\tilde{\alpha}(S)$ to a suite $S$ of $T$.

We let $\tilde{\alpha}(T) = \tilde{\alpha}([T])$: the metric of an AT is given by its semantics.
**Definition (AT).** An attack tree is a tuple $T = (N, t, ch)$ where:
- $N$ is a finite set of nodes;
- $t: N \rightarrow \{\text{BAS, OR, AND, SAND}\}$ gives the type of each node;
- $ch: N \rightarrow N^*$ 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: $\forall v \in N. t(v) = \text{BAS} \iff ch(v) = \varepsilon$.

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

1. an attribution $\alpha: \text{BAS} \rightarrow V$ assigns an attribute value $\alpha(a)$ to each basic attack step $a$;
2. an attack metric $\hat{\alpha}: \mathcal{A}_T \rightarrow V$ assigns a value $\hat{\alpha}(A)$ to an attack $A$; a security metric $\tilde{\alpha}: \mathcal{S}_T \rightarrow V$ assigns a value $\tilde{\alpha}(S)$ to a suite $S$ of $T$.

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

\[ T = \text{AND}(\text{OR}(a,b),\text{OR}(b,c)) \]

\[ \text{cost}(a) = \text{cost}(c) = 2, \text{cost}(b) = 1 \]

\[ \text{mincost}(T) = \min(2,1) + \min(1,2) = 2 \]

\[ \alpha \]

\[ V = \{3, 1, 4\} \]

\[ \hat{\alpha}(\text{“do } a \text{ and } c\text{”}) = 7 \]

\[ \tilde{\alpha}(\text{“any } T \text{ attack”}) = 1 \]
Static AT semantics (no order in attacks)

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

\[
f_T(v, A) = \begin{cases} 
\top & \text{if } t(v) = \text{OR and } \exists u \in ch(v). f_T(u, A) = \top, \\
\top & \text{if } t(v) = \text{AND and } \forall u \in ch(v). f_T(u, A) = \top, \\
\top & \text{if } t(v) = \text{BAS and } v \in A, \\
\bot & \text{otherwise.}
\end{cases}
\]
Static AT semantics (no order in attacks)

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

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

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

\[ \llbracket T \rrbracket = \{ A \subseteq \text{BAS} | f_T(A) \land A \text{ is minimal} \} \]

Theorem: computing \( \llbracket T \rrbracket \) is an **NP-complete** problem
Static AT metrics

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

\[
T = \text{AND}(\text{OR}(a, b), \text{OR}(b, c))
\]

\[
\begin{align*}
\text{cost}(a) &= 2, \\
\text{cost}(c) &= 2, \\
\text{cost}(b) &= 1
\end{align*}
\]

\[
\min\text{cost}(T) = \min(2, 1) + \min(1, 2) = 2
\]

$[T] = \{\{b\}, \{a, c\}\}$
Static AT metrics

- An **attribute domain** is a tuple $D = (V, \nabla, \Delta)$ where:
  - $\nabla: V^2 \rightarrow V$ is a **disjunctive** operator
  - $\Delta: V^2 \rightarrow V$ is a **conjunctive** operator

\[
T = \text{AND}(\text{OR}(a,b), \text{OR}(b,c))
\]

<table>
<thead>
<tr>
<th>Metric</th>
<th>$V$</th>
<th>$\nabla$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>min cost</td>
<td>$\mathbb{N}_\infty$</td>
<td>min</td>
<td>+</td>
</tr>
<tr>
<td>min time</td>
<td>$\mathbb{N}_\infty$</td>
<td>min</td>
<td>+</td>
</tr>
<tr>
<td>min skill</td>
<td>$\mathbb{N}_\infty$</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>max challenge</td>
<td>$\mathbb{N}_\infty$</td>
<td>max</td>
<td>max</td>
</tr>
<tr>
<td>max damage</td>
<td>$\mathbb{N}_\infty$</td>
<td>max</td>
<td>+</td>
</tr>
<tr>
<td>discrete prob.</td>
<td>$[0,1]_Q$</td>
<td>max</td>
<td>*</td>
</tr>
<tr>
<td>continu. prob.</td>
<td>$\mathbb{R} \rightarrow [0,1]_Q$</td>
<td>max</td>
<td>*</td>
</tr>
</tbody>
</table>

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

Carlos E. Budde
An attribute domain is a tuple $D = (V, \nabla, \Delta)$ where:
- $\nabla: V^2 \rightarrow V$ is a disjunctive operator
- $\Delta: V^2 \rightarrow V$ is a conjunctive operator

