Lecture #7

- Schematic Protection Model
  - Safety question
- Expressive Power
  - HRU and SPM
- Multiparent create
  - ESPM
Main Theorem

- System has acyclic attenuating scheme
- For every history \( H \) deriving state \( h \) from initial state, there is a history \( G \) without create operations that derives \( g \) from the fully unfolded state \( u \) such that
  \[
  (\forall X,Y \in SUB^h)[flow^h(X, Y) \subseteq flow^g(\sigma(X), \sigma(Y))]
  \]
- Meaning: any history derived from an initial state can be simulated by corresponding history applied to the fully unfolded state derived from the initial state
Proof

• Outline of proof: show that every $\text{path}^h(X,Y)$ has corresponding $\text{path}^g(\sigma(X), \sigma(Y))$ such that $\text{cap}(\text{path}^h(X,Y)) = \text{cap}(\text{path}^g(\sigma(X), \sigma(Y)))$
  – Then corresponding sets of tickets flow through systems derived from $H$ and $G$
  – As initial states correspond, so do those systems

• Proof by induction on number of links
Basis and Hypothesis

• Length of $path^h(X, Y) = 1$. By definition of $path^h$, $link^h_i(X, Y)$, hence $link^g_i(\sigma(X), \sigma(Y))$. As $\sigma$ preserves type, this means $\text{cap}(path^h(X, Y)) = \text{cap}(path^g(\sigma(X), \sigma(Y)))$

• Now assume this is true when $path^h(X, Y)$ has length $k$
Step

• Let $path^h(X, Y)$ have length $k+1$. Then there is a $Z$ such that $path^h(X, Z)$ has length $k$ and $link^h_j(Z, Y)$.
• By IH, there is a $path^g(\sigma(X), \sigma(Z))$ with same capacity as $path^h(X, Z)$
• By corollary, $link^g_j(\sigma(Z), \sigma(Y))$
• As $\sigma$ preserves type, there is $path^g(\sigma(X), \sigma(Y))$ with

$$\text{cap}(path^h(X, Y)) = \text{cap}(path^g(\sigma(X), \sigma(Y)))$$
Implication

• Let maximal state corresponding to \( v \) be \( #u \)
  - Deriving history has no creates
  - By theorem,
    \[
    (\forall X, Y \in SUB^h)[flow^h(X, Y) \subseteq flow^{#u}(\sigma(X), \sigma(Y))]
    \]
    - If \( X \in SUB^0, \sigma(X) = X \), so:
      \[
      (\forall X, Y \in SUB^0)[flow^h(X, Y) \subseteq flow^{#u}(X, Y)]
      \]
  
• So \( #u \) is maximal state for system with acyclic attenuating scheme
  - \( #u \) derivable from \( u \) in time polynomial to \( |SUB^u| \)
  - Worst case computation for \( flow^{#u} \) is exponential in \( |TS| \)
Safety Result

• If the scheme is acyclic and attenuating, the safety question is decidable
Expressive Power

• How do the sets of systems that models can describe compare?
  – If HRU equivalent to SPM, SPM provides more specific answer to safety question
  – If HRU describes more systems, SPM applies only to the systems it can describe
HRU vs. SPM

- SPM more abstract
  - Analyses focus on limits of model, not details of representation
- HRU allows revocation
  - SMP has no equivalent to delete, destroy
- HRU allows multiparent creates
  - SMP cannot express multiparent creates easily, and not at all if the parents are of different types because can\textbullet create allows for only one type of creator
Multiparent Create

• Solves mutual suspicion problem
  – Create proxy jointly, each gives it needed rights

• In HRU:

```
command multicreate(s_0, s_1, o)
if r in a[s_0, s_1] and r in a[s_1, s_0]
then
  create object o;
  enter r into a[s_0, o];
  enter r into a[s_1, o];
end
```
SPM and Multiparent Create

- **cc extended in obvious way**
  - \( cc \subseteq TS \times \ldots \times TS \times T \)
- **Symbols**
  - \( X_1, \ldots, X_n \) parents, \( Y \) created
  - \( R_{1,i}, R_{2,i}, R_{3}, R_{4,i} \subseteq R \)
- **Rules**
  - \( cr_{P,i}(\tau(X_1), \ldots, \tau(X_n)) = Y/R_{1,1} \cup X_i/R_{2,i} \)
  - \( cr_{C}(\tau(X_1), \ldots, \tau(X_n)) = Y/R_{3} \cup X_1/R_{4,1} \cup \ldots \cup X_n/R_{4,n} \)
Example

