Lecture #7

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

Formal Definition

- Definition: $g \leq_0 h$ holds iff for all $\mathbf{X}$, $\mathbf{Y} \in SUB^0$, $\mathit{flow}^g(\mathbf{X},\mathbf{Y}) \subseteq \mathit{flow}^h(\mathbf{X},\mathbf{Y})$.
  - Note: if $g \leq_0 h$ and $h \leq_0 g$, then g, h equivalent
  - 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

Maximal States

- Lemma. Given arbitrary finite set of states H, there exists a derivable state m such that for all $h \in H$, $h \leq_0 m$
- Theorem: every system has a maximal state $*$

Safety Question

- In this model:

  Is there a derivable state with **X**/*r*:*c* $\in$ *dom*(**A**), or does there exist a subject **B** with ticket **X**/*rc* in the initial state in *flow**(**B**,**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 $\mathbf{X}$ can create entity $\mathbf{Y}$ of type y, it does so (call this the y-surrogate of $\mathbf{X}$); 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

- First loop terminates as SUB^0 finite
- Second loop terminates:
  - Each subject in SUB^0 can create at most $| TS |$ children, and $| TS |$ is finite
  - Each folded subject in $| SUB^i |$ can create at most $| TS | - i$ children
  - When $i = | TS |$, subject cannot create more children; thus, folded is finite
  - Each loop removes one element
- Third loop terminates as SUB^h is finite

Surrogate

- 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 *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

- $\tau(\sigma(\mathbf{X})) = \tau(\mathbf{X})$
- If $\tau(\mathbf{X}) = \tau(\mathbf{Y})$, then $\sigma(\mathbf{X}) = \sigma(\mathbf{Y})$
- 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 **X** creates **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

- Idea
  - 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 deriving 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
  - “**X** demands **Y**/*r*:*c*” becomes “σ(**X**) demands σ(**Y**)/*r*:*c*”
  - “**Y** copies **X** /*r*:*c* from **Y**” becomes “σ(**Y**) copies σ(**X**)/*r*:*c* from σ(**Y**)

Tickets in Parallel

- Theorem
  - All transitions in G legal; if $\mathbf{X}/r{:}c \in dom^h(Y)$, then $\sigma(\mathbf{X})/r{:}c \in dom^h(\sigma(\mathbf{Y}))$
- Outline of proof: induct on number of copy operations in H

Basis

- H has create, demand only; so G has demand only. σ preserves type, so by construction every demand operation in G 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 $\sigma(\mathbf{X})/r{:}c \in dom^g(\sigma(\mathbf{Y}))$

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'

Step

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

Step

- H' legal, so for H to be legal, we have:
  1. $\mathbf{X}/rc \in dom^{h'}(\mathbf{Y})$
  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, 2, as $\mathbf{X}/\mathrm{r{:}c} \in dom^{h'}(\mathbf{Y})$,

  $\sigma(\mathbf{X})/r{:}c \in dom^{g'}(\sigma(\mathbf{Y}))$ and $link_i^{g'}(\sigma(\mathbf{Y}), \sigma(\mathbf{Z}))$
- As σ preserves type, IH and 3 imply

  $\tau(\sigma(\mathbf{X})/r{:}c) \in f_i(\tau((\sigma(\mathbf{Y})), \tau(\sigma(\mathbf{Z})))$
- IH says G' 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

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

- Meaning: any history derived from an initial statecan be simulated by corresponding history applied to the fully unfolded state derived from the initial state

Proof

- Outline of proof: 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
- Proof by induction on number of links

Basis and Hypothesis

- Length of $path^h(\mathbf{X}, \mathbf{Y}) = 1$. By definition of $path^h$, $link_i^h(\mathbf{X}, \mathbf{Y})$, hence $link_i^g(\sigma(\mathbf{X}), \sigma(\mathbf{Y}))$. As σ preserves type, this means

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

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

Step

- Let $path^h(\mathbf{X}, \mathbf{Y})$ have length k+1. Then there is a $\mathbf{Z}$ 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

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

Implication

- Let maximal state corresponding to *v* be #*u*
  - Deriving history has no creates
  - By theorem,

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

  - If $\mathbf{X} \in SUB^0$, $\sigma(\mathbf{X}) = \mathbf{X}$, so:

$$(\forall \mathbf{X}, \mathbf{Y} \in SUB^0)[flow^h(\mathbf{X}, \mathbf{Y}) \subseteq flow^{\#u}(\mathbf{X}, \mathbf{Y})]$$

