Decidability

January 21, 2014

Formal Definition of Maximal State

- Definition: $g \leq_0 h$ holds iff for all $\mathbf{X}, \mathbf{Y} \in SUB^0$, $flow^g(\mathbf{X}, \mathbf{Y}) \subseteq flow^h(\mathbf{X}, \mathbf{Y})$
  - Note: if $g \leq_0 h$ and $h \leq_0 g$, then g, h are equivalent states
  - Defines set of equivalence classes on set of derivable states
- Definition: for a given system, state m is maximal iff $h \leq_0 m$ for every derivable state h
- Intuition: flow function contains all tickets that can be transferred from one subject to another
  - All maximal states in same equivalence class, answering first question (uniqueness of maximal state)

Useful Lemma

Lemma. Given an arbitrary finite set of states H, there exists a derivable state m such that for all $h \in H$, $h \leq_0 m$

Proof of Useful Lemma

By induction on the size of H

BASIS: For $H = \varnothing$, $|H| = 0$, claim is trivially true

INDUCTION HYPOTHESIS: For $|H| = n$, claim holds

INDUCTION STEP: $|H'| = n + 1$, where $H' = G \cup \{h\}$. By hypothesis, there is a $g \in G$ such that $x \leq_0 g$ for all $x \in G$
Let M be an interleaving of histories of g, h, which:

- Preserves relative order of transitions in g, h
- Omits second create operation if duplicated

M ends up in state m
If $path^g(\mathbf{X}, \mathbf{Y})$ for $\mathbf{X}, \mathbf{Y} \in SUB^g$, $path^m(\mathbf{X}, \mathbf{Y})$, so $g \leq_0 m$
If $path^h(\mathbf{X}, \mathbf{Y})$ for $\mathbf{X}, \mathbf{Y} \in SUB^h$, $path^m(\mathbf{X}, \mathbf{Y})$, so $h \leq_0 m$
Hence m is a maximal state in H'

Answer to "Does Every System Have a Maximal State"

Theorem: every system has a maximal state *

Outline of proof: Let K be the set of derivable states containing exactly one state from each equivalence class of derivable states

- Let $\mathbf{X}, \mathbf{Y} \in SUB^0$.
- Flow function's range is $2^{T \times R}$, so it can take on at most $|2^{T \times R}|$ values.
- There are $|SUB^0|^2$ pairs of subjects in SUB^0
- So at most $|2^{T \times R}|\ |SUB^0|^2$ distinct equivalence classes
- So K is finite

So the lemma's conditions hold, giving the answer "yes"

Safety Question

- In this model, is there a derivable state with $\mathbf{X}/r{:}c \in dom(\mathbf{A})$, or does there exist a subject $\mathbf{B}$ with ticket $\mathbf{X}/rc$ in the initial state in $flow^*(\mathbf{B}, \mathbf{A})$?
- To answer: construct maximal state and test
  - Consider acyclic attenuating schemes; how do we construct maximal state?

Intuition

- Consider state h
- State u corresponds to h but with minimal number of new entities created such that maximal state m can be derived with no create operations
    - So if in history from h to m, subject **X** creates two entities of type a, in u only one would be created; surrogate for both
- m can be derived from u in polynomial time, so if u can be created by adding a finite number of subjects to h, safety question decidable

Fully Unfolded State

- State u derived from state 0 as follows:
  - Delete all loops in cc; new relation cc'
  - Mark all subjects as folded
  - While any $\mathbf{X} \in SUB^0$ is folded:
    - Mark it unfolded
    - If **X** can create entity **Y** of type y, it does so (call this the y-surrogate of **X**); if entity $\mathbf{Y} \in SUB^g$, mark it folded
  - If any subject in state h can create an entity of its own type, do so
- Now in state u

Termination

- $|SUB^0|$ is finite, so marking all subjects as folded terminates
- $|SUB^h|$ is finite, so subjects in state h creating entities of their own type terminates
- Consider while loop:
  - Each subject in SUB^0 can create at most $|TS|$ children; $|TS|$ is finite
  - Each folded subject in $|SUB^i|$ can create at most $|TS| - i$ children
  - When $i = |TS|$, subject cannot create more children
- Thus, folding is finite
- Each loop removes one element, so loop terminates