- Anna, Bill must do something cooperatively
  - But they don’t trust each other
- Jointly create a proxy
  - Each gives proxy only necessary rights
- In ESPM:
  - Anna, Bill type $a$; proxy type $p$; right $x \in R$
  - $cc(a, a) = p$
  - $cr_{Anna}(a, a, p) = cr_{Bill}(a, a, p) = \emptyset$
  - $cr_{proxy}(a, a, p) = \{ \text{Anna}/x, \text{Bill}/x \}$
2-Parent Joint Create Suffices

- Goal: emulate 3-parent joint create with 2-parent joint create
- Definition of 3-parent joint create (subjects $P_1, P_2, P_3$; child $C$):
  - $cc(\tau(P_1), \tau(P_2), \tau(P_3)) = Z \subseteq T$
  - $cr_{P_1}(\tau(P_1), \tau(P_2), \tau(P_3)) = C/R_{1,1} \cup P_1/R_{2,1}$
  - $cr_{P_2}(\tau(P_1), \tau(P_2), \tau(P_3)) = C/R_{2,1} \cup P_2/R_{2,2}$
  - $cr_{P_3}(\tau(P_1), \tau(P_2), \tau(P_3)) = C/R_{3,1} \cup P_3/R_{2,3}$
General Approach

• Define agents for parents and child
  – Agents act as surrogates for parents
  – If create fails, parents have no extra rights
  – If create succeeds, parents, child have exactly same rights as in 3-parent creates
    • Only extra rights are to agents (which are never used again, and so these rights are irrelevant)
Entities and Types

- Parents $P_1, P_2, P_3$ have types $p_1, p_2, p_3$
- Child $C$ of type $c$
- Parent agents $A_1, A_2, A_3$ of types $a_1, a_2, a_3$
- Child agent $S$ of type $s$
- Type $t$ is parentage
  - if $X/t \in dom(Y)$, $X$ is $Y$’s parent
- Types $t, a_1, a_2, a_3, s$ are new types
Can•Create

• Following added to can•create:
  – \( \text{cc}(p_1) = a_1 \)
  – \( \text{cc}(p_2, a_1) = a_2 \)
  – \( \text{cc}(p_3, a_2) = a_3 \)
    • Parents creating their agents; note agents have maximum of 2 parents
  – \( \text{cc}(a_3) = s \)
    • Agent of all parents creates agent of child
  – \( \text{cc}(s) = c \)
    • Agent of child creates child
Creation Rules

• Following added to create rule:
  
  – \( cr_p(p_1, a_1) = \emptyset \)
  
  – \( cr_c(p_1, a_1) = p_1/Rtc \)
    
      • Agent’s parent set to creating parent; agent has all rights over parent
  
  – \( cr_{P_{\text{first}}}(p_2, a_1, a_2) = \emptyset \)
  
  – \( cr_{P_{\text{second}}}(p_2, a_1, a_2) = \emptyset \)
  
  – \( cr_c(p_2, a_1, a_2) = p_2/Rtc \cup a_1/tc \)
    
      • Agent’s parent set to creating parent and agent; agent has all rights over parent (but not over agent)
Creation Rules

- \( cr_{P_{first}}(p_3, a_2, a_3) = \emptyset \)
- \( cr_{P_{second}}(p_3, a_2, a_3) = \emptyset \)
- \( cr_C(p_3, a_2, a_3) = p_3/Rtc \cup a_2/tc \)
  - Agent’s parent set to creating parent and agent; agent has all rights over parent (but not over agent)
- \( cr_p(a_3, s) = \emptyset \)
- \( cr_C(a_3, s) = a_3/tc \)
  - Child’s agent has third agent as parent \( cr_p(a_3, s) = \emptyset \)
- \( cr_p(s, c) = C/Rtc \)
- \( cr_C(s, c) = c/R_3t \)
  - Child’s agent gets full rights over child; child gets \( R_3 \) rights over agent
Link Predicates

- Idea: no tickets to parents until child created
  - Done by requiring each agent to have its own parent rights
  - \( \text{link}_1(A_1, A_2) = A_1/t \in \text{dom}(A_2) \land A_2/t \in \text{dom}(A_2) \)
  - \( \text{link}_1(A_2, A_3) = A_2/t \in \text{dom}(A_3) \land A_3/t \in \text{dom}(A_3) \)
  - \( \text{link}_2(S, A_3) = A_3/t \in \text{dom}(S) \land C/t \in \text{dom}(C) \)
  - \( \text{link}_3(A_1, C) = C/t \in \text{dom}(A_1) \)
  - \( \text{link}_3(A_2, C) = C/t \in \text{dom}(A_2) \)
  - \( \text{link}_3(A_3, C) = C/t \in \text{dom}(A_3) \)
  - \( \text{link}_4(A_1, P_1) = P_1/t \in \text{dom}(A_1) \land A_1/t \in \text{dom}(A_1) \)
  - \( \text{link}_4(A_2, P_2) = P_2/t \in \text{dom}(A_2) \land A_2/t \in \text{dom}(A_2) \)
  - \( \text{link}_4(A_3, P_3) = P_3/t \in \text{dom}(A_3) \land A_3/t \in \text{dom}(A_3) \)
Filter Functions