The metric for a static AT $T$, attribution $\alpha$, and domain $D$ is:

$$\tilde{\alpha}(T) = \bigtriangledown \bigtriangleup_{a \in A} \alpha(a)$$

$T$ = $\{(b), \{a, c\}\}$
Static AT metrics

- An **attribute domain** is a tuple $D = (V, \nabla, \Delta)$ where:
  - $\nabla : V^2 \rightarrow V$ is a **disjunctive** operator
  - $\Delta : V^2 \rightarrow V$ is a **conjunctive** operator

- The **metric for a static AT** $T$, attribution $\alpha$, and domain $D$ is:
  \[
  \tilde{\alpha}(T) = \bigtriangleup_{a \in \hat{\alpha}} \bigtriangledown_{A \in [T]} \alpha(a)
  \]

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

- $D = (V, \nabla, \Delta) = (\mathbb{N}, \min, +)$

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

**Table:**

<table>
<thead>
<tr>
<th>Metric</th>
<th>$V$</th>
<th>$\nabla$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>min cost</td>
<td>$\mathbb{N}_\infty$</td>
<td>$\min$</td>
<td>$+$</td>
</tr>
<tr>
<td>min time</td>
<td>$\mathbb{N}_\infty$</td>
<td>$\min$</td>
<td>$+$</td>
</tr>
<tr>
<td>min skill</td>
<td>$\mathbb{N}_\infty$</td>
<td>$\min$</td>
<td>$\max$</td>
</tr>
<tr>
<td>max challenge</td>
<td>$\mathbb{N}_\infty$</td>
<td>$\max$</td>
<td>$\max$</td>
</tr>
<tr>
<td>max damage</td>
<td>$\mathbb{N}_\infty$</td>
<td>$\max$</td>
<td>$+$</td>
</tr>
<tr>
<td>discrete prob.</td>
<td>$[0, 1]_\mathbb{Q}$</td>
<td>$\max$</td>
<td>$*$</td>
</tr>
<tr>
<td>continu. prob.</td>
<td>$\mathbb{R} \rightarrow [0, 1]_\mathbb{Q}$</td>
<td>$\max$</td>
<td>$*$</td>
</tr>
</tbody>
</table>
Static AT metrics

- An **attribute domain** is a tuple $D = (V, \nabla, \Delta)$ where:
  - $\nabla : V^2 \rightarrow V$ is a **disjunctive** operator
  - $\Delta : V^2 \rightarrow V$ is a **conjunctive** operator

- The **metric for a static AT** $T$, attribution $\alpha$, and domain $D$ is:
  
  $\hat{\alpha}(T) = \bigtriangleup_{A \in [T]} \bigwedge_{a \in A} \alpha(a)$

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

- $D = (V, \nabla, \Delta) = (\mathbb{N}, \min, +)$
  
  **min cost**

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

  $\hat{\alpha}(T) = \bigtriangleup_{A \in [T]} \bigwedge_{a \in A} \alpha(a)$

  $= \left(\alpha(b)\right) \nabla \left(\alpha(a) \Delta \alpha(c)\right)$

  $= (1) \min (3 + 4)$

  $= 1$

---

<table>
<thead>
<tr>
<th>Metric</th>
<th>$V$</th>
<th>$\nabla$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>min cost</td>
<td>$\mathbb{N}_\infty$</td>
<td>min</td>
<td>+</td>
</tr>
<tr>
<td>min time</td>
<td>$\mathbb{N}_\infty$</td>
<td>min</td>
<td>+</td>
</tr>
<tr>
<td>min skill</td>
<td>$\mathbb{N}_\infty$</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>max challenge</td>
<td>$\mathbb{N}_\infty$</td>
<td>max</td>
<td>max</td>
</tr>
<tr>
<td>max damage</td>
<td>$\mathbb{N}_\infty$</td>
<td>max</td>
<td>*</td>
</tr>
<tr>
<td>discrete prob.</td>
<td>$[0, 1]_Q$</td>
<td>max</td>
<td>*</td>
</tr>
<tr>
<td>continu. prob.</td>
<td>$\mathbb{R} \rightarrow [0, 1]_Q$</td>
<td>max</td>
<td>*</td>
</tr>
</tbody>
</table>
$T$ is a static tree

- Domain $D = (V, \sqcap, \triangleright)$ is a semiring if $\triangleright$ distributes over $\sqcap$
\( T \) is a static tree

- Domain \( D = (V, \nabla, \Delta) \) is a **semiring** if \( \Delta \) distributes over \( \nabla \)

