

ECS 235B Module 11 (*con't*)

Expressiveness

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

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 $\mathbf{X}/t \in \text{dom}(\mathbf{Y})$, \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:
 - $cr_p(p_1, a_1) = \emptyset$
 - $cr_c(p_1, a_1) = p_1/Rtc$
 - Agent's parent set to creating parent; agent has all rights over parent
 - $cr_{pfirst}(p_2, a_1, a_2) = \emptyset$
 - $cr_{psecond}(p_2, a_1, a_2) = \emptyset$
 - $cr_c(p_2, a_1, a_2) = p_2/Rtc \cup a_1/tc$
 - Agent's parent set to creating parent and agent; agent has all rights over parent (but not over agent)

Creation Rules

- $cr_{pfirst}(p_3, a_2, a_3) = \emptyset$
- $cr_{psecond}(p_3, a_2, a_3) = \emptyset$
- $cr_C(p_3, a_2, a_3) = p_3/Rtc \cup a_2/tc$
 - 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 $cr_p(a_3, s) = \emptyset$
- $cr_p(s, c) = \mathbf{C}/Rtc$
- $cr_C(s, c) = c/R_3t$
 - 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
 - $link_1(\mathbf{A}_2, \mathbf{A}_1) = \mathbf{A}_1/t \in dom(\mathbf{A}_2) \wedge \mathbf{A}_2/t \in dom(\mathbf{A}_2)$
 - $link_1(\mathbf{A}_3, \mathbf{A}_2) = \mathbf{A}_2/t \in dom(\mathbf{A}_3) \wedge \mathbf{A}_3/t \in dom(\mathbf{A}_3)$
 - $link_2(\mathbf{S}, \mathbf{A}_3) = \mathbf{A}_3/t \in dom(\mathbf{S}) \wedge \mathbf{C}/t \in dom(\mathbf{C})$
 - $link_3(\mathbf{A}_1, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_1)$
 - $link_3(\mathbf{A}_2, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_2)$
 - $link_3(\mathbf{A}_3, \mathbf{C}) = \mathbf{C}/t \in dom(\mathbf{A}_3)$
 - $link_4(\mathbf{A}_1, \mathbf{P}_1) = \mathbf{P}_1/t \in dom(\mathbf{A}_1) \wedge \mathbf{A}_1/t \in dom(\mathbf{A}_1)$
 - $link_4(\mathbf{A}_2, \mathbf{P}_2) = \mathbf{P}_2/t \in dom(\mathbf{A}_2) \wedge \mathbf{A}_2/t \in dom(\mathbf{A}_2)$
 - $link_4(\mathbf{A}_3, \mathbf{P}_3) = \mathbf{P}_3/t \in dom(\mathbf{A}_3) \wedge \mathbf{A}_3/t \in dom(\mathbf{A}_3)$

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_3t

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 $\mathbf{P}_2/Rtc \cup \mathbf{A}_1/t \cup \mathbf{C}/Rtc$
- 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 \mathbf{C} fails, $link_2$ fails, 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

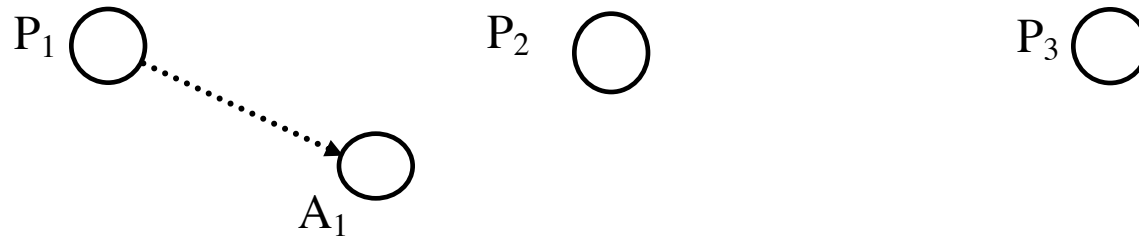
- Monotonic ESPM and the monotonic HRU model are equivalent.
- Safety question in ESPM also decidable if acyclic attenuating scheme
 - Proof similar to that for SPM

