Labeled Transition Systems

CS4605
George W. Dinolt

This work is taken from Dave Bibighause PhD thesis: APPLYING DOUBLY LABELED TRANSITION SYSTEMS TO THE REFINEMENT PARADOX which you can find at http://bosun.nps.edu/uhtbin/hyperion.exe/05Sep%5FBibighaus%5FPDF
Elements

$\mathcal{A}$: Set of actions

$\mathcal{A}^*$: The set of all sequences of elements of $\mathcal{A}$. The elements of $\mathcal{A}^*$ are sometimes called traces. They may be written:

$$t \in \mathcal{A}^*: \ t = \langle a_1, a_2, \ldots, a_n \rangle$$

SYS: $\text{SYS} \subseteq \mathcal{A}^*$.

SL: $\text{SL} = \{\text{high}, \text{low}\}$

$sl: \ sl: \mathcal{A} \to \text{SL}$ maps the actions to security labels
Refinement

If

$$SYS_1, SYS_2 \subseteq A^*$$

then $$SYS_2$$ is a refinement of $$SYS_1$$ if and only if

$$SYS_2 \subseteq SYS_1.$$

The idea is that an abstraction describes many more possible traces than an implementation.
Safety and Liveness

- A trace satisfies a safety property if that property is true for every element of the trace.
- A trace satisfies a liveness property if that property is true for one element of the trace.
- Safety and Liveness are properties of a single trace.
- A system $SYS$ satisfies a safety or liveness property if every one of its traces does.
- Bell & LaPadula Security is a safety property.
Security

Let \( \text{purge} : \mathcal{A}^* \to \mathcal{A}^* \) such that

\[
\forall t = \langle a_1, a_2, \ldots, a_n \rangle \in \mathcal{A}^*
\]

\[
\text{purge}(t) = \begin{cases} 
\langle \rangle & \text{if } t = \langle \rangle \\
\text{purge}(\langle a_2, a_3, \ldots, a_n \rangle) & \text{if } \text{sl}(a_1) = \text{high} \\
\langle a_1, \text{purge}(\langle a_2, a_3 \ldots, a_n \rangle) \rangle & \text{if } \text{sl}(a_1) = \text{low}
\end{cases}
\]

Then \( \mathcal{S} \mathcal{Y} \mathcal{S} \subseteq \mathcal{A}^* \) is **secure** if and only if:

\[
\forall t \in \mathcal{S} \mathcal{Y} \mathcal{S} \Rightarrow \text{purge}(t) \in \mathcal{S} \mathcal{Y} \mathcal{S}
\]
“Refinement Paradox”

- Note that this version of security is a property of a set of traces, not a single trace.

- Suppose $SYS_1$ satisfies the security property defined above and $SYS_2 \subseteq SYS_1$ ($SYS_2$ is a refinement of $SYS_1$). Then it may not be the case that $SYS_2$ satisfies the security property.

- Note this is the “opposite” of the “Bell & LaPadula” style models.
Labeled Transition Systems

\( \mathcal{A} \): Set of actions

\( \Sigma \): Set of States (disjoint from actions)

\( s_0 \): Then element \( s_0 \in \Sigma \) is the \textit{Start State} of the system.

\( \mathcal{E} \): Set of events, \( \mathcal{E} = \Sigma \times \mathcal{A} \times \Sigma \). For \( e \in \mathcal{E} \) we may use the notations:

\[
e = (s_1, a, s_2) = s_1 \xrightarrow{a} s_2
\]

\( \mathcal{LTS} \): Any set \( E \subseteq \mathcal{E} \) defines a \textit{Labeled Transition System}. We usually write:

\[
\mathcal{LTS} = \{ \Sigma, \mathcal{A}, E, s_0 \}
\]

to describe the system.
Example LTS's

\[ \begin{align*}
\text{LTS}_1 \\
& \quad s_1 \\
& \quad \quad \downarrow a_1 \\
& \quad s_2 \\
& \quad \quad \downarrow a_2 \\
& \quad s_3 \\
& \quad \quad \downarrow a_3 \\
& \quad s_4 \\
\end{align*} \]

\[ \begin{align*}
\text{LTS}_2 \\
& \quad s_1 \\
& \quad \quad \downarrow a_1 \\
& \quad \quad \quad \downarrow a_1 \\
& \quad s_2 \\
& \quad \quad \downarrow a_2 \\
& \quad \quad \quad \downarrow a_2 \\
& \quad s_3 \\
& \quad \quad \downarrow a_3 \\
& \quad \quad \quad \downarrow a_3 \\
& \quad s_4 \\
& \quad \quad \downarrow a_2 \\
& \quad \quad \quad \downarrow a_2 \\
& \quad s_5 \\
& \quad \quad \downarrow a_2 \\
& \quad \quad \quad \downarrow a_2 \\
& \quad s_6 \\
& \quad \quad \downarrow a_3 \\
& \quad \quad \quad \downarrow a_3 \\
& \quad s_7 \\
\end{align*} \]
SubTraces in LTS’s