Surrogates

- Intuition: surrogate collapses multiple subjects of same type into single subject that acts for all of them
- Definition: given initial state 0, for every derivable state h define a *surrogate function* $\sigma : ENT^h \rightarrow ENT^h$ by:
  - if $\mathbf{X} \in ENT^0$, then $\sigma(\mathbf{X}) = \mathbf{X}$
  - if $\mathbf{Y}$ creates $\mathbf{X}$ and $\tau(\mathbf{Y}) = \tau(\mathbf{X})$, then $\sigma(\mathbf{X}) = \sigma(\mathbf{Y})$
  - if $\mathbf{Y}$ creates $\mathbf{X}$ and $\tau(\mathbf{Y}) \neq \tau(\mathbf{X})$, then $\sigma(\mathbf{X}) = \tau(\mathbf{Y})$-surrogate of $\sigma(\mathbf{Y})$

Implications

- $\tau(\sigma(\mathbf{A})) = \tau(\mathbf{A})$
- If $\tau(\mathbf{X}) = \tau(\mathbf{Y})$, than $\sigma(\mathbf{X}) = \sigma(\mathbf{Y})$
- If $\tau(\mathbf{X}) \neq \tau(\mathbf{Y})$, then:
  - $\sigma(\mathbf{X})$ creates $\sigma(\mathbf{Y})$ in the construction of u
  - $\sigma(\mathbf{X})$ creates entities $\mathbf{X}'$ of type $\tau(\mathbf{X}) = \tau(\sigma(\mathbf{X}))$
- From these, for a system with an acyclic attenuating scheme, if $\mathbf{X}$ creates $\mathbf{Y}$, then tickets that would be introduced by pretending that $\sigma(\mathbf{X})$ creates $\sigma(\mathbf{Y})$ are in $dom^u(\sigma(\mathbf{X}))$ and $dom^u(\sigma(\mathbf{Y}))$

Deriving Maximal State

- Reorder operations so that all creates come first and replace history with equivalent one using surrogates
- Show maximal state of new history is also that of original history
- Show maximal state can be derived from initial state

Reordering

- H legal history that derives state h from state 0
- Order operations: first create, then demand, then copy operations
- Build new history G from H as follows:
  - Delete all creates
  - "$\mathbf{X}$ demands $\mathbf{Y}/r{:}c$" becomes "$\sigma(\mathbf{X})$ demands $\sigma(\mathbf{Y})/r{:}c$"
  - '$\mathbf{Y}$ copies $\mathbf{X}/r{:}c$ from $\mathbf{Y}$" becomes "$\sigma(\mathbf{Y})$ copies $\sigma(\mathbf{X})/r{:}c$ from $\sigma(\mathbf{Y})$"

Tickets in Parallel

Theorem:

1. All transitions in G legal
2. If $\mathbf{X}/r{:}c \in dom^h(\mathbf{Y})$, then $\sigma(\mathbf{X})/r{:}c \in dom^g(\sigma(\mathbf{Y}))$

Outline of proof: induct on number of copy operations in H

Induction Basis: No Copy Operations

- H has create, demand only; so G has demand only. σ preserves type, so by construction every demand operation in G is legal
- 3 ways for $\mathbf{X}/r{:}c$ to be in $dom^h(\mathbf{Y})$:
  - $\mathbf{X}/r{:}c \in dom^0(\mathbf{Y})$ means $\mathbf{X}$, $\mathbf{Y} \in ENT^0$, so trivially $\sigma(\mathbf{X})/r{:}c \in dom^g(\sigma(\mathbf{Y}))$ holds
  - A create added $\mathbf{X}/r{:}c \in dom^h(\mathbf{Y})$: previous lemma says $\sigma(\mathbf{X})/r{:}c \in dom^g(\sigma(\mathbf{Y}))$ holds
  - A demand added $\mathbf{X}/r{:}c \in dom^h(\mathbf{Y})$: corresponding demand operation in G gives $\sigma(\mathbf{X})/r{:}c \in dom^g(\sigma(\mathbf{Y}))$

Induction Hypothesis

- Claim holds for all histories with k copy operations
- History H has $k+1$ copy operations
  - H' initial sequence of H composed of k copy operations
  - h' state derived from H'

Induction Step ($\sigma(\mathbf{X})$)

