

ECS 235B Module 12

Typed Access Matrix Model

Typed Access Matrix Model

- Like ACM, but with set of types T
 - All subjects, objects have types
 - Set of types for subjects TS
- Protection state is (S, O, τ, A)
 - $\tau: O \rightarrow T$ specifies type of each object
 - If \mathbf{X} subject, $\tau(\mathbf{X})$ in TS
 - If \mathbf{X} object, $\tau(\mathbf{X})$ in $T - TS$

Create Rules

- Subject creation
 - **create subject s of type ts**
 - s must not exist as subject or object when operation executed
 - $ts \in TS$
- Object creation
 - **create object o of type to**
 - o must not exist as subject or object when operation executed
 - $to \in T - TS$

Create Subject

- Precondition: $s \notin S$
- Primitive command: **create subject s of type t**
- Postconditions:
 - $S' = S \cup \{s\}, O' = O \cup \{s\}$
 - $(\forall y \in O)[\tau'(y) = \tau(y)], \tau'(s) = t$
 - $(\forall y \in O')[a'[s, y] = \emptyset], (\forall x \in S')[a'[x, s] = \emptyset]$
 - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

Create Object

- Precondition: $o \notin O$
- Primitive command: **create object o of type t**
- Postconditions:
 - $S' = S, O' = O \cup \{o\}$
 - $(\forall y \in O)[\tau'(y) = \tau(y)], \tau'(o) = t$
 - $(\forall x \in S')[a'[x, o] = \emptyset]$
 - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

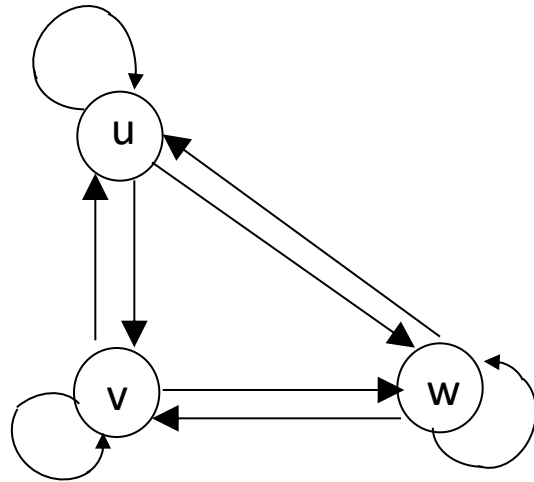
Definitions

- MTAM Model: TAM model without **delete, destroy**
 - MTAM is Monotonic TAM
- $\alpha(x_1:t_1, \dots, x_n:t_n)$ create command
 - t_i child type in α if any of **create subject x_i of type t_i** or **create object x_i of type t_i** occur in α
 - t_i parent type otherwise

Cyclic Creates

```
command cry•havoc( $s_1 : u, s_2 : u, o_1 : v, o_2 : v,$   
                   $o_3 : w, o_4 : w$ )  
  
  create subject  $s_1$  of type  $u$ ;  
  create object  $o_1$  of type  $v$ ;  
  create object  $o_3$  of type  $w$ ;  
  
  enter  $r$  into  $a[s_2, s_1]$ ;  
  enter  $r$  into  $a[s_2, o_2]$ ;  
  enter  $r$  into  $a[s_2, o_4]$   
  
end
```

Creation Graph

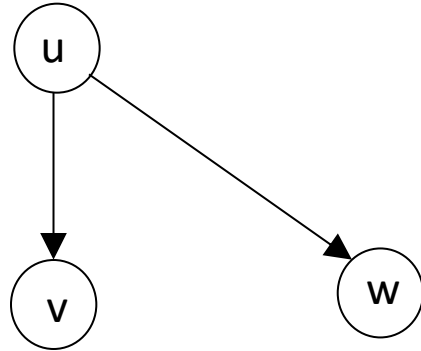


- u, v, w child types
- u, v, w also parent types
- Graph: lines from parent types to child types
- This one has cycles

Acyclic Creates

```
command cry•havoc( $s_1 : u, s_2 : u, o_1 : v, o_3 : w$ )  
  create object  $o_1$  of type  $v$ ;  
  create object  $o_3$  of type  $w$ ;  
  enter  $r$  into  $a[s_2, s_1]$ ;  
  enter  $r$  into  $a[s_2, o_1]$ ;  
  enter  $r$  into  $a[s_2, o_3]$   
end
```

Creation Graph



- v , w child types
- u parent type
- Graph: lines from parent types to child types
- This one has no cycles

Theorems

- Safety decidable for systems with acyclic MTAM schemes
 - In fact, it's *NP-hard*
- Safety for acyclic ternary MATM decidable in time polynomial in the size of initial ACM
 - “Ternary” means commands have no more than 3 parameters
 - Equivalent in expressive power to MTAM