Suppose $\Sigma$ is a set of states and $\mathcal{A}$ is a set of actions. We can define $\mathcal{LTS}$ to be the set of all Labeled Transition Systems over the sets $\Sigma$ and $\mathcal{A}$. Each element $LTS \in \mathcal{LTS}$ is denoted by $LTS = \{\Sigma, \mathcal{A}, E, s_0\}$. We define the function

\[
\text{subtrace?} : \mathcal{A}^* \times \mathcal{LTS} \times \Sigma \to \text{bool}
\]

as follows: for $t = \langle a_1, a_2, \ldots, a_n \rangle$ and $t' = \langle a_2, a_3, \ldots, a_n \rangle$

\[
\text{subtrace?}(t, LTS, s) = \begin{cases} 
  \text{true} & \text{if } t = \langle \rangle \\
  \text{true} & \text{if } \exists s \xrightarrow{a_1} s' \in E \land \text{subtrace?}(t', LTS, s') \\
  \text{false} & \text{otherwise}
\end{cases}
\]
Traces in an LTS

The element $t \in \mathcal{A}^*$ is a trace of $LTS = \{\Sigma, \mathcal{A}, E, s_0\}$ if and only if $\text{subtrace?(}t, LTS, s_0) = \text{true}$.

$$
\text{Traces}(LTS) = \{t \in \mathcal{A}^* : \text{subtrace?(}t, LTS, s_0) = \text{true}\}
$$
Example Traces - Single State

\[
LT S_1 = \{ \{s_1\}, \{a_3\}, \{s_1 \xrightarrow{a_3} s_1\}, s_1 \}
\]

Potential traces:

\[
<> , \ <a_3> , \ <a_3,a_3> , \ <a_3,a_3,a_3>
\]
Example Traces - Two States

\[ LTS_2 = \{ \{s_1, s_2\}, \{a_1, a_2\}, \{s_1 \xrightarrow{a_1} s_2, s_2 \xrightarrow{a_2} s_1\}, s_1\} \]

Potential traces:

\(<\>, <a_1>, <a_1, a_2>, <a_1, a_2, a_1>\)
Example Traces - Three Actions

\[ LTS_3 = \{\{s_1, s_2\}, \{a_1, a_2, a_3\}, \{s_1 \xrightarrow{a_1} s_2, s_2 \xrightarrow{a_2} s_1, s_1 \xrightarrow{a_3} s_1\}, s_1\} \]

Potential traces:

\[
\langle\rangle, \langle a_3 \rangle, \langle a_3, a_3 \rangle, \langle a_3, a_3, a_3 \rangle, \\
\langle a_1 \rangle, \langle a_1, a_2 \rangle, \langle a_1, a_2, a_1 \rangle, \\
\langle a_3, a_1 \rangle, \langle a_3, a_1, a_2 \rangle, \langle a_1, a_2, a_3 \rangle
\]
Trace Refinement of LTS’s

Note that

\[ \text{Traces}(LTS_1) \subset \text{Traces}(LTS_3) \]

and

\[ \text{Traces}(LTS_2) \subset \text{Traces}(LTS_3) \]

So \( LTS_1 \) and \( LTS_2 \) are \textit{trace refinements} of \( LTS_3 \).

I find this definition of “refinement” somewhat incongruous. One might expect that a “refinement” has more detail (more states, more actions?) than the more abstract specification.
Suppose \( \text{LTS}_A = \{\Sigma_A, A, E_A, s^A_0\} \) and \( \text{LTS}_C = \{\Sigma_C, A, E_C, s^C_0\} \) are Labeled Transition Systems. Suppose 
\[ R \subseteq \Sigma_A \times \Sigma_C \]
is Left/Right Total. We say that \( \text{LTS}_C \) simulates \( \text{LTS}_A \) if:

\[ \forall s_c \in \Sigma_c, s_a \in \Sigma_A, s_c \xrightarrow{e} s'_c \in E_C, (s_a, s_c) \in R \text{ implies } \exists s'_a \in \Sigma_A \text{ such that } s_a \xrightarrow{e} s'_a \in E_A \text{ and } (s'_a, s'_c) \in R \]

*Left/Right Total* means that that \( \forall s_a \in \Sigma_A, \exists (s_a, s'_c) \in R \) and \( \forall s_c \in \Sigma_B, \exists (s'_a, s_c) \in R \)