- $f_1(a_2, a_1) = a_1/t \cup c/Rtc$
- $f_1(a_3, a_2) = a_2/t \cup c/Rtc$
- $f_2(s, a_3) = a_3/t \cup c/Rtc$
- $f_3(a_1, c) = p_1/R_{4,1}$
- $f_3(a_2, c) = p_2/R_{4,2}$
- $f_3(a_3, c) = p_3/R_{4,3}$
- $f_4(a_1, p_1) = c/R_{1,1} \cup p_1/R_{2,1}$
- $f_4(a_2, p_2) = c/R_{1,2} \cup p_2/R_{2,2}$
- $f_4(a_3, p_3) = c/R_{1,3} \cup p_3/R_{2,3}$
Construction

Create $A_1, A_2, A_3, S, C$; then

- $P_1$ has no relevant tickets
- $P_2$ has no relevant tickets
- $P_3$ has no relevant tickets
- $A_1$ has $P_1/Rtc$
- $A_2$ has $P_2/Rtc \cup A_1/tc$
- $A_3$ has $P_3/Rtc \cup A_2/tc$
- $S$ has $A_3/tc \cup C/Rtc$
- $C$ has $C/R_3$
Construction

- Only $link_2(S, A_3)$ true $\Rightarrow$ apply $f_2$
  - $A_3$ has $P_3/Rtc \cup A_2/t \cup A_3/t \cup C/Rtc$
- Now $link_1(A_3, A_2)$ true $\Rightarrow$ apply $f_1$
  - $A_2$ has $P_2/Rtc \cup A_1/tc \cup A_2/t \cup C/Rtc$
- Now $link_1(A_2, A_1)$ true $\Rightarrow$ apply $f_1$
  - $A_1$ has $P_2/Rtc \cup A_1/tc \cup A_1/t \cup C/Rtc$
- Now all $link_3$s true $\Rightarrow$ apply $f_3$
  - $C$ has $C/R_3 \cup P_1/R_{4,1} \cup P_2/R_{4,2} \cup P_3/R_{4,3}$
Finish Construction

• Now $link_4$ is true $\Rightarrow$ apply $f_4$
  - $P_1$ has $C/R_{1,1} \cup P_1/R_{2,1}$
  - $P_2$ has $C/R_{1,2} \cup P_2/R_{2,2}$
  - $P_3$ has $C/R_{1,3} \cup P_3/R_{2,3}$

• 3-parent joint create gives same rights to $P_1, P_2, P_3, C$

• If create of $C$ fails, $link_2$ does not hold, so construction fails
Theorem

- The two-parent joint creation operation can implement an $n$-parent joint creation operation with a fixed number of additional types and rights, and augmentations to the link predicates and filter functions.

- **Proof**: by construction, as above
  - Difference is that the two systems need not start at the same initial state
Theorems

• Monotonic ESPM and the monotonic HRU model are equivalent.

• Safety question in ESPM also decidable if acyclic attenuating scheme
  – Proof similar to that for SPM
Expressiveness

- Graph-based representation to compare models
- Graph
  - Vertex: represents entity, has static type
  - Edge: represents right, has static type
- Graph rewriting rules:
  - Initial state operations create graph in a particular state
  - Node creation operations add nodes, incoming edges
  - Edge adding operations add new edges between existing vertices
Example: 3-Parent Joint Creation

- Simulate with 2-parent
  - Nodes $P_1, P_2, P_3$ parents
  - Create node $C$ with type $c$ with edges of type $e$
  - Add node $A_1$ of type $a$ and edge from $P_1$ to $A_1$ of type $e''$

```
P_1
  -
  ↓
A_1

P_2

P_3
```
Next Step

- $A_1, P_2$ create $A_2$; $A_2, P_3$ create $A_3$
- Type of nodes, edges are $a$ and $e'$

![Diagram showing nodes and edges connecting $A_1$, $P_2$, $A_2$, $P_3$, and $A_3$.]
Next Step