- Let G' be a sequence of modified operations corresponding to H'; g' the derived state
  - G' legal history by hypothesis
- Final operation is "$\mathbf{Z}$ copied $\mathbf{X}/r{:}c$ from $\mathbf{Y}$"
  - Construction of G means final operation is "$\sigma(\mathbf{Z})$ copies $\sigma(\mathbf{X})/r{:}c$ from $\sigma(\mathbf{Y})$
  - So h, h' differ by at most $\mathbf{X}/r{:}c \in dom^h(\mathbf{Z})$
  - Result is G has $\sigma(\mathbf{X})/r{:}c \in dom^h(\sigma(\mathbf{Z}))$
- Proves second part of claim

Induction Step (Legal Transitions)

- H' legal, so we have:
  1. $\mathbf{X}/r{:}c \in dom^{h'}(\mathbf{Y})$
  2. $link_i^{h'}(\mathbf{Y}, \mathbf{Z})$
  3. $\tau(\mathbf{X}/r : c) \in f_i(\tau(\mathbf{Y}), \tau(\mathbf{Z}))$
- By IH, 1, and 2, as $\mathbf{X}/r{:}c \in dom^{h'}(\mathbf{Y})$,

$$\sigma(\mathbf{X})/r{:}c \in dom^{g'}(\sigma(\mathbf{Y})) \text{ and } link^{g'}(\sigma(\mathbf{Y}), \sigma(\mathbf{Z}))$$

- As σ preserves type, IH and 3 imply

$$\tau(\sigma(\mathbf{X})/r{:}c) \in f_i(\tau(\sigma(\mathbf{Y})), \tau(\sigma(\mathbf{Z})))$$

- By IH, G' is legal, so G is legal

Corollary

If $link_i^h(\mathbf{X}, \mathbf{Y})$, then $link_i^g(\sigma(\mathbf{X}), \sigma(\mathbf{Y}))$

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

$$(\forall \mathbf{X}, \mathbf{Y} \in SUB^h)[flow^h(\mathbf{X}, \mathbf{Y}) \subseteq flow^g(\sigma(\mathbf{X}), \sigma(\mathbf{Y}))]$$

- 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

- Enough to show that every $path^h(\mathbf{X}, \mathbf{Y})$ has corresponding $path^g(\sigma(\mathbf{X}), \sigma(\mathbf{Y}))$ such that

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

  - Then corresponding sets of tickets flow through systems derived from H and G
  - As initial states correspond, so do those systems
- Prove by induction on the number of links

Induction Basis and Hypothesis

- BASIS: Length of $path^h(\mathbf{X}, \mathbf{Y}) = 1$
  By definition of $path^h$, $link_i^h(\mathbf{X}, \mathbf{Y})$, so $link_i^g(\sigma(\mathbf{X}), \sigma(\mathbf{Y}))$; as σ preserves type, this means

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

- HYPOTHESIS: Now assume this is true when $path^h(\mathbf{X}, \mathbf{Y})$ has length k

Induction Step

Let $path^h(\mathbf{X}, \mathbf{Y})$ have length $k + 1$

- Then there is a **Z** such that $path^h(\mathbf{X}, \mathbf{Z})$ has length k and $link_j^h(\mathbf{Z}, \mathbf{Y})$
- By IH, there is a $path^g(\sigma(\mathbf{X}), \sigma(\mathbf{Z}))$ with same capacity as $path^h(\mathbf{X}, \mathbf{Z})$
- By corollary, $link_j^g(\sigma(\mathbf{Z}), \sigma(\mathbf{Y}))$
- As σ preserves type, there is $path^g(\sigma(\mathbf{X}), \sigma(\mathbf{Y}))$ with