\text{gwdinolt@nps.edu}   \text{September 2, 2008}
Simple LTS Simulation

$LTS_A = \{\{s_1\}, \{a_3\}, \{s_1 \xrightarrow{a} s_1\}, s_1\},$

$LTS_{C1} = \{\{y_1 y_2\}, \{a_3\}, \{y_1 \xrightarrow{a_3} y_1, y_1 \xrightarrow{a_3} y_2\}, y_1\}$

$R = \{(s_1, y_1), (s_1, y_2)\}.$
Another Simulation

\[ LTSC_1 = \{y_1 y_2, \{a_3\}, \{y_1 \xrightarrow{a_3} y_1, y_1 \xrightarrow{a_3} y_2\}, y_1\} \]
\[ LTSC_2 = \{x_1 x_2, x_3, \{a_3\}, \{x_1 \xrightarrow{a_3} x_2, x_2 \xrightarrow{a_3} x_2, x_2 \xrightarrow{a_3} x_3\}, x_1\} \]
\[ R = \{(y_1, x_1), (y_1, x_2), (y_2, x_3)\} \]
A More Complex Simulation

\[
\begin{array}{c}
LTS_1 \\
\begin{array}{c}
s_1 \\
a_1 \downarrow \\
s_2 \\
a_2 \downarrow \\
s_3 \\
a_3 \downarrow \\
s_4 \\
\end{array} \\
\begin{array}{c}
LTS_2 \\
\begin{array}{c}
s_1 \\
a_1 \downarrow \\
s_2 \\
a_2 \downarrow \\
s_3 \\
a_3 \downarrow \\
s_4 \\
\end{array} \\
\begin{array}{c}
s_5 \\
a_2 \downarrow \\
s_6 \\
a_3 \downarrow \\
s_7 \\
\end{array}
\end{array}
\end{array}
\]
Traces and Simulation

If $LTS_C$ simulates $LTS_A$ with relation $R$, we will write this as:

$$LTS_C \mathrel{\triangleleft_R} LTS_A$$

**Theorem 1** If $LTS_C \mathrel{\triangleleft_R} LTS_A$ then $Traces(C) \subseteq Traces(A)$

**Theorem 2** If $LTS_C \mathrel{\triangleleft_R} LTS_A$ and every $t_a \in Traces(LTS_A)$ satisfies some safety property, then so does every $t_c \in Traces(LTS_C)$.

If we define security as a safety property on traces then, as a result of the above, if an abstract specification satisfies the security property, then the concrete system that is a simulation will also satisfy that requirement.
Traces of this LTS are

\[<> , <l>, <l, h>\]

This system is \textit{trace secure} since the \textit{purge} of each of the traces is

\[<> , <l>, <l>\]
The traces of this system are:

\[<>\], \(<h>\), \(<h,l>\]

The purges of the each of the traces is:

\[<>\], \(<>\), \(<l>\]

Note that the \(\text{purge}(<h,l>) \equiv <l>\) is not one of the traces of the system. This system is \textit{not} trace secure.
The traces of this system are

$$<> , < h >, < l >, < h, l >$$

The \textit{purges} are:

$$<> , <> , < l > , < l >$$

and each of the \textit{purges} are traces of the original system. Hence this system is \textit{trace secure}.

gwdinolt@nps.edu

September 2, 2008
Simulation of Trace Secure System is not Trace Secure.
Security and Refinements so Far

- If we specify systems using Traces, define Security using “Purging” and Refinement by subsets of traces, Security is not maintained (in general) by refinement.

- If we specify systems using LTS’s, define security using “Purging” and Refinement by “Simulation”, security is still not preserved under refinement.

- We need to construct a new kind of system specification, a new definition of security and a new definition of refinement.
Doubly Labeled Transition Systems

The elements

$\Sigma$: A set of States

$\mathcal{A}$: The Set of Actions

$E^{May}$: $\subseteq \Sigma \times \mathcal{A} \times \Sigma$, the set of *may* Events. These events may or may not occur in any refinement of the system.

$E^{Must}$: $\subseteq \Sigma \times \mathcal{A} \times \Sigma$, the set of *must* Events. These events must appear in any refinement of the system. Note $E^{Must} \subseteq E^{May}$.

$s_0$: The initial state of the system.

We use the notation $DLTS = \{\Sigma, \mathcal{A}, E^{May}, E^{Must}, s_0\}$ to denote a *doubly labeled transition system*.
Traces in DLTS’s

We can form an \( LTS \) out of the DLTS \( X = \{ \Sigma, A, E^{May}, E^{Must}, s_0 \} \) by ignoring the “must” events, since \( E^{Must} \subseteq E^{May} \).

\[
X^{May} = \{ \Sigma, A, E^{May}, s_0 \}
\]

