利用安全圖對重視安全之系統進行模型驗證

陳盈如、林彥宏、熊博安

國立中正大學資訊工程學系 嵌入式系統實驗室

一、簡介

隨著科技日新月異，在我們日常生活中，隨處可見的交通運輸系統、電子商業交易系統，以及醫療系統...等等，都與安全問題息息相關。然而在設計與安全問題相關的系統中，若存在著設計者無法預知的潛在錯誤，將可能導致系統發生危及生命財產的重大災害。為了確定這些對安全甚為重視的系統(safety-critical system)能夠做到真正的安全，所以需要對這些系統做正規驗證(formal verification)。

正規驗證，是一種以數學的方式，來證明系統是否滿足(satisfy)使用者所需要的特性(properties)。為了驗證這些對安全甚為重視的系統，我們採用了目前很普遍的一種正規驗證的方法，稱為「模型驗證」(Model Checking)。在對系統進行模型驗證之前，必須要先將系統正規模型化(formal model)。

然而，能夠用來將系統模型化的方式有很多，例如以「有限狀態機」(finite state machine, FSM)、狀態圖(Statecharts)、以及處理控制事件圖(Process Control Event Diagrams)…等等。我們選擇了以「安全圖」(Safecharts)的方式，將這些與安全問題甚為相關的系統模型化，主要的原因是，安全圖是由狀態圖衍生而來，同時具有能夠描述安全要求的特性。

在接下來的文章中，我們將分成幾個部份逐一說明：在第二節，我們說明何謂系統模型以及模型驗證；第三節中，我們將對安全圖做模型驗證；在第四節，將描述如何將理論實作在我們所發展的驗證工具－狀態圖形操作器(State-Graph Manipulator, SGM)中；第四節中，我們將舉一個實際應用的例子；然後在最後一節做結論。

二、系統模型以及模型驗證

在進一步討論用安全圖進行模型驗證之前，我們需要先了解以下基本而且正式的定義：

1. 狀態圖(Statecharts): 以$F = (S, T, E, \theta, V, \phi)$來描述一個狀態圖，其中$S$表示所有狀態所成的集合，$T$表示所有可能的狀態轉換(transitions)所成的集合，E表示所有事件(events)所成的集合，θ表示在狀態圖中的狀態可能的型態(type)，也就是說θ={AND,OR, BASIC}，V表示由整數變數所成的集合，而ϕ ::= v ~ c | $\phi_1 \wedge \phi_2$ | $\neg\phi_1$，其中$v \in V$，$\sim \in \{<, \leq, =, \geq, >\}$，$c$為一個整數，而$\phi_1$和$\phi_2$為命題(predicates)。當t表示一個狀態轉換，$l(t) = (e, fcond, a)$表示為這個狀態轉換的標籤(label)，一般會將此標籤改寫為$e[fcond]/a$這樣的表示法；其中，e，$fcond$及a，分別表示驅動狀態轉換發生的事件(trigger event)，監守的條件(guarding condition)，以及因為狀態轉換後所發生的行為(generated action)所成的集合。

2. 安全圖(Safecharts)**:** 安全圖是由狀態圖再加上一層對安全要求(safety requirements)的描述，幫助使用者能夠在模型化系統時，將系統主要功能以及安全要求，能分別將它們描述在功能層(functional layer)和安全層(safety layer)。在安全層的描述中，有二個符號：↰及↱，這二個符號能輔助使用者描述安全要求的特性。它們分別代表的意義為：

●↰：表示禁止(prohibition)之意。如果有一個狀態轉換的標籤表示為↰[G]，則它的意義為：只要命題G存在，則禁止執行此狀態轉換。

●↱：表示命令(mandatory)之意。如果有一個狀

態轉換的標籤表示為$[l, u)$ ↱$[G]$，則它的意義為：即使在缺乏驅動狀態轉換發生的事件的情況下，只要命題G存在，則必須在$[l, u)$的時間內執行此狀態轉換。

3. 廣泛的真時自動機(Extended Timed Automaton, ETA): 一般以$A_i = (M_i, m_{i0}, C_i, D_i, L_i, \chi_i, T_i, \lambda_i, \pi_i, \mu_i, \tau_i, \rho_i)$來描述一個廣泛的真時自動機。$M_i$表示模式(modes)所成的有限集合；$m_{i0} \in M$，表示為初始模式；$C_i$是時間變數(clock variables)所成的集合；D_i是一些一般性的不連續變數(discrete variables)所成的集合；L_i是同步標籤(synchronization labels)所成的集合；χ_i表示在一個模式中要保持的條件的不變性；λ_i是將一條狀態轉換和一個同步標籤結合在一起表示；π_i是將一條狀態轉換和一個用來表示優先權(priority)的整數數字，結合在一起表示；μ_i則是用來表示狀態轉換的型態，它可能是{lazy, eager, delayable}其中之一；τ_i定義了狀態轉換的驅動條件；ρ_i則表示了在一個狀態轉換時所必須要做的一些設定，例如對時間變數的設定，或是設定一般變數的值等等。

