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 **X** subject, $\tau(\mathbf{X})$ in TS
  - If **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:
  - $S' = S \cup \{ s \}$, $O' = O \cup \{ s \}$
  - $(\forall y \in O)[\tau'(y) = \tau(y)]$, $\tau'(s) = t$
  - $(\forall y \in O')[a'[s, y] = \varnothing]$, $(\forall x \in S')[a'[x, s] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

Create Object

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

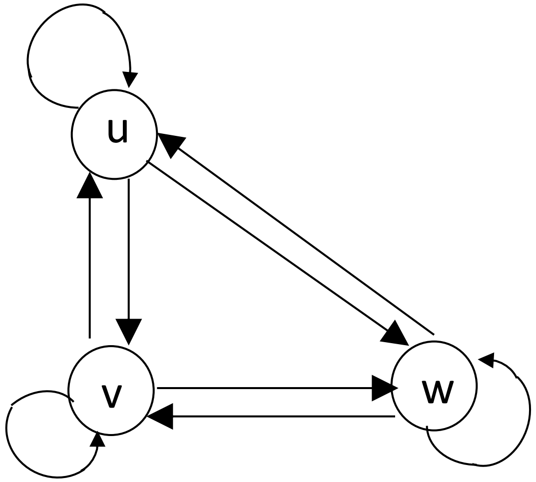
Definitions

- MTAM Model: TAM model without **delete**, **destroy**
  - MTAM is Monotonic TAM
- $\alpha(x_1{:}t_1, \ldots, x_n{:}t_n)$ 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(s1 : u, s2 : u, o1 : v, o2 : v,
                                    o3 : w, o4 : w)
  create subject s1 of type u;
  create object o1 of type v;
  create object o3 of type w;
  enter r into a[s2, s1];
  enter r into a[s2, o2];
  enter r into a[s2, o4]
end
```

Creation Graph

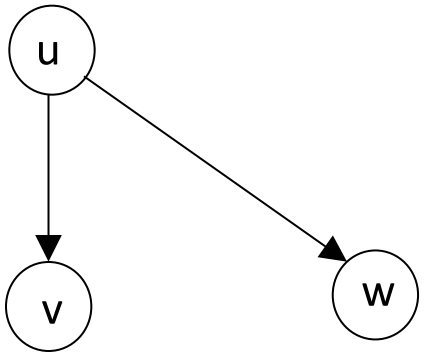


- *u*, *v*, *w* child types
- *u*, *v*, *w* also parent types
- Graph: lines from parent types to child types
- This one has cycles

Acyclic Creates

```
command cry•havoc(s1 : u, s2 : u, o1 : v, o3 : w)
  create object o1 of type v;
  create object o3 of type w;
  enter r into a[s2, s1];
  enter r into a[s2, o1];
  enter r into a[s2, o3]
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