Model Checking Multiple Logs

A. Arasteh M. Debbabi A. Sakha M. Saleh

Concordia University, Montreal, Canada

The Seventh Annual Digital Forensic Research Workshop

Presented by: Mohamed Saleh

Outline

Objectives

Problem statement:

Given an IT system with logs collected from various heterogeneous sources, can we reconstruct the events leading to a system failure.

We need to clarify what properties we expect from:

- Logs.
- Heterogenous sources.
- Events.

Clarifications

Logs

A log is a *stored* sequence of *events*. It originates from a certain system, and each event is a string containing information about a certain action of interest that occurred in this system.

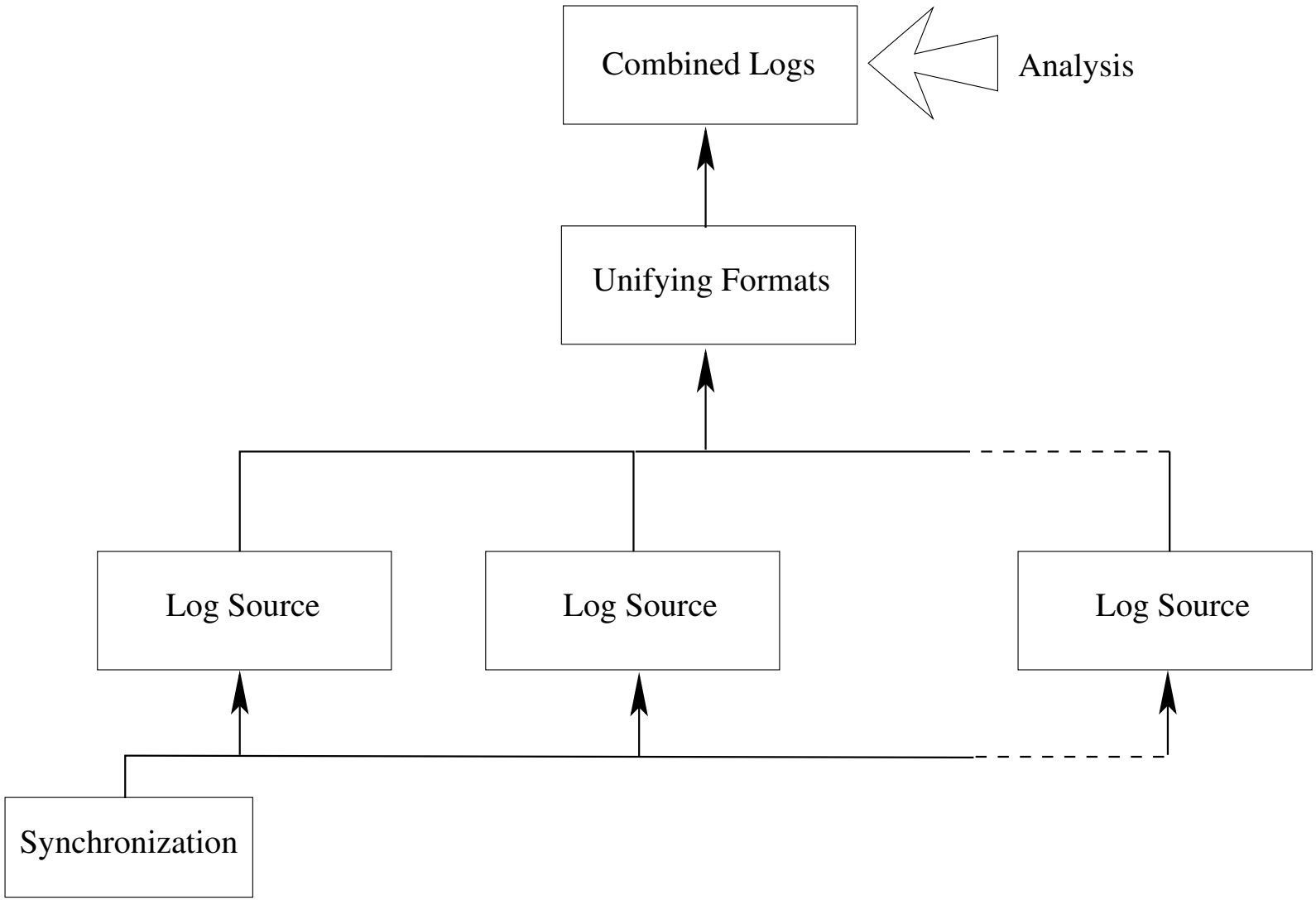
Heterogeneous sources of logs

Logs may originate from various systems such as different operating systems or different networking equipment. These systems may have different rules for generating, storing or transmitting logs.