- 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•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
  - $\mathbf{X}_1, \ldots, \mathbf{X}_n$ parents, $\mathbf{Y}$ created
  - $R_{1,i}, R_{2,i}, R_3, R_{4,i} \subseteq R$
- Rules
  - $cr_{\mathrm{P},i}(\tau(\mathbf{X}_1), \ldots, \tau(\mathbf{X}_n)) = \mathbf{Y}/R_{1,1} \cup \mathbf{X}_i/R_{2,i}$
  - $cr_{\mathrm{C}}(\tau(\mathbf{X}_1), \ldots, \tau(\mathbf{X}_n)) = \mathbf{Y}/R_3 \cup \mathbf{X}_1/R_{4,1} \cup \ldots \cup \mathbf{X}_n/R_{4,n}$

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 ESPM:
  - Anna, Bill type a; proxy type p; right $x \in R$
  - $cc(a, a) = p$
  - $cr_{\text{Anna}}(a, a, p) = cr_{\text{Bill}}(a, a, p) = \varnothing$
  - $cr_{\text{proxy}}(a, a, p) = \{ \text{Anna}/x, \text{Bill}/x \}$

2-Parent Joint Create Suffices

- 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 $\mathbf{C}$):
  - $cc(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c \subseteq T$
  - $cr_{\mathbf{P}1}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_{1,1} \cup \tau(\mathbf{P}_1)/R_{2,1}$
  - $cr_{\mathbf{P}2}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_{2,1} \cup \tau(\mathbf{P}_2)/R_{2,2}$
  - $cr_{\mathbf{P}3}(\tau(\mathbf{P}_1), \tau(\mathbf{P}_2), \tau(\mathbf{P}_3)) = c/R_{3,1} \cup \tau(\mathbf{P}_3)/R_{2,3}$

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 $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ have types p_1, p_2, p_3
- Child $\mathbf{C}$ of type c
- Parent agents $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ of 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

- Following added to can•create:
  - 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

Creation Rules

- Following added to create rule:
  - $cr_P(p_1, a_1) = \varnothing$
  - $cr_C(p_1, a_1) = p_1/Rtc$
    - Agent's parent set to creating parent; agent has all rights over parent
  - $cr_{Pfirst}(p_2, a_1, a_2) = \varnothing$
  - $cr_{Psecond}(p_2, a_1, a_2) = \varnothing$
  - $cr_C(p_2, a_1, a_2) = p_2/Rtc \cup a_1/tc$
    - Agent's parent set to creating parent and agent; agent has all rights over parent (but not over agent)

Creation Rules

- $cr_{Pfirst}(p_3, a_2, a_3) = \varnothing$
- $cr_{Psecond}(p_3, a_2, a_3) = \varnothing$
- $cr_C(p_3, a_2, a_3) = p_3/Rtc \cup a_2/tc$
  - Agent's parent set to creating parent and agent; agent has all rights over parent (but not over agent)
- $cr_P(a_3, s) = \varnothing$
- $cr_C(a_3, s) = a_3/tc$
  - Child's agent has third agent as parent $cr_P(a_3, s) = \varnothing$
- $cr_P(s, c) = \mathbf{C}/Rtc$
- $cr_C(s, c) = c/R_3t$
  - 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
  - $link_1(\mathbf{A}_1, \mathbf{A}_2) = \mathbf{A}_1/t \in dom(\mathbf{A}_2) \wedge \mathbf{A}_2/t \in dom(\mathbf{A}_2)$
  - $link_1(\mathbf{A}_2, \mathbf{A}_3) = \mathbf{A}_2/t \in dom(\mathbf{A}_3) \wedge \mathbf{A}_3/t \in dom(\mathbf{A}_3)$
  - $link_2(\mathbf{S}, \mathbf{A}_3) = \mathbf{A}_3/t \in dom(\mathbf{S}) \wedge \mathbf{C}/t \in dom(\mathbf{C})$
  - $link_3(\mathbf{A}_1, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_1)$
  - $link_3(\mathbf{A}_2, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_2)$
  - $link_3(\mathbf{A}_3, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_3)$
  - $link_4(\mathbf{A}_1, \mathbf{P}_1) = \mathbf{P}_1/t \in dom(\mathbf{A}_1) \wedge \mathbf{A}_1/t \in dom(\mathbf{A}_1)$
  - $link_4(\mathbf{A}_2, \mathbf{P}_2) = \mathbf{P}_2/t \in dom(\mathbf{A}_2) \wedge \mathbf{A}_2/t \in dom(\mathbf{A}_2)$
  - $link_4(\mathbf{A}_3, \mathbf{P}_3) = \mathbf{P}_3/t \in dom(\mathbf{A}_3) \wedge \mathbf{A}_3/t \in dom(\mathbf{A}_3)$