\( T \) is a static tree

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

**BU_{\text{SAT}}** algorithm, linear in \( T \)

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

**Output:** Metric value \( \tilde{\alpha}(T) \in V \).

```plaintext
if \( t(v) = \text{OR} \) then
  return \( \bigtriangledown_{u \in ch(v)} \text{BU}_{\text{SAT}}(T, u, \alpha, D) \)
else if \( t(v) = \text{AND} \) then
  return \( \Delta_{u \in ch(v)} \text{BU}_{\text{SAT}}(T, u, \alpha, D) \)
else // \( t(v) = \text{BAS} \)
  return \( \alpha(v) \)
```

$T$ is a static tree

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

**BU_{SAT} algorithm, linear in $T$**

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

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

if $t(v) = \text{OR}$ then
  return $\bigtriangledown_{u \in ch(v)} BU_{SAT}(T, u, \alpha, D)$
else if $t(v) = \text{AND}$ then
  return $\Delta_{u \in ch(v)} BU_{SAT}(T, u, \alpha, D)$
else // $t(v) = \text{BAS}$
  return $\alpha(v)$

**Theorem.** Let $T$ be a static AT with tree structure, $\alpha$ an attribution on $V$, and $D = (V, \triangledown, \Delta)$ a semiring attribute domain.

Then $\tilde{\alpha}(T) = BU_{SAT}(T, R_T, \alpha, D)$.

$T$ is a static DAG

$\tilde{\alpha}(T) = 1 \quad \{3, 1, 4\}$

$BU_{SAT}(T) = (3 \text{ min } 1) + (1 \text{ min } 4) = 2$
$T$ is a static DAG

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

$\text{BU}_{\alpha\lambda}(T) = (3 \text{ min } 1) + (1 \text{ min } 4) = 2$
$T$ is a static DAG

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

\[
\text{BAS}_T \cup \{1, 0\} \quad f_T
\]

\[
\tilde{\alpha}(T) = 1 \quad \{3, 1, 4\}
\]

\[
\text{BU}_{\text{MIN}}(T) = (3 \text{ min } 1) + (1 \text{ min } 4) = 2
\]
$T$ is a static DAG

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

$$BAS_T \cup \{1, 0\} \quad f_T$$

$\begin{array}{c}
\text{min}\{3, 1, 4\} \\
\text{b < a < c}
\end{array}$

failure success
$T$ is a static DAG

- **Binary Decision Diagram (BDD)**

  $B_T = (W, \text{Lab}, \text{Low}, \text{High})$

  \[ \text{BDD}_{\text{SAT}} \text{ algorithm, linear in } B_T \]

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

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

  if $\text{Lab}(w) = 0$ then
  
  return $1_V$

  else if $\text{Lab}(w) = 1$ then
  
  return $1_\Delta$

  else // either do $\text{Lab}(w) = v \in \text{BAS}$, or not

  return $(\alpha(\text{Lab}(w)) \Delta \cdots

  \cdots \text{BDD}_{\text{SAT}}(B_T, \text{High}(w), \alpha, D_*))

  \nabla \text{BDD}_{\text{SAT}}(B_T, \text{Low}(w), \alpha, D_*)$
$T$ is a static DAG

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

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

  **Input:** BDD $B_T = (W, Low, High, Lab)$,
  node $w \in W$,
  attribution $\alpha$,
  semiring attribute domain $D_\ast = (V, \vee, \wedge, 1_\vee, 1_\wedge)$.

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

  if $\text{Lab}(w) = 0$ then
  return $1_\vee$
  else if $\text{Lab}(w) = 1$ then
  return $1_\wedge$
  else // either do $\text{Lab}(w) = v \in \text{BAS}$, or not
  return $\bar{\alpha}(\text{Lab}(w)) \wedge \cdots$
   
   $\cdots \text{BDD}\textsubscript{SAT}(B_T, \text{High}(w), \alpha, D_\ast))$
  $\vee \text{BDD}\textsubscript{SAT}(B_T, \text{Low}(w), \alpha, D_\ast))$

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

  \[
  T = \text{AND}(\text{OR}(a, b), \text{OR}(b, c))
  \]

  cost
  \[
  \begin{align*}
  \text{cost}(a) &= 2, \\
  \text{cost}(c) &= 2, \\
  \text{cost}(b) &= 1
  \end{align*}
  \]

  $\min_{\text{cost}}(T) = 1 \min(2, 1) + \min(1, 2) = 2$

  $b < a < c$

  \[
  \{ 3, 1, 4 \}
  \]
$T$ is a static DAG

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

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

  **Input:** BDD $B_T = (W, Low, High, Lab)$,
  node $w \in W$,
  attribution $\alpha$,
  semiring attribute domain $D_{\star} = (V, \triangledown, \triangledown_{\star}, \mathbb{1}_{\triangledown}, \mathbb{1}_{\triangledown_{\star}})$.