4. 真時計算樹邏輯(Timed Computation Tree Logic, TCTL)：

一個真時計算樹邏輯的公式所表示的語法為：

$\phi ::= \eta \mid EG\ \phi' \mid E\ \phi' U_{\sim c}\ \phi'' \mid \neg\phi' \mid \phi' \vee \phi''$

其中η為模式的命題，ϕ'及ϕ''為 TCTL 的方程式，$\sim$表示$\{<, \leq, =, \geq, >\}$其中之一，c 為正整數。$EG\ \phi'$的意義為「有一個計算由目前的狀態開始，在持續下去的狀態轉換過程中，ϕ'都永遠是真實的。」。$E\ \phi' U_{\sim c} \phi''$的意義為「存在一個計算，使得從目前的狀態開始，直到ϕ''成立之前，ϕ'都是成立的。」。還有像 EF, AF, AG, AU 等等，都是可以用來幫助使用者描述系統所要滿足的特性。這些 TCTL 的語法，都可以在 Symbolic model checking for real-time systems (由 T.A. Henzinger, X. Nicollin, J. Sifakis, 及 S. Yovine 所著，1992 年發表於 *Proceedings of the IEEE International Conference on Logics in Computer Science*)一文中找到正式的定義。

三、用安全圖做模型驗證

由於用安全圖描述系統，可能因為系統的設計，安全圖的描述會具有階層性質(hierarchical)。但是目前大部份的模型驗證工具無法接受具有階層性質描述的模型輸入，所以我們需要先將模型轉換成為單層描述的輸入。我們所利用以下的技術，依不同型態的狀態，將AND, OR或BASIC型態的狀態，其功能層和安全層，均轉換成在語義上等價的廣泛真時自動機。BASIC型態的狀態可以直接轉換成廣泛真時自動機。然而AND或OR的狀態都是由二個以上的子狀態所組成的。AND狀態，表示其初始的子狀態是同時發生的，而OR狀態則只有一個初始的子狀態。根據這二種不同的意義，分別轉換出相對的廣泛真時自動機。

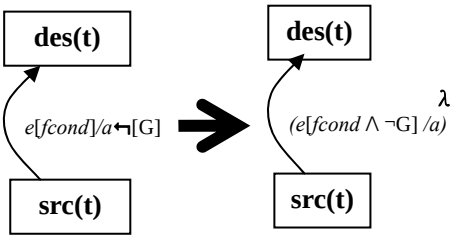
除了不同型態的狀態要做不同的轉換，更進一步要做到如何將安全的語義引入模型驗證中。最重要的就是如何轉換↰及↱這二個具有安全性質描述的符號。在詳細說明如何轉換這二個符號之前，我們需要先仔細介紹三種不同型態的狀態轉換：

- 迫切型(eager evaluation)：這表示狀態轉換要越快越好。我們用ε來表示。在狀態轉換的監守條件成立時，我們假定時間不向前演進。
- 可延遲型(delayable evaluation)：可以將狀態轉換延遲直到最後期限來臨之前。我們用δ來表示。同時我們也假定，在監守條件不成立之前，時間不向前演進。
- 懶惰型(lazy evaluation)：狀態轉換發生與否都可以。我們用λ來表示。

利用這三種不同的型態，接下來介紹如何轉換↰及↱這二個符號。用來表示安全性質

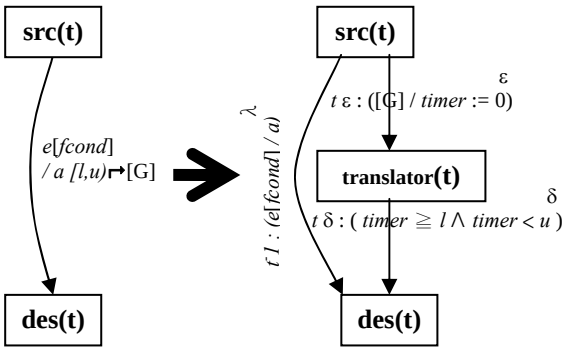
的狀態轉換的描述，可以分為以下三類：

- $e[fcond]/a$：這種是不具有安全性質的描述；這種狀態轉換，我們認為它是屬於懶惰型的。
- $e[fcond]/a$ $\lnot[G]$ ：這種具有「禁止」意味的狀態轉換，我們會將它改以 $e[fcond \wedge \neg G]/a$這種表示法，這種也是懶惰型的狀態轉換。如下圖一所示：



圖一: 懶惰型的狀態轉換

- $e[fcond]/a$ $[l, u)$ $\lceil[G]$ ：這是具有「命令」意義的狀態轉換。這個狀態轉換t要轉成二個子部份：首先是功能層，將$e[fcond]/a$轉成t_1，這個子狀態轉換是懶惰型的。另一個子狀態轉換為t_2，包括二個部份：t_ε及t_δ。其中，t_ε為$[G]/timer:=0$，$timer$是用來做時間限制的時間變數，t_ε是迫切型的狀態轉換，而且它的目的地是一個中間模式，稱作「轉換中間狀態」，記作 $translator(t)$ 。而t_δ的起始狀態就是 $translator(t)$，而它的目的地和t相同。因為這個狀態轉換具有時間限制，必須在$[timer \geq l \wedge timer < u]$的條件下發生，所以$t_\delta$是屬於可延遲型。整個轉換結果如下圖二所示：