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

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
  - SPM has no equivalent to delete, destroy
- HRU allows multiparent creates
  - SPM cannot express multiparent creates easily, and not at all if the parents are of different types because *cc* (can 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(x, y, o)
    if r in A[x, y] and r in A[y, x]
    then
        create object o;
        enter r into A[x, o];
        enter r into A[y, o];
end
```

SPM and Multiparent Create

- *cc* extended in obvious way
  - $cc \subseteq TS \times \ldots \times TS \times T$
- Symbols
  - $\mathbf{X}_1, \ldots, \mathbf{X}_n$ parents, $\mathbf{Y}$ created
  - $R_{1,i}, R_{2,i}, R_3, R_{4,i} \subseteq R$
- Rules
  - $cr_{\mathbf{P},i}(\tau(\mathbf{X}_1), \ldots, \tau(\mathbf{X}_n)) = \mathbf{Y}/R_{1,i} \cup \mathbf{X}_i/R_{2,i}$
  - $cr_{\mathrm{C}}(\tau(\mathbf{X}_1), \ldots, \tau(\mathbf{X}_n)) = \mathbf{Y}/R_3 \cup \mathbf{X}_1/R_{4,1} \cup \ldots \cup \mathbf{X}_n/R_{4,n}$

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 *E*SPM:
  - Anna, Bill are of type a; proxy is of type p; right $x \in R$
  - $cc(a, a) = p$
  - $cr_{\text{Anna}}(a, a, p) = cr_{\text{Bill}}(a, a, p) = \varnothing$
  - $cr_{\text{proxy}}(a, a, p) = \{\text{Anna}/x, \text{Bill}/x\}$

Does 2-Parent Joint Create Suffice?

- Goal: emulate 3-parent joint create with 2-parent joint create
- Definition of 3-parent joint create (subjects $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$; child $\mathbf{C}$):
  - $cc(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c \subseteq T$
  - $cr_{\mathrm{P},\mathbf{P}_1}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_{1,1} \cup \tau(\mathbf{P}_1)/R_{2,1}$
  - $cr_{\mathrm{P},\mathbf{P}_2}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_{1,1} \cup \tau(\mathbf{P}_2)/R_{2,2}$
  - $cr_{\mathrm{P},\mathbf{P}_1}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_{1,1} \cup \tau(\mathbf{P}_3)/R_{2,3}$
  - $cr_{\mathbf{C}}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_3 \cup \tau(\mathbf{P}_1)/R_{4,1} \cup \tau(\mathbf{P}_2)/R_{4,2} \cup \tau(\mathbf{P}_3)/R_{4,3}$

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 $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ of types p_1, p_2, p_3
- Child $\mathbf{C}$ of type c
- Parent agents $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$ have types a_1, a_2, a_3
- Child agent $\mathbf{S}$ of type s
- Type t is parentage
  - if $\mathbf{X}/t \in dom(\mathbf{Y})$, $\mathbf{X}$ is $\mathbf{Y}$'s parent
- Types t, a_1, a_2, a_3, s are new types

Can Create

- Add the following to the *cc* relation:
  - $cc(p_1) = a_1$
  - $cc(p_2, a_1) = a_2$
  - $cc(p_3, a_2) = a_3$
    - Parents creating their agents; note agents have maximum of 2 parents
  - $cc(a_3) = s$
    - Agent of all parents creates agent of child
  - $cc(s) = c$
    - Agent of child creates child

Create Rules

- Add the following to the create rule:
  - $cr_{\mathrm{P}}(p_1, a_1) = \varnothing$
  - $cr_{\mathrm{C}}(p_1, a_1) = p_1/Rtc$
    - Agent's parent set to creating parent; agent has all rights over parent