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

  if $Lab(w) = 0$ then
  \[ \text{return } \mathbb{1}_{\triangledown} \]
  else if $Lab(w) = 1$ then
  \[ \text{return } \mathbb{1}_{\triangledown_{\star}} \]
  else // either do $Lab(w) = v \in BAS$, or not
  \[ \text{return } (\alpha(Lab(w)) \triangledown \cdots \cdots BDD_{\text{SAT}}(B_T, High(w), \alpha, D_{\star})) \]
  \[ \triangledown BDD_{\text{SAT}}(B_T, Low(w), \alpha, D_{\star}) \]

  do not attack \( v \)

  $$f_T = \min_{a < b < c} \{ 3, 1, 4 \}$$

  $$\text{mincost}(T) = 1 \min(2, 1) + \min(1, 2) = 2$$
\( T \) is a static DAG

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

  \[ \text{BDD}_{\text{SAT}} \text{ algorithm, linear in } B_T \]

  **Input:** BDD \( B_T = (W, \text{Low}, \text{High}, \text{Lab}) \),
  node \( w \in W \),
  attribution \( \alpha \),
  semiring attribute domain
  \( D_* = (V, \sqcap, \Delta, 1_{\sqcap}, 1_{\Delta}) \).

  **Output:** Metric value \( \bar{\alpha}(T) \in V \).

  ```
  if Lab(w) = 0 then
    return 1_{\sqcap}
  else if Lab(w) = 1 then
    return 1_{\Delta}
  else // either do Lab(w) = v ∈ BAS, or not
    return (\alpha(Lab(w)) \sqcap \cdots
    \cdots \text{BDD}_{\text{SAT}}(B_T, \text{High}(w), \alpha, D_*))
    \sqcup \text{BDD}_{\text{SAT}}(B_T, \text{Low}(w), \alpha, D_*)
  ```

- \( T = \text{AND}((\text{OR}(a,b)), (\text{OR}(b,c))) \)
  
  \( \text{cost}(a) = 2 \), \( \text{cost}(c) = 2 \), \( \text{cost}(b) = 1 \)
  
  \( \text{mincost}(T) = 1 \text{ min}(2,1) + \text{min}(1,2) = 2 \neq \text{min} \)

- **Binary Decision Diagram (BDD)**
  
  \( a \quad b \quad c \)

  \( T = \text{AND}((\text{OR}(a,b)), (\text{OR}(b,c))) \)

  \( b < a < c \)

  \( \{ 3, 1, 4 \} \)

  \( \text{success/failure} \)
**$T$ is a static DAG**

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

**BDD\text{sat} algorithm, linear in $B_T$**

**Input:** BDD $B_T = (W, \text{Low}, \text{High}, \text{Lab})$
- node $w \in W$
- attribution $\alpha$
- semiring attribute domain $D_* = (V, \triangledown, \Delta, 1_\triangledown, 1_\Delta)$

**Output:** Metric value $\alpha(T) \in V$.

\[
\text{if } \text{Lab}(w) = 0 \text{ then} \\
\quad \text{return } 1_\triangledown \\
\text{else if } \text{Lab}(w) = 1 \text{ then} \\
\quad \text{return } 1_\Delta \\
\text{else} \quad \text{either do Lab}(w) = v \in \text{BAS}, \text{or not} \\
\quad \text{return } (\alpha(\text{Lab}(w)) \Delta \cdots \\
\quad \quad \cdots \text{BDD}\text{sat}(B_T, \text{High}(w), \alpha, D_*) \\
\quad \quad \triangledown \text{BDD}\text{sat}(B_T, \text{Low}(w), \alpha, D_*) )
\]

\[D = (V, \triangledown, \Delta, 1_\triangledown, 1_\Delta) = (\text{cost}, \text{min}, +, \infty, 0)\]

\[\text{BDD}\text{sat}(w_b) = \]

\[
\begin{array}{c}
\text{lab} \\
b \\
\text{a} \\
\text{c} \\
0 \\
1
\end{array}
\]

\[
\begin{array}{c}
\text{min} \\
\text{min} \\
\text{b} < a < c \\
\{ 3 , 1 , 4 \}
\end{array}
\]
\( T \) is a static DAG

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

**BDD\textsubscript{SAT} algorithm, linear in \( B_T \)**

**Input:** BDD \( B_T = (W, \text{Low, High, Lab}), \)

node \( w \in W, \)

attribute \( \alpha, \)

semiring attribute domain \( D_* = (V, \nabla, \triangle, 1_{\nabla}, 1_{\triangle}). \)