- $A_3$ creates $S$, of type $a$
- $S$ creates $C$, of type $c$
Last Step

- Edge adding operations:
  - $P_1 \rightarrow A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow S \rightarrow C$: $P_1$ to $C$ edge type $e$
  - $P_2 \rightarrow A_2 \rightarrow A_3 \rightarrow S \rightarrow C$: $P_2$ to $C$ edge type $e$
  - $P_3 \rightarrow A_3 \rightarrow S \rightarrow C$: $P_3$ to $C$ edge type $e$
Definitions

• *Scheme*: graph representation as above
• *Model*: set of schemes
• Schemes $A, B$ correspond if graph for both is identical when all nodes with types not in $A$ and edges with types in $A$ are deleted
Example

• Above 2-parent joint creation simulation in scheme \textit{TWO}

• Equivalent to 3-parent joint creation scheme \textit{THREE} in which $P_1, P_2, P_3, C$ are of same type as in \textit{TWO}, and edges from $P_1, P_2, P_3$ to $C$ are of type $e$, and no types $a$ and $e'$ exist in \textit{TWO}
Simulation

Scheme $A$ simulates scheme $B$ iff

• every state $B$ can reach has a corresponding state in $A$ that $A$ can reach; and

• every state that $A$ can reach either corresponds to a state $B$ can reach, or has a successor state that corresponds to a state $B$ can reach
  
  – The last means that $A$ can have intermediate states not corresponding to states in $B$, like the intermediate ones in $TWO$ in the simulation of $THREE$
Expressive Power

• If there is a scheme in $MA$ that no scheme in $MB$ can simulate, $MB$ less expressive than $MA$

• If every scheme in $MA$ can be simulated by a scheme in $MB$, $MB$ as expressive as $MA$

• If $MA$ as expressive as $MB$ and vice versa, $MA$ and $MB$ equivalent
Example

• Scheme A in model $M$
  – Nodes $X_1, X_2, X_3$
  – 2-parent joint create
  – 1 node type, 1 edge type
  – No edge adding operations
  – Initial state: $X_1, X_2, X_3$, no edges

• Scheme B in model $N$
  – All same as A except no 2-parent joint create
  – 1-parent create

• Which is more expressive?
Can A Simulate B?

- Scheme A simulates 1-parent create: have both parents be same node
  - Model $M$ as expressive as model $N$
Can $B$ Simulate $A$?

- Suppose $X_1, X_2$ jointly create $Y$ in $A$
  - Edges from $X_1, X_2$ to $Y$, no edge from $X_3$ to $Y$
- Can $B$ simulate this?
  - Without loss of generality, $X_1$ creates $Y$
  - Must have edge adding operation to add edge from $X_2$ to $Y$
  - One type of node, one type of edge, so operation can add edge between any 2 nodes
No

- All nodes in A have even number of incoming edges
  - 2-parent create adds 2 incoming edges
- Edge adding operation in B that can edge from $X_2$ to C can add one from $X_3$ to C
  - A cannot enter this state
    - A cannot have node (C) with 3 incoming edges
  - B cannot transition to a state in which Y has even number of incoming edges
    - No remove rule

- So B cannot simulate A; N less expressive than M
Theorem

- Monotonic single-parent models are less expressive than monotonic multiparent models
- Proof by contradiction
  - Scheme $A$ is multiparent model
  - Scheme $B$ is single parent create
  - Claim: $B$ can simulate $A$, without assumption that they start in the same initial state
    - Note: example assumed same initial state
Outline of Proof

- **X<sub>1</sub>, X<sub>2</sub>** nodes in **A**
  - They create **Y<sub>1</sub>, Y<sub>2</sub>, Y<sub>3</sub>** using multiparent create rule
  - **Y<sub>1</sub>, Y<sub>2</sub>** create **Z**, again using multiparent create rule
  - *Note*: no edge from **Y<sub>3</sub>** to **Z** can be added, as **A** has no edge-adding operation
Outline of Proof

- **W, X₁, X₂ nodes in B**
  - **W** creates **Y₁, Y₂, Y₃** using single parent create rule, and adds edges for **X₁, X₂** to all using edge adding rule
  - **Y₁** creates **Z**, again using single parent create rule; now must add edge from **X₂** to **Z** to simulate **A**
  - Use same edge adding rule to add edge from **Y₃** to **Z**: cannot duplicate this in scheme **A**!
Meaning

• Scheme $B$ cannot simulate scheme $A$, contradicting hypothesis

• ESPM more expressive than SPM
  – ESPM multiparent and monotonic
  – SPM monotonic but single parent