  - $cr_{\mathbf{P}_1}(p_2, a_1, a_2) = \varnothing$
  - $cr_{\mathbf{P}_2}(p_2, a_1, a_2) = \varnothing$
  - $cr_{\mathbf{C}}(p_2, a_1, a_2) = p_2/Rtc \cup a_1/tc$
    - Agent's parent set to creating parent and agent; agent has all rights over parent (but not over agent)

Create Rules (*con't*)

- Also add the following to the create rule:
  - $cr_{\mathbf{P}_1}(p_3, a_2, a_3) = \varnothing$
  - $cr_{\mathbf{P}_2}(p_3, a_2, a_3) = \varnothing$
  - $cr_{\mathbf{C}}(p_3, a_2, a_3) = p_3/Rtc \cup a_2/tc$
    - Agent's parent set to creating parent and agent; agent has all rights over parent (but not over agent)
  - $cr_{\mathrm{P}}(a_3, s) = \varnothing$
  - $cr_{\mathrm{C}}(a_3, s) = a_3/tc$
    - Child's agent has third agent as parent $cr_{\mathrm{P}}(a_3, s) = \varnothing$
  - $cr_{\mathrm{P}}(s, c) = \mathbf{C}/Rtc$
  - $cr_{\mathrm{C}}(s, c) = c/R_3 t$
    - 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
  - $link_1(\mathbf{A}_1, \mathbf{A}_2) = \mathbf{A}_1/t \in dom(\mathbf{A}_2) \land \mathbf{A}_2/t \in dom(\mathbf{A}_2)$
  - $link_1(\mathbf{A}_2, \mathbf{A}_3) = \mathbf{A}_2/t \in dom(\mathbf{A}_3) \land \mathbf{A}_3/t \in dom(\mathbf{A}_3)$
  - $link_2(\mathbf{S}, \mathbf{A}_3) = \mathbf{A}_3/t \in dom(\mathbf{S}) \land \mathbf{C}/t \in dom(\mathbf{C})$
  - $link_3(\mathbf{A}_1, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_1)$
  - $link_3(\mathbf{A}_2, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_2)$
  - $link_3(\mathbf{A}_3, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_3)$
  - $link_4(\mathbf{A}_1, \mathbf{P}_1) = \mathbf{P}_1/t \in dom(\mathbf{A}_1) \land \mathbf{A}_1/t \in dom(\mathbf{A}_1)$
  - $link_4(\mathbf{A}_2, \mathbf{P}_2) = \mathbf{P}_2/t \in dom(\mathbf{A}_2) \land \mathbf{A}_2/t \in dom(\mathbf{A}_2)$
  - $link_4(\mathbf{A}_3, \mathbf{P}_3) = \mathbf{P}_3/t \in dom(\mathbf{A}_3) \land \mathbf{A}_3/t \in dom(\mathbf{A}_3)$

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 $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, $\mathbf{S}$, $\mathbf{C}$; then

- $\mathbf{P}_1$ has no relevant tickets
- $\mathbf{P}_2$ has no relevant tickets
- $\mathbf{P}_3$ has no relevant tickets
- $\mathbf{A}_1$ has $\mathbf{P}_1/Rtc$
- $\mathbf{A}_2$ has $\mathbf{P}_2/Rtc \cup \mathbf{A}_1/tc$
- $\mathbf{A}_3$ has $\mathbf{P}_3/Rtc \cup \mathbf{A}_2/tc$
- $\mathbf{S}$ has $\mathbf{A}_3/tc \cup \mathbf{C}/Rtc$
- $\mathbf{C}$ has $\mathbf{C}/R_3$

Construction (*con't*)

Only $link_2(\mathbf{S}, \mathbf{A}_3)$ true $\Rightarrow$ apply f_2

- $\mathbf{A}_3$ has $\mathbf{P}_3/Rtc \cup \mathbf{A}_2/t \cup \mathbf{A}_3/t \cup \mathbf{C}/Rtc$

Now $link_1(\mathbf{A}_3, \mathbf{A}_2)$ true $\Rightarrow$ apply f_1

- $\mathbf{A}_2$ has $\mathbf{P}_2/Rtc \cup \mathbf{A}_1/tc \cup \mathbf{A}_2/t \cup \mathbf{C}/Rtc$

Now $link_1(\mathbf{A}_2, \mathbf{A}_1)$ true $\Rightarrow$ apply f_1

- $\mathbf{A}_1$ has $\mathbf{P}_2/Rtc \cup \mathbf{A}_1/t \cup \mathbf{A}_1/t \cup \mathbf{C}/Rtc$

Now all $link_3$s true $\Rightarrow$ apply f_3

- $\mathbf{C}$ has $\mathbf{C}/R_3 \cup \mathbf{P}_1/R_{4,1} \cup \mathbf{P}_2/R_{4,2} \cup \mathbf{P}_3/R_{4,3}$