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

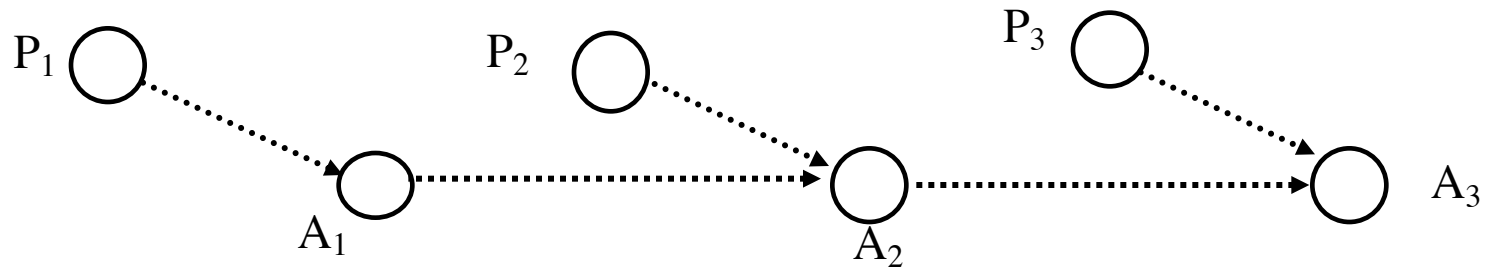
Example: 3-Parent Joint Creation

- Simulate with 2-parent
 - Nodes \mathbf{P}_1 , \mathbf{P}_2 , \mathbf{P}_3 parents
 - Create node \mathbf{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'