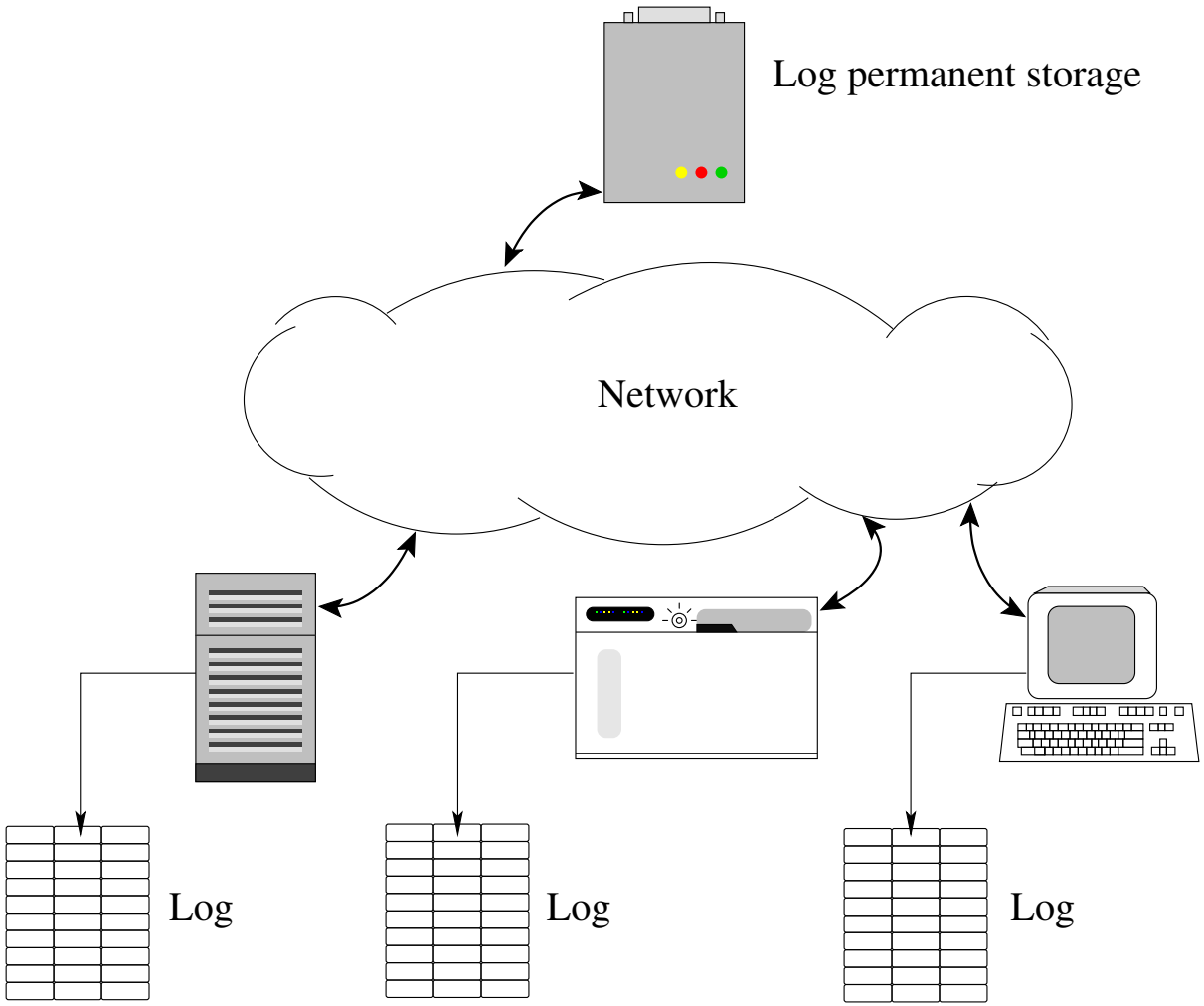
Log events

An event is a string that describes an action that occurred in the log source. It must have a *time-stamp* and contain information about the action and the identity of the entity that carried it out.