**Output:** Metric value \( \bar{\alpha}(T) \in V. \)

\[
\begin{align*}
\text{if } \text{Lab}(w) = 0 & \text{ then} \\
& \text{return } l_{\nabla} \\
\text{else if } \text{Lab}(w) = 1 & \text{ then} \\
& \text{return } l_{\triangle} \\
\text{else } // \text{ either do } \text{Lab}(w) = v \in \text{BAS, or not} \\
& \text{return } (\alpha(\text{Lab}(w)) \triangle \cdots \\
& \cdots \text{BDD}_{\text{SAT}}(B_T, \text{High}(w), \alpha, D_*)) \\
& \nabla \text{BDD}_{\text{SAT}}(B_T, \text{Low}(w), \alpha, D_*)
\end{align*}
\]

\[
D = (V, \nabla, \triangle, 1_{\nabla}, 1_{\triangle}) = (\text{cost, min, +, } \infty, 0)
\]

\[
\text{BDD}_{\text{SAT}}(w_b) = (1 + 0) \min \text{BDD}_{\text{SAT}}(w_a)
\]
\( T \) is a static DAG

- Binary Decision Diagram (BDD): \( B_T = (W, \text{Lab}, \text{Low}, \text{High}) \)

**BDD\(_\text{SAT}\) algorithm, linear in \( B_T \)**

**Input:** BDD \( B_T = (W, \text{Low}, \text{High}, \text{Lab}) \), node \( w \in W \), attribution \( \alpha \), semiring attribute domain \( D_* = (V, \triangledown, \Delta, 1_\triangledown, 1_\Delta) \).

**Output:** Metric value \( \bar{\alpha}(T) \in V \).

\[
\text{if } \text{Lab}(w) = 0 \; \text{then} \\
\quad \text{return } 1_\triangledown \\
\text{else if } \text{Lab}(w) = 1 \; \text{then} \\
\quad \text{return } 1_\Delta \\
\text{else} // \text{either do } \text{Lab}(w) = v \in \text{BAS}, \text{or not} \\
\quad \text{return } (\alpha(\text{Lab}(w)) \Delta \cdots \Delta \text{BDD}\(_\text{SAT}\)(B_T, \text{High}(w), \alpha, D_*)) \\
\quad \triangledown \text{BDD}\(_\text{SAT}\)(B_T, \text{Low}(w), \alpha, D_*)
\]

\[
D = (V, \triangledown, \Delta, 1_\triangledown, 1_\Delta) = (\text{cost, min, +, } \infty, 0)
\]

\[
\text{BDD}\(_\text{SAT}\)(w_b) = (1 + 0) \min \text{BDD}\(_\text{SAT}\)(w_a) \\
= (1) \min ((3 + \text{BDD}\(_\text{SAT}\)(w_c)) \min \infty)
\]

\[
= \begin{cases} 
3, & b < a < c \\
1, & \{3, 1, 4\} \\
4, & \text{not } b < a < c 
\end{cases}
\]
$T$ is a static DAG

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

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

**Input:** BDD $B_T = (W, \text{Low}, \text{High}, \text{Lab})$, node $w \in W$, attribution $\alpha$, semiring attribute domain $D_s = (V, \triangledown, \triangle, 1_V, 1_\triangle)$.

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

if $\text{Lab}(w) = 0$ then
  return $1_V$
else if $\text{Lab}(w) = 1$ then
  return $1_\Delta$
else
  // either do $\text{Lab}(w) = v \in \text{BAS}$, or not
  return $\left( \alpha(\text{Lab}(w)) \triangle \cdots \right.$
  $\left. \cdots \text{BDD}_{\text{SAT}}(B_T, \text{High}(w), \alpha, D_s) \right) \triangledown \text{BDD}_{\text{SAT}}(B_T, \text{Low}(w), \alpha, D_s)$

\[
D = (V, \triangledown, \triangle, 1_V, 1_\triangle) = (\text{cost}, \min, +, \infty, 0)
\]

\[
\text{BDD}_{\text{SAT}}(w_b) = (1 + 0) \min \text{BDD}_{\text{SAT}}(w_a) = (1) \min ((3 + \text{BDD}_{\text{SAT}}(w_c)) \min \infty) = (1) \min (3 + ((4 + 0) \min \infty)) = 1
\]
**2** \( T \) is a static DAG

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

\[
\text{BDD}_{\text{SAT}} \text{ algorithm, linear in } B_T
\]

**Input:** BDD \( B_T = (W, Low, High, Lab) \),
node \( w \in W \),
attribute function \( \alpha \),
semiring attribute domain
\( D_* = (V, \land, \lor, 1_{\land}, 1_{\lor}) \).