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

- 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 $\mathbf{P}_2/Rtc \cup \mathbf{A}_1/tc \cup \mathbf{A}_1/t \cup \mathbf{C}/Rtc$
- 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 $link_4$ is 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 **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

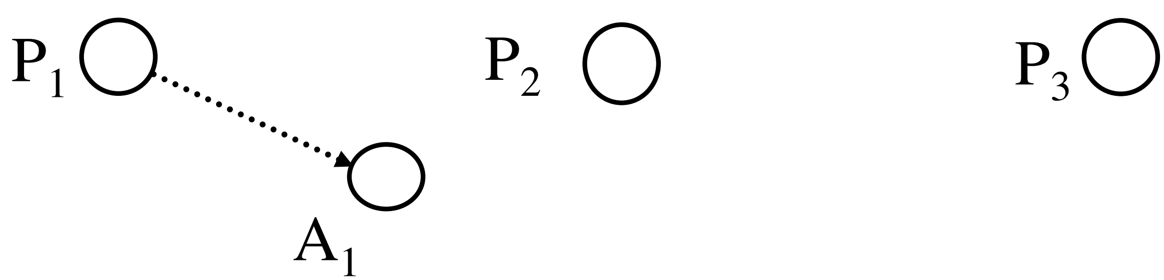
- Monotonic ESPM and the monotonic HRU model are equivalent.
- Safety question in ESPM also decidable if acyclic attenuating scheme
  – Proof similar to that for SPM

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

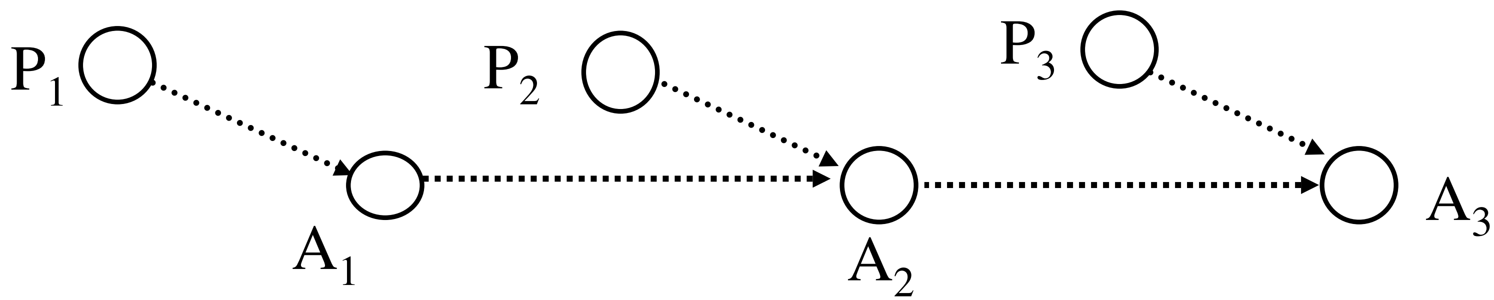
Example: 3-Parent Joint Creation

- Simulate with 2-parent
  - Nodes $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ 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´*