We can form another \( LTS \) out of the DLTS by using the “must” events:

\[
X^{Must} = \{ \Sigma, A, E^{Must}, s_0 \}
\]
Refinement of $DLTS$'s

Suppose

$DLTS_A = \{ \Sigma_A, A, \epsilon_A^{May}, \epsilon_A^{Must}, s_0^A \} = A$

$DLTS_C = \{ \Sigma_C, A, \epsilon_C^{May}, \epsilon_C^{Must}, s_0^C \} = C$

$R \subseteq \Sigma_A \times \Sigma_C$ is Left/Right Total

$C^{May} \triangleleft_R A^{May}$

$A^{Must} \triangleleft_{R^{-1}} C^{Must}$

One says that “May” events of $C$ simulates the “May” events of $A$ and the “Must” events of $A$ simulate the “Must” events of $C$. In this case $C$ is a $DTLS$-refinement of $A$ and if $A$ and $C$ are $DLTS$’s we will abuse the notation and write

$C \triangleleft_R A$

for this as well.
Interpretation of $DLTS$ refinement

Since $C^{May} \triangleleft_R A^{May}$ we have every trace in $C$ is a trace in $A$. So any safety property satisfied by all the traces in $A$ will be satisfied by all the traces in $C$.

Since $A^{Must} \triangleleft_R^{-1} C^{Must}$, for each $s_a \xrightarrow{e} s'_a \in E_A^{Must}$ and $(s_c, s_a) \in R^{-1}$ there is a $s'_c$ such that $s_c \xrightarrow{e} s'_c \in E_C^{Must}$ and $(s'_c, s'_a) \in R^{-1}$. Every “Must” event in $A$ is guaranteed to occur in $C$. (Liveness property)
Properties of $DLTS$ refinement

If $A$, $C$ and $D$ are $DLTS$'s with the same set of actions, $A$, and $I$ is the identity relation and $R_1$ is a relation on $\Sigma_A \times \Sigma_C$ and $R_2$ is a relation on $\Sigma_C \times \Sigma_D$, then

$$A \triangleleft_I A$$

$$C \triangleleft_{R_1} A \text{ and } D \triangleleft_{R_2} C \Rightarrow D \triangleleft_{R_1(R_2)} A$$

As a result we have

$$traces(D^{May}) \subseteq traces(C^{May}) \subseteq traces(A^{May})$$

which implies the systems are ordered in the same way when viewed as “just” $\mathcal{LTS}$s.
Restricted Bi-Similarity

Suppose \( S = \{ \Sigma_S, A, \mathcal{E}_S^{\text{May}}, \mathcal{E}_S^{\text{Must}}, s_0 \} \) is a DLTS. Suppose \( A \subseteq A \). We can form the DLTS

\[
S|_A = \{ \Sigma, A, \mathcal{E}^{\text{May}}|_A, \mathcal{E}^{\text{Must}}|_A, s_0 \}
\]

where the | symbol is read as “restricted to” and \( \mathcal{E}|_A = \{ s \xrightarrow{e} s' \in \mathcal{E} : e \in A \} \) (for both “may” and “must” events).

Suppose \( R \subseteq \Sigma_S \times \Sigma_S \) is reflexive and commutative\(^\dagger\). Suppose also that \( S|_A \triangleleft_R S \) (as a DLTS).

We call \( \triangleleft_R^A \) a **Restricted Bi-Similarity** of \( S \) over the set of actions \( A \).

\(^\dagger\)“Reflexive” means that \( \forall s \in \Sigma_S, (s, s) \in R \) and commutative means that if \( (s_1, s_2) \in R \) then \( (s_2, s_1) \in R \).
Bi-Simulation Security

Suppose $S$ is a DLT $S$ and we have partitioned the actions $\mathcal{A}$ into $High$ and $Low$ actions. Suppose there exists a relation $R$ such that $\triangleright_{R}^{Low}$ is a restricted Bi-Simulation over $Low$.

We say that $S$ is **Bi-Simulation Secure** if

$$\forall s_1, s_2 \in \Sigma, e \in High \text{ and } s_1 \xrightarrow{e} s_2 \in \mathcal{E}^{May} \Rightarrow (s_1, s_2) \in R$$

If $S$ is “Bi-Simulation secure” then it is non-interfering.
The DLT S is \textit{Low View Complete} if and only if

\[ \forall e \in \text{Low}, s_1 \xrightarrow{e} s_2 \in \mathcal{E}^{May} \Rightarrow s_1 \xrightarrow{e} s_2 \in \mathcal{E}^{Must} \]

i.e. every low event is a “must” event.
Secure Refinement of Traces

Let $A$ and $C$ be $DLTS$'s and suppose that

- $A$ is Bi-Simulation Secure over the $Low$ actions in $A$
- $A$ is Low-View Complete
- $R$ is a left-right total relation in $\Sigma_A \times \Sigma_c$
- $C \prec_R A$ ($C$ is a $DLTS$ refinement of $A$)

Then $C$ is both Bi-Simulation Secure over $Low$ and is Low-View Complete