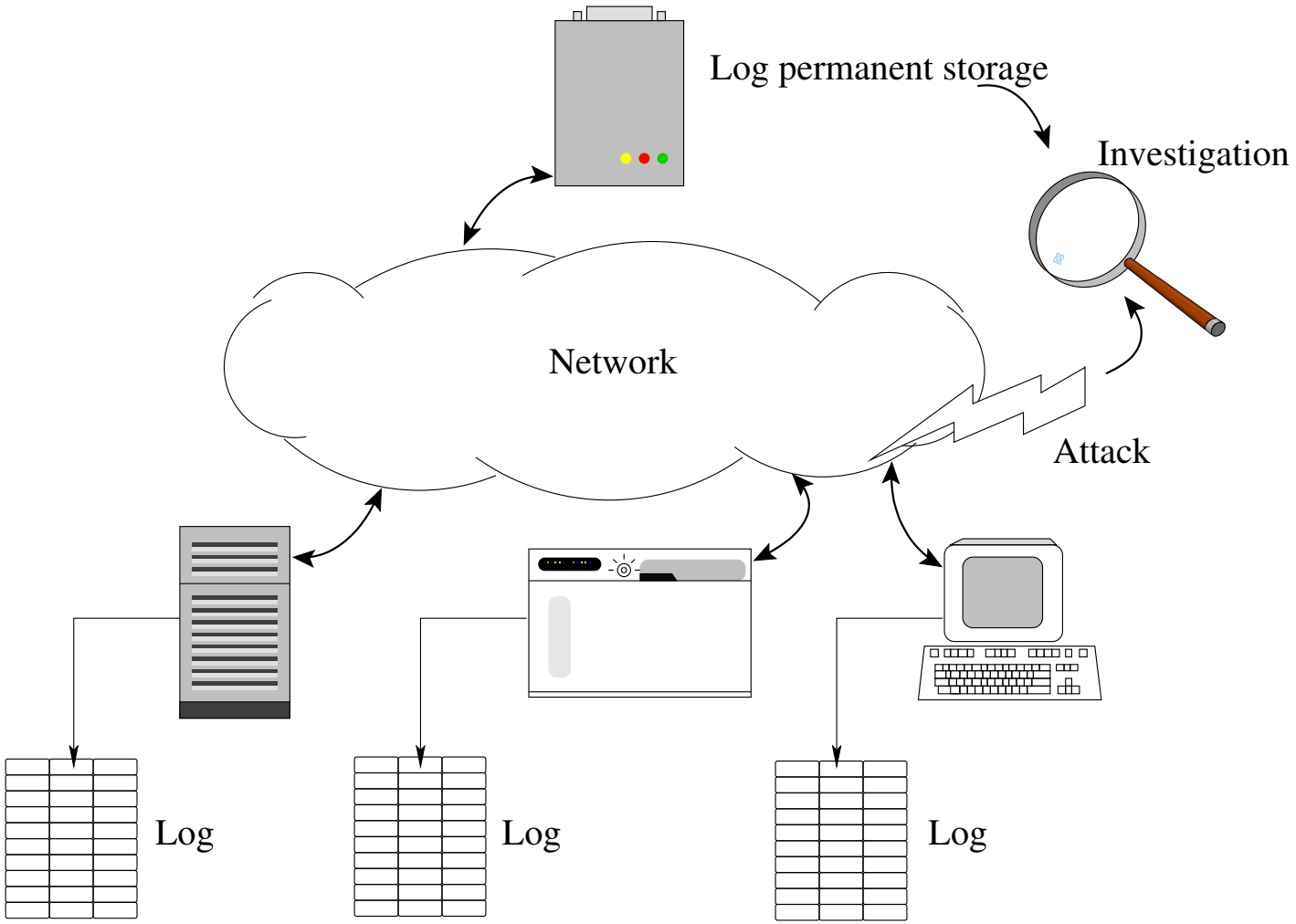
An Abstract View



A Scenario



A Scenario


Log permanent storage
Investigation
Network
Attack
Log
Log
Log

Goal and Challenges

From the previous discussion some of the issues are:

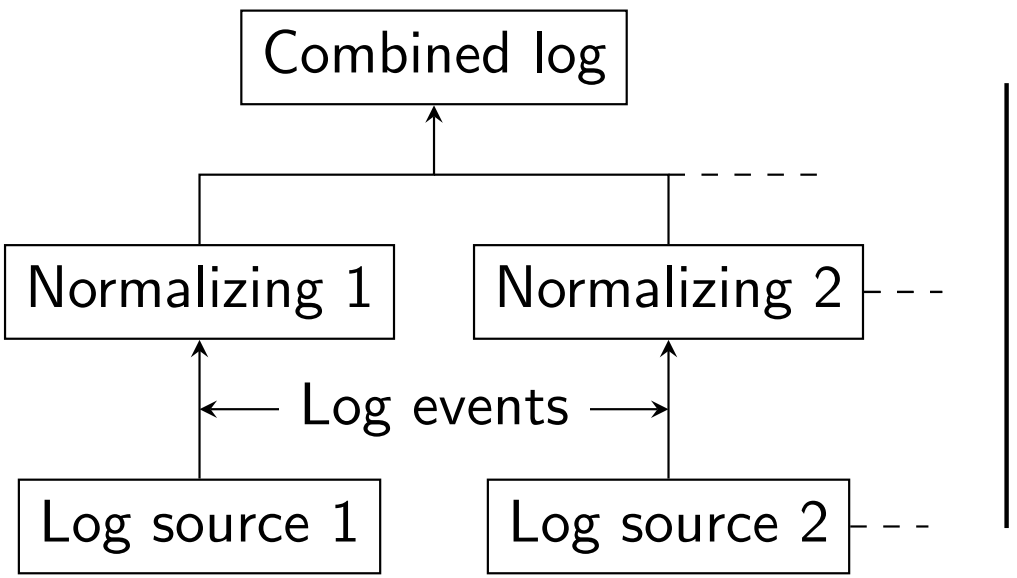
- Synchronization of time stamps between different log sources.
- Integrity of logs.
- Security of logs during transmission.
- Unifying formats from different log sources.
- Correlating logs from different sources for forensic purposes.