**Output:** Metric value \( \tilde{\alpha}(T) \in V \).

if \( \text{Lab}(w) = 0 \) then
  return \( 1_\land \)
else if \( \text{Lab}(w) = 1 \) then
  return \( 1_\lor \)
else // either do \( \text{Lab}(w) = v \in \text{BAS} \), or not
  return \( \alpha(\text{Lab}(w)) \land \ldots \land \text{BDD}_{\text{SAT}}(B_T, \text{High}(w), \alpha, D_*) \land \text{BDD}_{\text{SAT}}(B_T, \text{Low}(w), \alpha, D_*) \land \text{BDD}_{\text{SAT}}(B_T, \text{Lab}(w), \alpha, D_*) \)

\[
\begin{align*}
1_\land &= \infty \\
0 &= 1_\lor
\end{align*}
\]

\[
D = (V, \lor, \land, 1_\lor, 1_\land) = (\text{cost, min, +, } \infty, 0)
\]

\[
\text{BDD}_{\text{SAT}}(w_b) = (1 + 0) \min \text{BDD}_{\text{SAT}}(w_a)
\]

\[
= (1) \min ((3 + \text{BDD}_{\text{SAT}}(w_c)) \min \infty)
\]

\[
= (1) \min (3 + ((4 + 0) \min \infty)) = 1
\]

**Theorem.** Let \( T \) be a static AT, \( B_T \) its BDD encoding, \( \alpha \) an attribution on \( V \), and \( D_* = (V, \lor, \land, 1_\lor, 1_\land) \) a semiring attr. dom. with neutral elements resp. for \( \lor \) and \( \land \).

Then \( \tilde{\alpha}(T) = \text{BDD}_{\text{SAT}}(B_T, R_B, \alpha, D_*) \).
Dynamic AT semantics  (order in attacks)
Dynamic AT semantics (order in attacks)

- An **attack** is a partially ordered set: $\langle A, \prec \rangle$
- $a \prec b$ iff $a \in \text{BAS}$ must finish before $b$ begins
Dynamic AT semantics (order in attacks)

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

- The **ordering graph** $G_T = (\text{BAS}_T, \to)$ of a dynamic AT has the edge $a \to b \iff \exists \text{SAND}(v_1, \ldots, v_n) \text{ s.t. } a \in \text{BAS}(v_i) \land b \in \text{BAS}(v_{i+1})$

- 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 \( a \in \text{BAS} \) must finish before \( b \) begins

- The **ordering graph** \( G_T = (\text{BAS}_T, \rightarrow) \) of a dynamic AT has the edge \( a \rightarrow b \) iff \( \exists \text{SAND}(v_1, \ldots, v_n) \) s.t. \( a \in \text{BAS}(v_i) \land b \in \text{BAS}(v_{i+1}) \)

- 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 \( a \in \text{BAS} \) must finish before \( b \) begins

• The **ordering graph** \( G_T = (\text{BAS}_T, \rightarrow) \) of a dynamic AT
  
  has the edge \( a \rightarrow b \) iff \( \exists \text{SAND}(v_1, \ldots, v_n) \) s.t. \( a \in \text{BAS}(v_i) \land b \in \text{BAS}(v_{i+1}) \)

• 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 \( a \in \text{BAS} \) must finish before \( b \) begins

- The **ordering graph** \( G_T = (\text{BAS}_T, \rightarrow) \) of a dynamic AT has the edge \( a \rightarrow b \) iff \( \exists \ \text{SAND}(v_1, \ldots, v_n) \) s.t. \( a \in \text{BAS}(v_i) \land b \in \text{BAS}(v_{i+1}) \)

- 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 \( A \subseteq \text{BAS} \)) of the edges of \( G_T \)
• An **attack** is a partially ordered set: \( \langle A, \prec \rangle \)
  • \( a \prec b \iff a \in \text{BAS} \text{ must finish before } b \text{ begins} \)

• The **ordering graph** \( G_T = (\text{BAS}_T, \to) \) of a dynamic AT has the edge \( a \to b \iff \exists \text{SAND}(v_1, \ldots, v_n) \text{ s.t. } a \in \text{BAS}(v_i) \wedge b \in \text{BAS}(v_{i+1}) \)

• 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 \( A \subseteq \text{BAS} \)) 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 \( A \subseteq \text{BAS} \) and \( \prec \subseteq \text{BAS}^2 \) are minimal
A **dynamic attribute domain** is a tuple $D = (V, \triangleright, \triangle, \triangleright)$

- $\triangleright: V^2 \rightarrow V$ is a **sequential** operator

  For the ordered steps. Also associative & commutative.

\[
[T] = \{\{ff, w\}, \emptyset\}, \{\{w, cc\}, \{w \prec cc\}\}\} 
\]
Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \nabla, \Delta, \triangleright)$
- $\triangleright : V^2 \rightarrow V$ is a **sequential** operator
  For the ordered steps. Also associative & commutative.