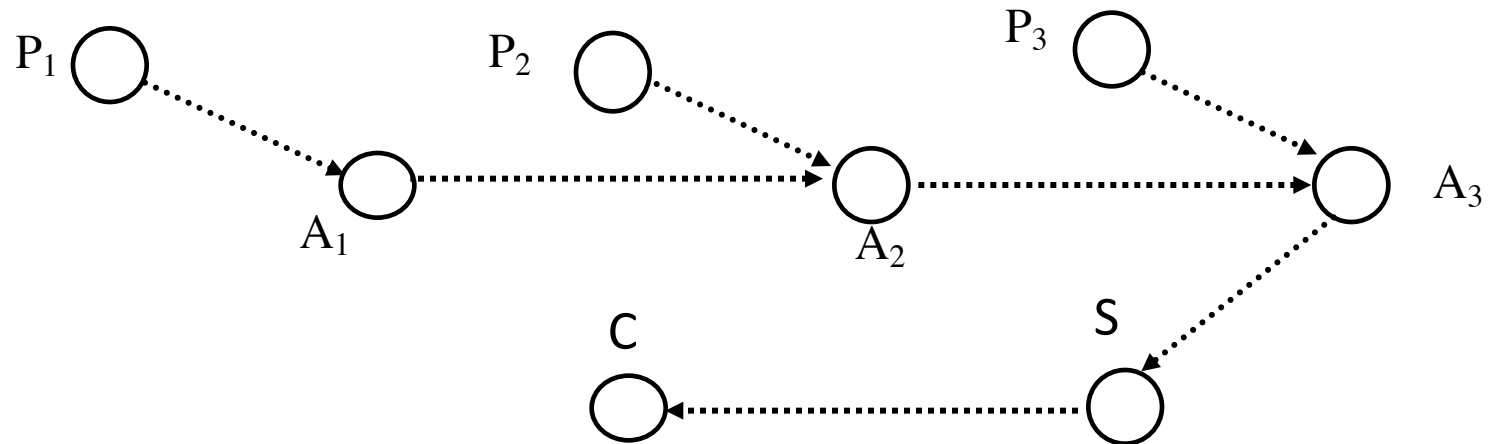
Next Step

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



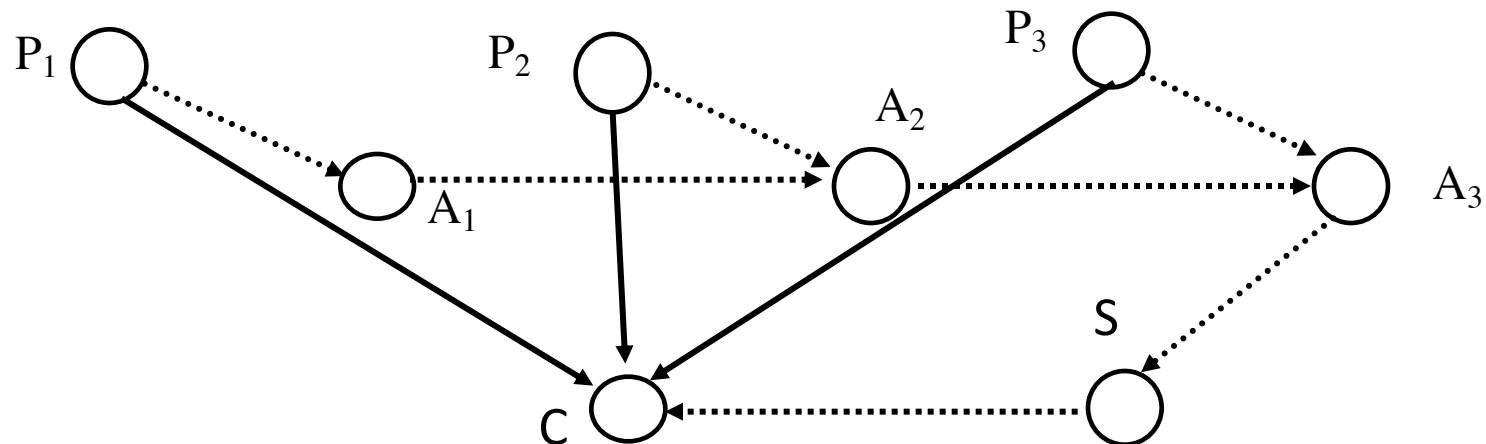
Next Step

- 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 *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*

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
 - 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

No

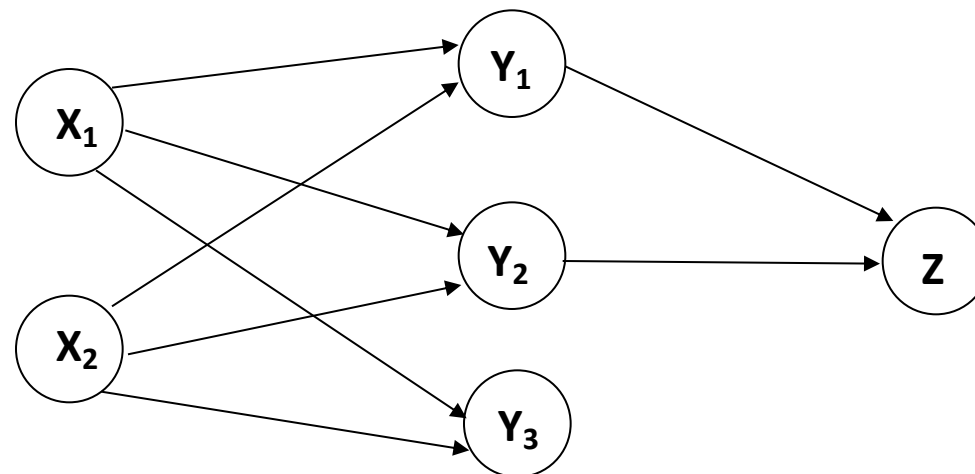
- All nodes in A have even number of incoming edges
 - 2-parent create adds 2 incoming edges
- Edge adding operation in B that can edge from X_2 to C can add one from X_3 to C
 - A cannot enter this state
 - B cannot transition to a state in which Y has even number of incoming edges
 - No remove rule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

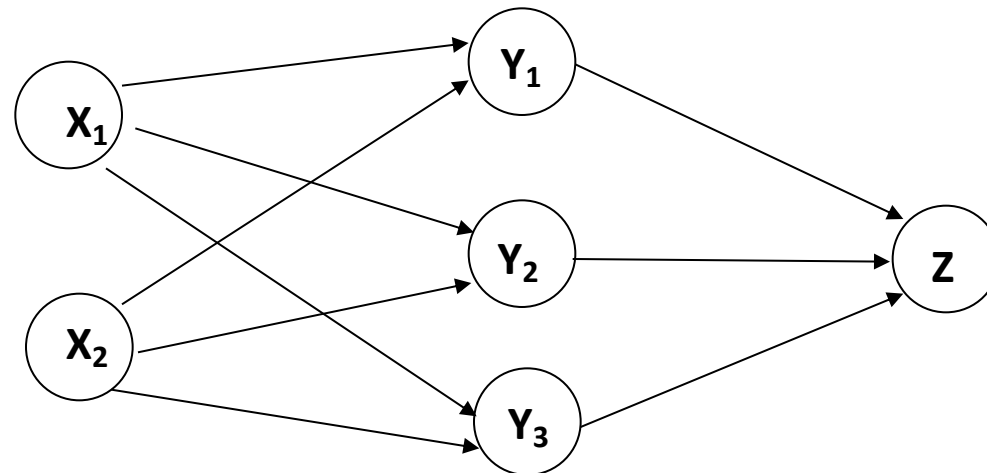
Outline of Proof

- X_1, X_2 nodes in A
 - They create Y_1, Y_2, Y_3 using multiparent create rule
 - Y_1, Y_2 create Z , again using multiparent create rule
 - *Note:* no edge from Y_3 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 **Y₂** 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