Goal and Challenges

From the previous discussion some of the issues are:

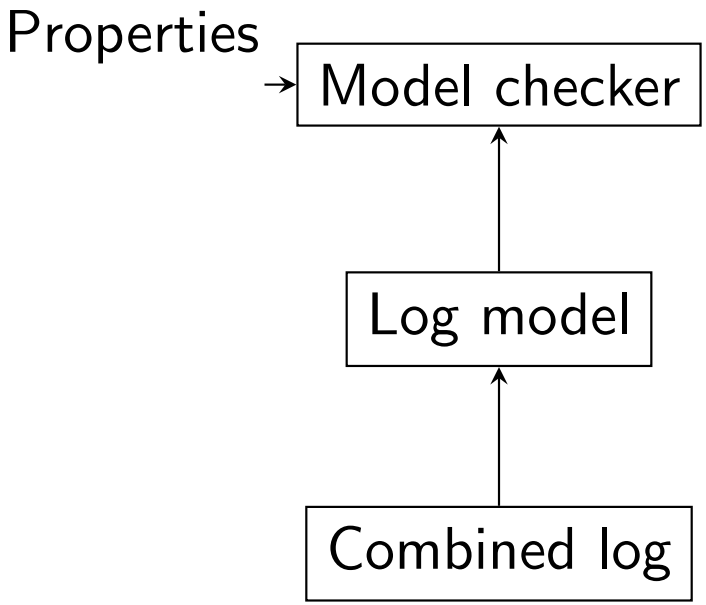
- Synchronization of time stamps between different log sources.
- Integrity of logs.
- Security of logs during transmission.
- Unifying formats from different log sources.
- Correlating logs from different sources for forensic purposes.

Combining Logs



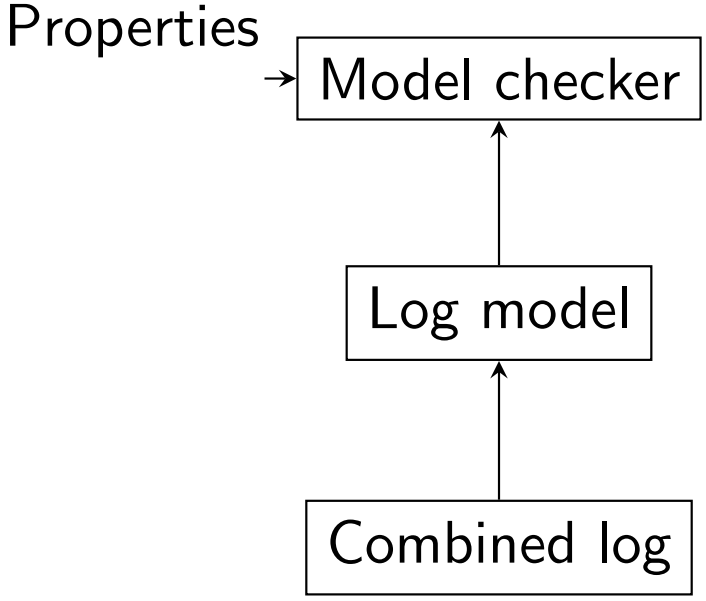
- Log events have heterogenous formats
- Normalizing produces events with a common format.
- Normalized events are combined and stored.
- Stored events are time-stamped and their sources are known.

Model Checking Approach



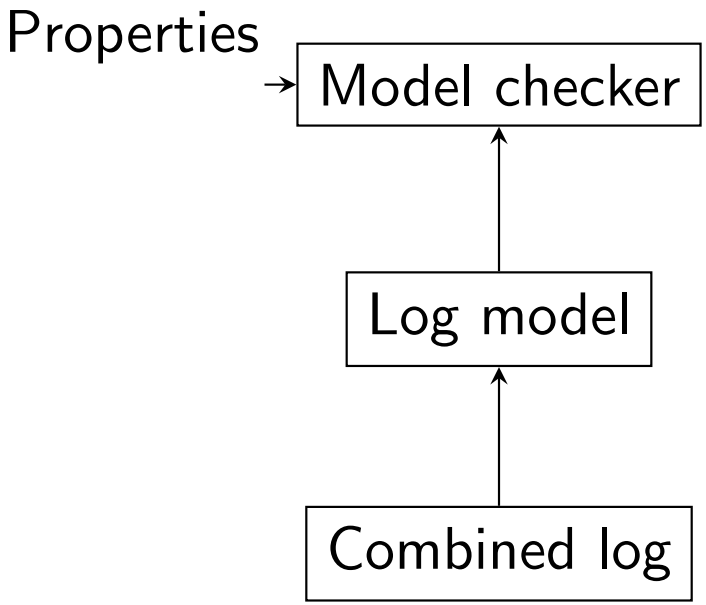
- Our objective is to analyze and correlate log events.
- Model checking is a good formal solution.
- A suitable model for logs is developed.
- Properties are expressed in a suitable logic.
- But .. What model? and What logic?

Model Checking Approach



- Our objective is to analyze and correlate log events.
- Model checking is a good formal solution.
  - We are searching for patterns of events.
  - Patterns are often expressed in formal languages.
  - The search then amounts to model checking.
- A suitable model for logs is developed.
- Properties are expressed in a suitable logic.
- But .. What model? and What logic?

Model Checking Approach



- Our objective is to analyze and correlate log events.
- Model checking is a good formal solution.
- A suitable model for logs is developed.
- Properties are expressed in a suitable logic.
- But .. What model? and What logic?