Finish Construction

- Now $link_4$ true $\Rightarrow$ apply f_4
    - $\mathbf{P}_1$ has $\mathbf{C}/R_{1,1} \cup \mathbf{P}_1/R_{2,1}$
    - $\mathbf{P}_2$ has $\mathbf{C}/R_{1,2} \cup \mathbf{P}_2/R_{2,2}$
    - $\mathbf{P}_3$ has $\mathbf{C}/R_{1,3} \cup \mathbf{P}_3/R_{2,3}$
- 3-parent joint create gives same rights to $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{C}$
- If create of $\mathbf{C}$ 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.

More Theorems

- The monotonic ESPM model and the monotonic HRU model are equivalent
- The safety question in ESPM also decidable for acyclic attenuating schemes
  - Proof is similar to that for SPM

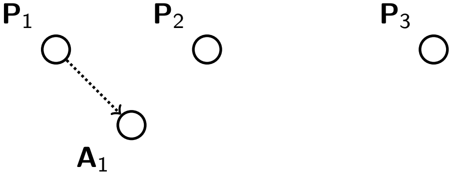
Expressiveness

- Graph-based representation to compare models
- Graph: vertex represents entity, edge represents right; static types
- 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 Create

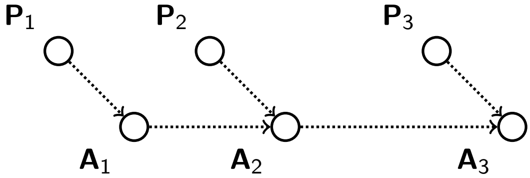
Simulate with 2-parent joint create

- Nodes $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ parents
- Create node **C** with type c with edges of type e
- Add node $\mathbf{A}_1$ of type a and edge from $\mathbf{P}_1$ to $\mathbf{A}_1$ of type e'



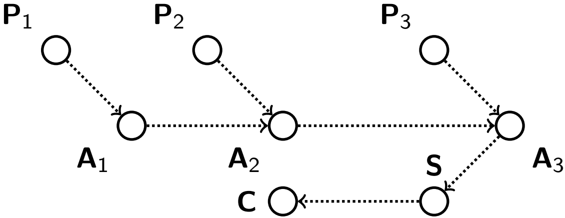
Next Step

- $\mathbf{A}_1$, $\mathbf{P}_2$ create $\mathbf{A}_2$; $\mathbf{A}_2$, $\mathbf{P}_3$ create $\mathbf{A}_3$
- Type of nodes, edges are a and e'



Next Step

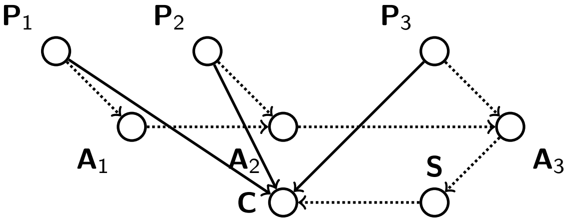
- $\mathbf{A}_3$ creates **S**, of type *a*
- **S** creates **C**, of type *c*



Last Step

Edge adding operations

- $\mathbf{P}_1 \to \mathbf{A}_1 \to \mathbf{A}_2 \to \mathbf{A}_3 \to \mathbf{S} \to \mathbf{C}$: $\mathbf{P}_1$ to $\mathbf{C}$ edge type e
- $\mathbf{P}_2 \to \mathbf{A}_2 \to \mathbf{A}_3 \to \mathbf{S} \to \mathbf{C}$: $\mathbf{P}_2$ to $\mathbf{C}$ edge type e
- $\mathbf{P}_3 \to \mathbf{A}_3 \to \mathbf{S} \to \mathbf{C}$: $\mathbf{P}_2$ to $\mathbf{C}$ 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 *TWO*
- Equivalent to 3-parent joint creation scheme *THREE* in which $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{C}$ are of same type as in *TWO*, and edges from $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ to $\mathbf{C}$ are of type e, and no types a and e' exist in *TWO*

Simulation

Scheme A simulates scheme B iff

1. every state B can reach has a corresponding state in A that A can reach; and
2. 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 $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$
    - 2-parent joint create
    - 1 node type, 1 edge type
    - No edge adding operations
    - Initial state: $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, no edges
- Scheme B in model N
    - All same as A except no 2-parent joint create
    - Has 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 $\mathbf{X}_1$, $\mathbf{X}_2$ jointly create $\mathbf{Y}$ in *A*
  - Edges from $\mathbf{X}_1$, $\mathbf{X}_2$ to $\mathbf{Y}$, no edge from $\mathbf{X}_3$ to $\mathbf{Y}$
- Can *B* simulate this?
  - Without loss of generality, $\mathbf{X}_1$ creates $\mathbf{Y}$
  - Must have edge adding operation to add edge from $\mathbf{X}_2$ to $\mathbf{Y}$