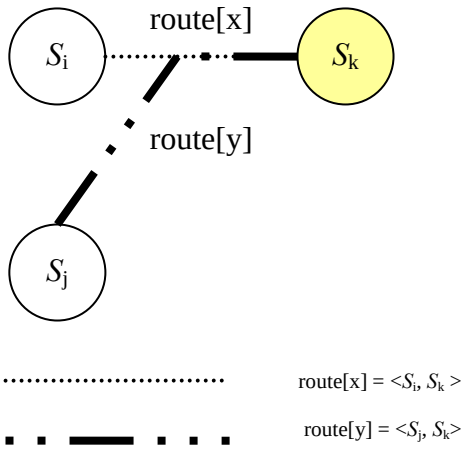
圖二: 可延遲型的狀態轉換

最後一個部份，就是如何描述「禁止」和「命令」相關的特性(properties)。我們對於所要驗證的這些安全的特性，根據轉換的方法，訂定了以下的特性的寫法：

- **↰[G]:** AG (($src(t) \wedge$ G) $\rightarrow \neg$ EX($des(t)$))
- **[l, u) ↱[G]:** AG (($src(t) \wedge G \rightarrow \neg$ EX ($\neg$ $translator(t)$))以及AG($translator(t) \wedge$ $timer < u$)

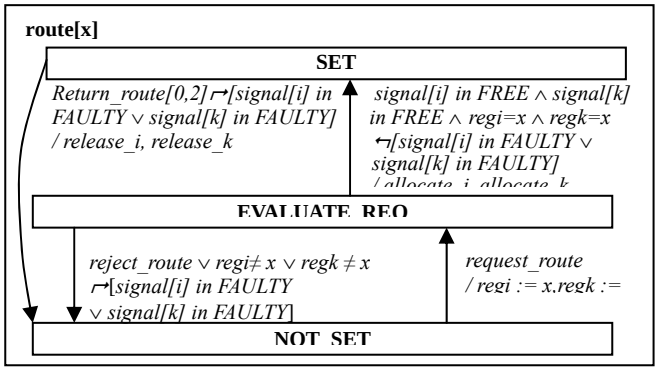
四. 應用實例

我們利用基本的鐵道訊號系統來做為一個實例。這個基本的鐵道訊號系統包括了二個不同的線路(routes)，分別表示為 route[x]及route[y]。其中，route[x]需要使用 signal[i]及signal[k]，而route[y]則需要signal[j]及signal[k]。例如圖三所表示：



圖三: 鐵道訊號系統的線路分佈

接著，我們要以安全圖來仔細描述圖三的鐵道訊號系統。在圖四的安全圖中，除了表示此系統之功能性和安全性，最重要的設計是對於共同的訊號 signal[k]也要滿足互斥性質(mutual exclusion)，避免同一時間內，route[x]及 route[y]都能同時存取到 signal[k]，而造成二個鐵道路線上的火車相撞。



圖四: 路線 x(route[x])的安全圖

需要驗證的安全特性：

1. AG(¬(route[x] in SET ∧ route[y] in SET))
2. AG(signal[i] in FAULTY → ¬(route[x] in SET))
3. AG(signal[j] in FAULTY → ¬(route[y] in SET))
4. AG(signal[k] in FAULTY → ¬(route[x] in SET ∨ route[y] in SET))

實作結果：

依我們所提出的方法，實際將這個例子模型化後，輸入由我們發展的一套模型驗證的工具－「狀態圖形操作器」(SGM)，我們可以得到以下表一的數據：

表一: 實例結果

元件名		每個安全圖		每個ETA		
名稱	#	|S|	|T|	#	|M|	|T|
route[x,y]	2	4	7	1	5	13
signal[i,j,k]	3	16	16	5	10	16

其中#表示個數，|S|表示每個安全圖中狀態(states)的個數，|M|表示每個安全圖中模式(modes)的個數，|T|表示狀態轉換(transitions)的個數。在整個系統，每個安全圖中則有56個狀態，62條狀態轉換；而轉成廣泛的真時狀態機(ETA)之後，則共有17個ETAs，每個ETA中有40個模式，74條狀態轉換。並且符合所要驗證的四個安全特性。這整個過程，則大約耗時230μs，需要記憶體空間約0.12 MB。

五、結論

在目前的生活中，越來越多的系統設計是和安全問題息息相關的。為了避免發生悲劇，我們提出了一個以正規驗證的方法來驗證所設計的系統是否真正達到安全特性的要求。只要能夠以模型描述的系統，都可以利用我們提出的方式做到模型驗證。這麼一來，也更加保障我們能生活在一個更安全的環境中。