Multi-sorted Term Algebras

Digression: Multi-sorted Algebra

$\Rightarrow$ **Starting from:**

- A set of sorts $\{s_1, s_n, \ldots\}$.
- For each sort s':
  - A set of variables of the sort.
  - A set of function symbols $f : s_1 \times s_2 \ldots \times s_n \rightarrow s'$ of arity n, where $n \geq 0$.
  - Function symbols of arity 0 are called constants of sort s'.

$\Rightarrow$ **The set of terms of sort s' contains:**

- All variables of sort s'.
- All terms $f(t_1, \ldots t_n)$ where f is a function symbol $f : s_1 \times s_2 \ldots \times s_n \rightarrow s'$ and each t_i is a term of sort s_i.

The multi-sorted term algebra is the set of terms of all sorts.

An example of an algebra

- Sorts: $\{int, bool\}$
- variables of sort int: $x_i, y_i, \ldots$ and of $bool$: $x_b, y_b, \ldots$
- Function symbols of sort int: $\{add, 0, 1\}$, where $add : int \times int \longrightarrow int$ and 0 and 1 are constants.
- Function symbols of sort $bool$: $compare : int \times int \longrightarrow bool$

All the following are terms of the term algebra:

- $add(x_i, y_i)$, $add(0, 1)$, $add(z_i, 1)$
- $compare(x_i, y_i)$, $compare(add(0, 1), 1)$, $compare(z_i, 1)$

Event and Log Models

- Each normalized log event is modeled as a term of a term algebra.
- The function symbols represent action, and the parameters provide information about this action.
- The combined log itself is modeled as a tree.
- Edges of the tree are the algebraic terms representing log events.
- Starting from the root, events appear from oldest to the most recent.
- Concurrent events appear as two branches at the same level.

Model for Log Events

A log event is modeled as a term of a term algebra with variables.

$ProcessExit : \text{process} \times \text{user} \rightarrow \text{bool}$	Function symbol "*ProcessExit*", sorts of domain and codomain are shown.
$Logon : \text{user} \times \text{userID} \times \text{group} \times \text{domain} \times \text{process} \times \text{logonType} \rightarrow \text{bool}$	Function symbol (Operation) "*Logon*".

Example:

- The term $ProcessExit(p_1, u_x)$ means user u_x terminated process p_1.

Examples of Events

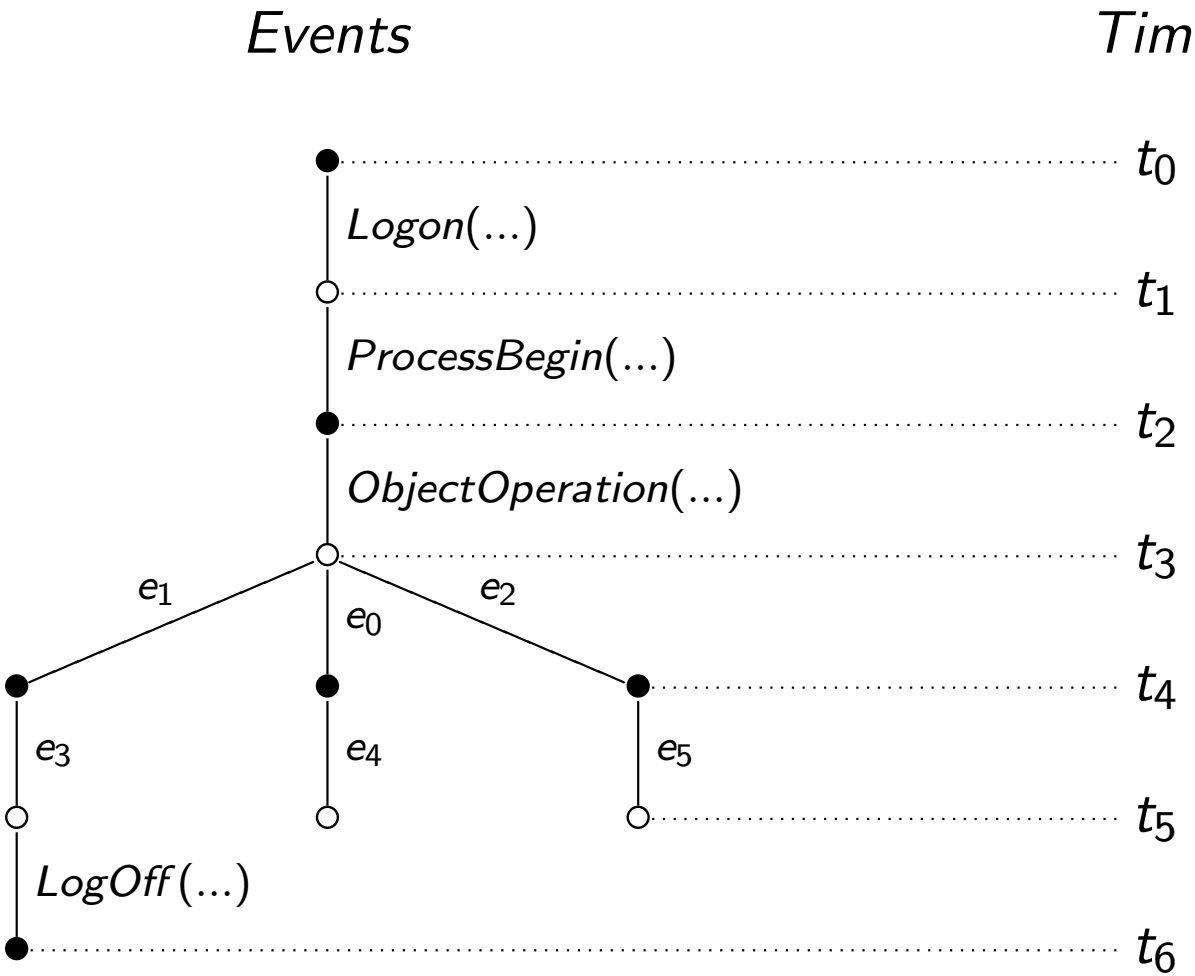
We can think of a large a number of events, we can mention:

- User-related events:
  Logon, *Logoff* , etc.
- Process-related events:
  ProcessBegin, *ProcessExit*, etc.
- Network-related events:
  Get, *Post*, etc.

Each of these can be represented by an appropriately designed function symbol from the algebra.
The parameters of each function symbol should convey required information about the event.
How about logs of events?

Model for Logs

A log is modeled as a tree whose edges are terms of a term algebra.



- e_0 to e_2 are concurrent.
- All terms contain no variables (are ground terms).

- How to describe properties of the tree?

Expressing Properties of Log Trees

Overview

Given a log tree whose edges are ground terms of the term algebra, we would like to search it for suspicious patterns of events.

The questions now are:

- How to express these patterns?
- How to express occurrences of patterns?
- How to search the tree?

Examples of Patterns

Some of the patterns we may be interested to look for:

- Signatures of known attacks.
- For every "*Logon*" of some user u_x there should be a "*Logoff*" by the same user.
- Suspicious acts, such the deactivation of the antivirus followed by a program installation.
- Checking the privileges given to a process against the privileges of the user who started it and making sure the security policy is not violated.

Syntax of Patterns

A pattern r is defined by the grammar:

$r ::=$

- ϵ | The empty sequence.
- $a.r$ | "a" followed by a pattern, "a" defined below.

  $a ::=$
  - t | term of the term algebra with variables.
  - $\lceil t \rceil$ term of the term algebra containing t as a subterm.
- $x_r.r$ x_r represents any sequence of terms.

- "a" represents single events in the log tree model
- "a" is either a term "t" of the algebra or a term containing "t" ($\lceil t \rceil$), it may contain variables.
- "x_r" represents a sequence of terms of zero or any finite length.

Examples of Patterns

The following are examples of patterns:

- $x_r.Logon(u_x, p_2, p_3, p_4, p_5, p_6).y_r.Logoff(u_x)$
  A sequence of events followed by a "*Logon*" of a user followed by another sequence then a "*Logoff*" by the same user.
  We note here:
  - We must distinguish between sequence variables $(x_r, y_r, \ldots)$ and term variables $(u_x, p_2, \ldots)$.
  - We must distinguish between constants and variables and between sorts, p_2 to p_6 above are variables.
  - If u_x is a variable then the pattern means a "*Logon*" by a user followed by a "*Logoff*" of the **same** user.
  - If u_x is a constant then the pattern means a "*Logon*" by a specific userfollowed by a "*Logoff*" of the **same** user.

Examples of patterns (2)

- $x_r.\lceil u_x \rceil.y_r.Logoff(u_x)$
  Assume u_x is of sort user , then the pattern means:
  A sequence of events followed by any event involving the user u_x followed by another sequence then a "*Logoff*" by the same user u_x.
- In the pattern above, x_r, y_r are sequence variables and u_x is a term variable.

Given a pattern like the above how do we match it to an actual sequence of the log tree?