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

$$\hat{\alpha}(T) = \bigtriangleup_{\langle A, \prec \rangle \in \llbracket T \rrbracket} \bigtriangleright_{\hat{\alpha}} \Delta_{\bar{\alpha}} \nabla_{\hat{\alpha}} \alpha(a)$$

- Pick pocket

- $[T] = \{\{ff, w\}, \emptyset\}$
  $, \{\{w, cc\}, \{w < cc\}\}$
Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \nabla, \Delta, \triangleright)$
  - $\triangleright : V^2 \to V$ is a **sequential** operator
    - For the ordered steps. Also associative & commutative.

- The **metric for a dynamic AT** $T$, attribution $\alpha$, and domain $D$ is:
  $$
  \tilde{\alpha}(T) = \bigtriangleup_{a \in C} \bigtriangleright_{c \in H_A} \bigtriangledown_{(a,x) \in \tau[T]} \alpha(a)
  $$

  **Example:**
  - Pick pocket
  - $[T] = \{ \{(ff, w), \emptyset\}, \{(w, cc), \{w < cc\}\} \}$
  - $D = (V, \nabla, \Delta, \triangleright) = (\mathbb{N}, \min, \max, +)$
  - **min time**
Dynamic AT metrics

- A **dynamic attribute domain** is a tuple $D = (V, \sqcap, \triangle, \triangleright)$
- $\triangleright: V^2 \rightarrow V$ is a **sequential** operator
  
  For the ordered steps. Also associative & commutative.

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

  $$\tilde{\alpha}(T) = \bigtriangleup_{A \in [T]} \bigtriangledown_{C \in H_A^c} \triangleright_{a \in C} \alpha(a)$$

  
  $[T] = \{ \langle \{ \text{ff, } w \}, \emptyset \rangle, \{ \{ w, cc \}, \{ w < cc \} \} \}$

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

  \[ \begin{align*}
  \tilde{\alpha}(T) &= \bigtriangleup_{A \in [T]} \bigtriangledown_{C \in H_A^c} \triangleright_{a \in C} \alpha(a) \\
  &= (\alpha(\text{ff}) \sqcap \alpha(w)) \sqcap (\alpha(w) \triangleright \alpha(cc)) \\
  &= (3 \max 15) \min (15 + 1) \\
  &= 15
  \end{align*} \]

  **min time**
$T$ is a dynamic tree

- $D = (V, \triangledown, \triangle, \triangleright)$ is a **semiring** if $\triangle$ distributes over $\triangledown$, and $\triangleright$ over $\triangledown$ and $\triangle$
$T$ is a dynamic tree

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

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

**Input:** S-tree $T = (N, t, ch)$,
- node $v \in N$,
- attribution $\alpha$,
- semiring dynamic attr. dom.
- $D = (V, \nabla, \Delta, \triangleright)$.

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

```plaintext
if $t(v) = \text{OR}$ then
  return $\bigtriangledown_{u \in ch(v)} \text{BU}_{DAT}(T, u, \alpha, D)$
else if $t(v) = \text{AND}$ then
  return $\bigtriangleup_{u \in ch(v)} \text{BU}_{DAT}(T, u, \alpha, D)$
else if $t(v) = \text{SAND}$ then
  return $\bigtriangledown_{u \in ch(v)} \text{BU}_{DAT}(T, u, \alpha, D)$
else // $t(v) = \text{SAS}$
  return $\alpha(v)$
```
\( T \) is a dynamic tree

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

**BU\(_{\text{DAT}}\) algorithm, linear in \( T \)**

**Input:** S-tree \( T = (N, t, \text{ch}) \),
node \( v \in N \),
attribute \( \alpha \),
semiring dynamic attr. dom.
\( D = (V, \triangledown, \Delta, \triangleright) \).

**Output:** Metric value \( \bar{\alpha}(T) \in V \).

```
if \( t(v) = \text{OR} \) then
  return \( \bigtriangledown_{u \in \text{ch}(v)} \text{BU}_{\text{DAT}}(T, u, \alpha, D) \)
else if \( t(v) = \text{AND} \) then
  return \( \bigtriangleup_{u \in \text{ch}(v)} \text{BU}_{\text{DAT}}(T, u, \alpha, D) \)
else if \( t(v) = \text{SAND} \) then
  return \( \bigtriangleright_{u \in \text{ch}(v)} \text{BU}_{\text{DAT}}(T, u, \alpha, D) \)
else // \( t(v) = \text{BAS} \)
  return \( \alpha(v) \)
```
$T$ is a dynamic tree

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

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

**Input:** S-tree $T = (N, t, ch)$,
- node $v \in N$,
- attribution $\alpha$,
- semiring dynamic attr. dom.
- $D = (V, \nabla, \Delta, \triangleright)$.