  - One type of node, one type of edge, so operation can add edge between any 2 nodes

B Cannot Simulate *A*

- 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 $\mathbf{X}_2$ to **C** can add one from $\mathbf{X}_3$ 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*; therefore *N* less expressive than *M*

Theorem

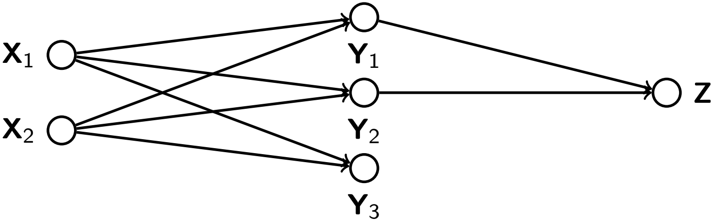
Monotonic single-parent models are less expressive than monotonic multiparent models

Proof: By contradiction

- Scheme A is in multiparent model
- Scheme B is in single parent model
- Claim: B can simulate A, without assumption that they start in the same initial state
    - Note: example assumed same initial state

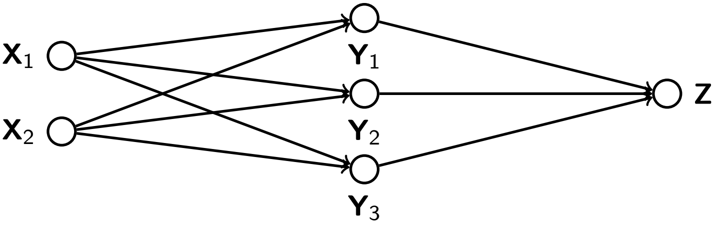
Outline of Proof

- $\mathbf{X}_1$, $\mathbf{X}_2$ nodes in A
  - They create $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$ using multiparent create rule
  - $\mathbf{Y}_1$, $\mathbf{Y}_2$ create $\mathbf{Z}$ using multiparent create rule
  - *Note*: no edge from $\mathbf{Y}_3$ to $\mathbf{Z}$ can be added, as A has no edge-adding operation



Outline of Proof (*con't*)

- **W**, $\mathbf{X}_1$, $\mathbf{X}_2$ nodes in *B*
  - **W** creates $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$ using single parent create rule, and adds edges for $\mathbf{X}_1$, $\mathbf{X}_2$ to all using edge adding rule
  - $\mathbf{Y}_1$ creates **Z** using single parent create rule; now must add edge from $\mathbf{X}_2$ to **Z** to simulate *A*
  - Use same edge adding rule to add edge from $\mathbf{Y}_3$ 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

Typed Access Control Matrix Model (TAM)

- 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:
  - $S' = S \cup \{s\}$, $O' = O \cup \{s\}$
  - $(\forall y \in O)[\tau'(y) = \tau(y)], \tau'(s) = t$
  - $(\forall y \in O')[A'[s, y] = \varnothing]$, $(\forall x \in S')[A'[x, s] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[A'[x, y] = A[x, y]]$

create object

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

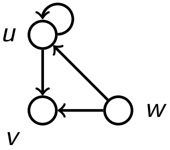
Monotonic Typed Access Control Matrix Model (MTAM)

- TAM without **delete**, **destroy**
- $\alpha(x_1 : t_1, \ldots, x_n : t_n)$ create command
  - t_i is a 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
  - Otherwise t_i is a parent type

Cyclic Creates

```
command havoc(s:u, p:u, f:v, q:w)
    create subject p of type u;
    create object f of type v;
    enter own into A[s,p];
    enter r into A[q,p];
    enter own into A[p,f];
    enter r into A[p,f];
end
```

Creation Graph

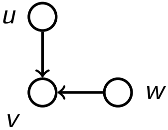


- u, v child types
- u, w parent type
- Graph: lines from parent types to child types
- This one has cycles

Acyclic Creates

```
command ahavoc(s:u, p:u, f:v, q:w)
    create object f of type v;
    enter own into A[s,p];
    enter r into A[q,p];
    enter own into A[p,f];
    enter r into A[p,f];
end
```

Creation Graph



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

Theorem

- Safety decidable for systems with acyclic MTAM schemes
  - In fact, it is 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