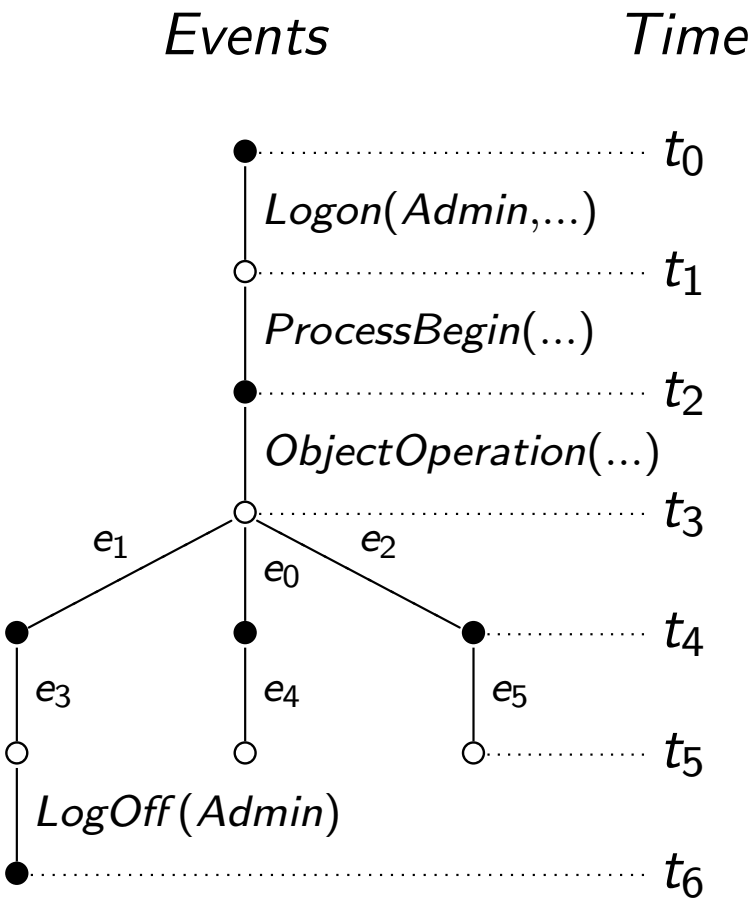
$\Longrightarrow$ Substituting sequence variables and term variables.

Examples of patterns (2)

- $x_r.\lceil u_x \rceil.y_r.Logoff(u_x)$
  Assume u_x is of sort user , then the pattern means:
  A sequence of events followed by any event involving the user u_x followed by another sequence then a "*Logoff*" by the same user u_x.
- In the pattern above, x_r, y_r are sequence variables and u_x is a term variable.

Given a pattern like the above how do we match it to an actual sequence of the log tree?
$\Rightarrow$ Substituting sequence variables and term variables.

Searching for patterns



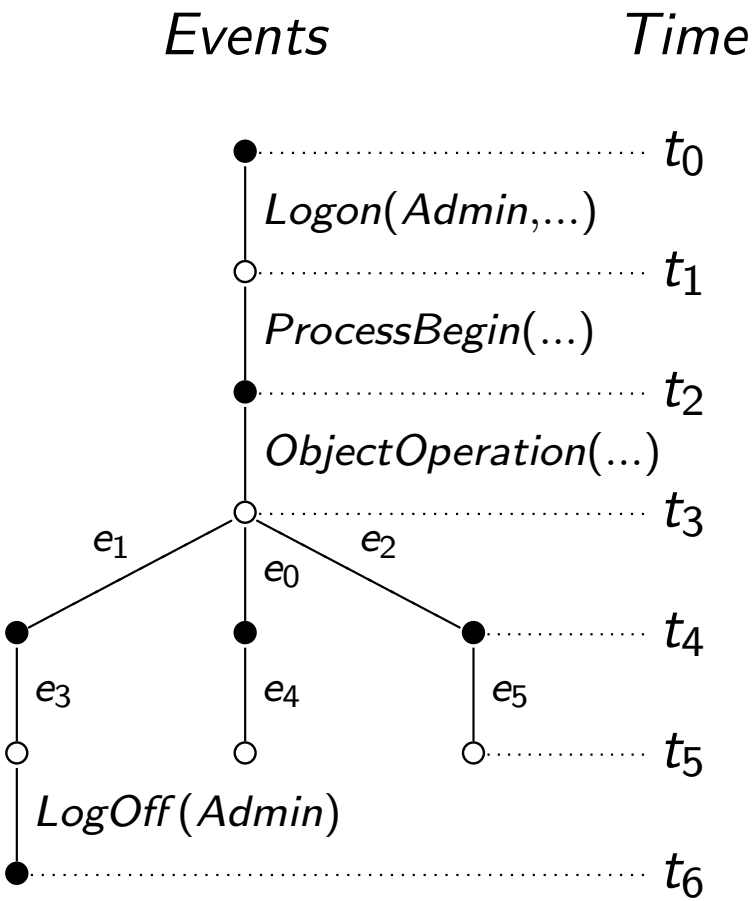
To look for pattern:
$x_r.Logon(u_x, \ldots).y_r.Logoff(u_x)$

We have to find two substitutions:
(1) for sequence variables x_r, y_r and
(2) for term variables u_x, such that:

Applying the substitutions to the pattern will result in a sequence in the tree.

In the example: $u_x \mapsto Admin$, $x_r \mapsto \epsilon$, and $y_r \mapsto ProcessBegin(\ldots).ObjectOperation(\ldots).e_1.e_3$ and so we can find one sequence that matches the pattern.

Searching for patterns



To look for pattern:
$x_r.Logon(u_x, \ldots).y_r.Logoff(u_x)$

We have to find two substitutions:
(1) for sequence variables x_r, y_r and
(2) for term variables u_x, such that:

Applying the substitutions to the pattern will result in a sequence in the tree.

In the example: $u_x \mapsto Admin$, $x_r \mapsto \epsilon$, and $y_r \mapsto ProcessBegin(\ldots).ObjectOperation(\ldots).e_1.e_3$ and so we can find one sequence that matches the pattern.