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

```python
if $t(v)$ = OR then
    return $\bigtriangleup$ $u \in ch(v) \text{BU}_{DAT}(T, u, \alpha, D)$
else if $t(v)$ = AND then
    return $\bigtriangledown$ $u \in ch(v) \text{BU}_{DAT}(T, u, \alpha, D)$
else if $t(v)$ = SAND then
    return $\triangleright$ $u \in ch(v) \text{BU}_{DAT}(T, u, \alpha, D)$
else // $t(v)$ = BAS
    return $\alpha(v)$
```

**Theorem.** Let $T$ be a dynamic AT with tree structure, $\alpha$ an attribution on $V$, and $D = (V, \nabla, \Delta, \triangleright)$ a semiring dyn. attr. dom. Then $\tilde{\alpha}(T) = \text{BU}_{DAT}(T, R_T, \alpha, D)$. 
$T$ is a dynamic DAG

Pick pocket

ff w cc
$T$ is a dynamic DAG

Pick pocket

Bottom-Up

BDD

ff \quad w \quad cc
T is a dynamic DAG

\[ \tilde{\alpha}(T) = \bigtriangleup_{\langle A, \prec \rangle \in [T]} \bigtriangleup_{C \in H_A^\prec} \triangleright_{a \in C} \alpha(a) \]

\([T]\) can be computed from the ordering graph \(G_T\) and the semantics of the static transform \(T'\)
$T$ is a dynamic DAG

\[
\bar{\alpha}(T) = \bigtriangleup_{\langle A, i \rangle \in [T]} \bigtriangleup_{C \in H_A^i} \bigtriangledown_{a \in C} \alpha(a)
\]

$[T]$ 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 :'

DOI: [10.1177/1748006X20974187](https://doi.org/10.1177/1748006X20974187)
# Summary of contributions

<table>
<thead>
<tr>
<th>Metric</th>
<th>Static tree</th>
<th>Static DAG</th>
<th>Dynamic tree</th>
<th>Dynamic DAG</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Any of the above</th>
<th>Algo. 1: (BU_{\text{SAT}})</th>
<th>Algo. 2: (BDD_{\text{DAG}})</th>
<th>Algo. 5: (BU_{\text{DAG}})</th>
<th>OPEN PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k)-top metrics</td>
<td>BU-projection [14]</td>
<td>Algo. 3: BDD shortest_paths</td>
<td>OPEN PROBLEM</td>
<td>OPEN PROBLEM</td>
</tr>
</tbody>
</table>
### Summary of contributions

<table>
<thead>
<tr>
<th>Metric</th>
<th>Static tree</th>
<th>Static DAG</th>
<th>Dynamic tree</th>
<th>Dynamic DAG</th>
</tr>
</thead>
</table>

Any of the above: **Algo. 1: BU_{SAT}**  **Algo. 2: BDD_{DAG}**  **Algo. 5: BU_{SAT}**  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
- **BDD_{DAG}**: BDD algorithm to compute metrics for static-DAG ATs
- **BDD_{shortest-path}**: algorithm to compute \( k \)-top best attacks

---

Carlos E. Budde
Summary of contributions

**Static**

- **Tree**: S-tree
- **DAG**: S-DAG

**Dynamic**

- **Tree**: D-tree
- **DAG**: D-DAG

<table>
<thead>
<tr>
<th>Metric</th>
<th>Static tree</th>
<th>Static DAG</th>
<th>Dynamic tree</th>
<th>Dynamic DAG</th>
</tr>
</thead>
</table>

**Any of the above**

- **Algo. 1**: BU_{SAT}
- **Algo. 2**: BDD_{DAG}
- **Algo. 5**: BU_{SAT}

**k-top metrics**

- BU-projection [14]
- Algo. 3: BDD shortest_paths

**NP-hard**: compute minimal successful attack in static ATs is NP-hard

**BDD_{DAG}**: BDD algorithm to compute metrics for static-DAG ATs

**BDD_{shortest-path}**: algorithm to compute $k$-top best attacks

**Poset semantics** (and well-formedness) for dynamic ATs

**BU_{DAT}**: Bottom-Up algorithm to compute metrics for dynamic-tree ATs

**Directions to analyse dynamic-DAG ATs** (open problem)
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