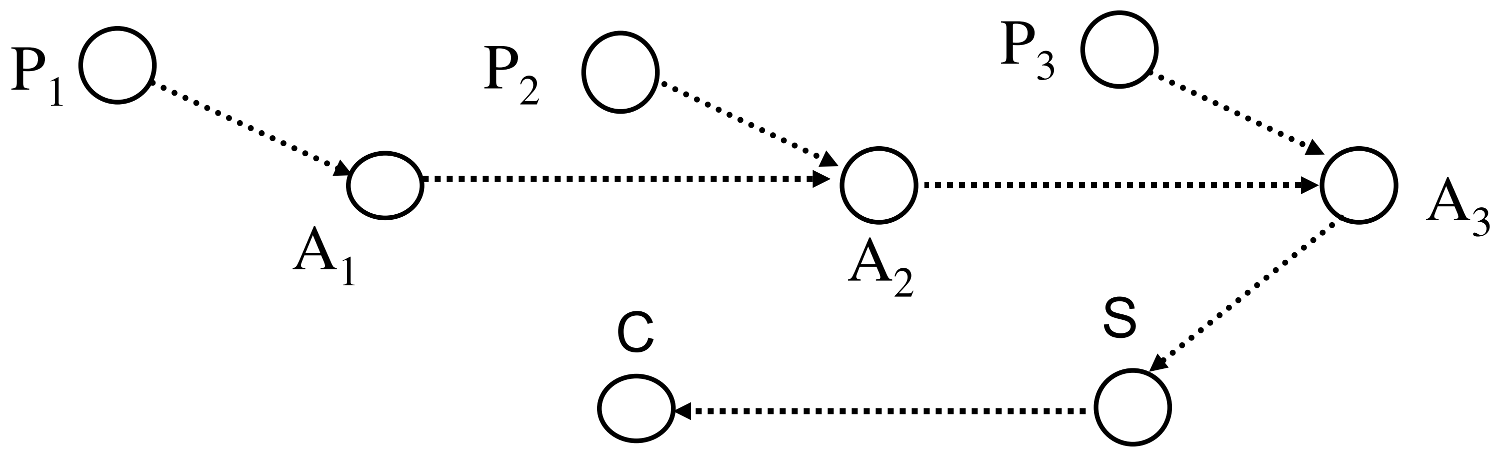
Next Step

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



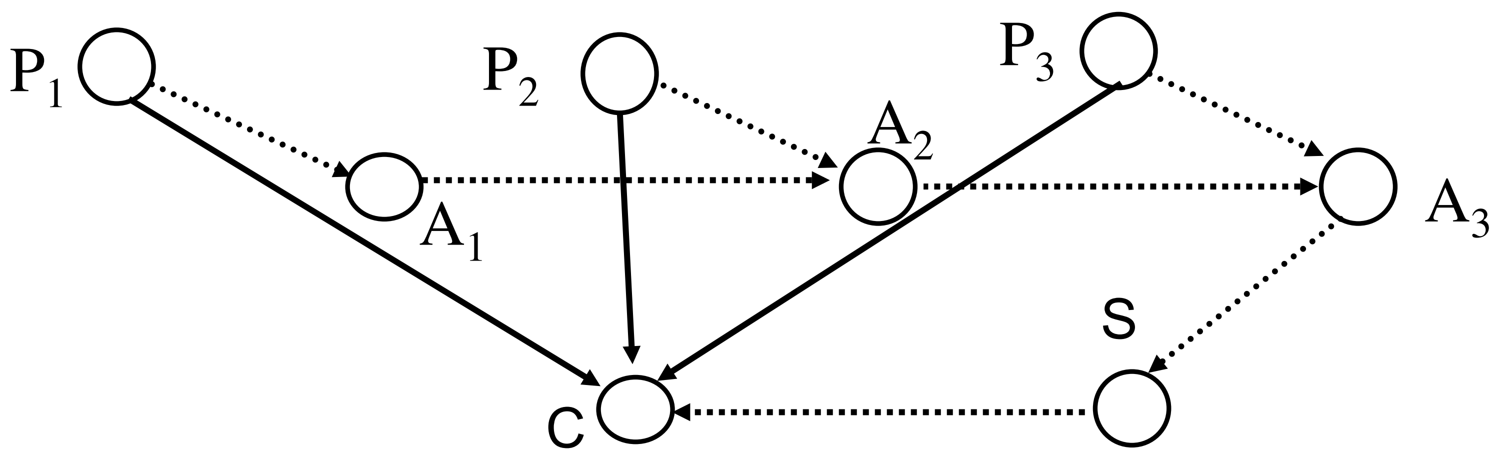
Next Step

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



Last Step

- Edge adding operations:
  - $\mathbf{P}_1 \rightarrow \mathbf{A}_1 \rightarrow \mathbf{A}_2 \rightarrow \mathbf{A}_3 \rightarrow \mathbf{S} \rightarrow \mathbf{C}$: $\mathbf{P}_1$ to **C** edge type *e*
  - $\mathbf{P}_2 \rightarrow \mathbf{A}_2 \rightarrow \mathbf{A}_3 \rightarrow \mathbf{S} \rightarrow \mathbf{C}$: $\mathbf{P}_2$ to **C** edge type *e*
  - $\mathbf{P}_3 \rightarrow \mathbf{A}_3 \rightarrow \mathbf{S} \rightarrow \mathbf{C}$: $\mathbf{P}_3$ 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 *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

- 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*