Need for Logic

- We need to be able to express more complicated properties of sequences, i.e., for every "*Login*" there is a "*Logoff*", for this we use logic formulas.
- A formula is true for a certain model (log tree) L if there are sequences "s" in L that satisfy the property expressed by the formula.
- The semantics of a formula is taken to be the set of sequences of the log tree L for which the formula is true.

Syntax of Logic

$$\varphi ::=$$

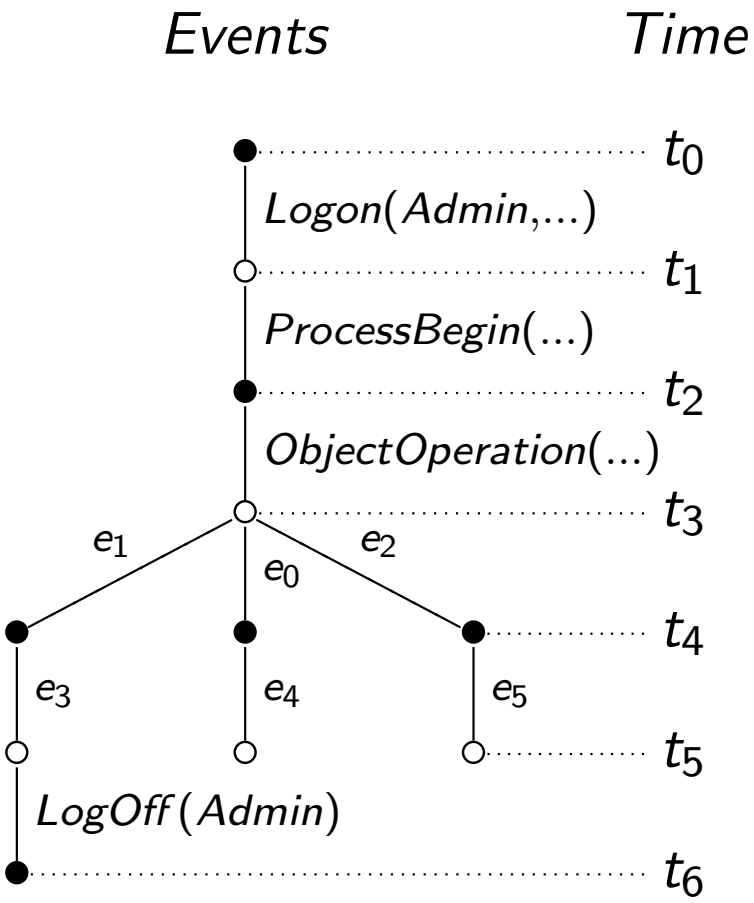
$Z \mid$	Formula variable.
$\neg\varphi \mid$	Negation.
$\varphi_1 \wedge \varphi_2 \mid$	Conjunction.
$[r_1 \rightarrowtail r_2]\varphi \mid$	All sequences in the log tree that match the pattern r_1 and when modified to match r_2 will satisfy φ.
$\langle\langle \mathcal{L} \rangle\rangle \varphi \mid$	Limits the search to events whose log source $l \in \mathcal{L}$
$\nu Z.\varphi$	The largest set of sequences that is a fixpoint of a semantic function mapping Z to a set of sequences.

More Formulas

We define the following:

$\neg(\neg\varphi_1 \wedge \neg\varphi_2) \equiv \varphi_1 \vee \varphi_2$	Disjunction.
$\neg\varphi_1 \vee \varphi_2 \equiv \varphi_1 \Rightarrow \varphi_2$	Implication.
$\neg[r_1 \looparrowright r_2]\neg\varphi \equiv \langle r_1 \looparrowright r_2\rangle\varphi$	There exists sequences in the log tree that match the pattern r_1 and when modified to match r_2 will satisfy φ.
$\neg\nu Z.\neg\varphi[\neg Z/Z] \equiv \mu Z.\varphi$	Least fix point, i.e., the largest set of sequences that is a fixpoint of a semantic function mapping Z to a set of sequences.
$\nu Z.Z \equiv \mathtt{tt}$	True.
$\mu Z.Z \equiv \mathtt{ff}$	False.

Examples of Formulas



Formulas satisfied by the tree:

- $\nu Z.[x_r.Logon().y_r.Logoff().z_r \not\mapsto x_r.y_r.z_r]\rangle Z$

- $\langle x_r.e_1.y_r \not\mapsto \epsilon\rangle \mathtt{tt}$

- $\langle x_r.e_3.y_r \not\mapsto x_r\rangle\langle x_r.e_3.y_r \not\mapsto \epsilon\rangle \mathtt{tt}$

Tableau-Based Proof System

- A tableau is a proof tree whose root is the formula to be proved.
- Rules of the tableau indicate how the tree grows downward.
- At some point we reach leaves of the tree.
- Rules indicate when a leaf is considered successful.
- A formula is proved if it has a successful tableau, i.e., all the leaves are successful.

Conclusion

- Model checking is proposed for log analysis.
- Logs are modeled as trees whose edges are algebraic terms.
- Properties of logs are expressed in a logic that deals with sequences of terms.
- A proof system is given to prove properties.