ECS 235B, Lecture 3

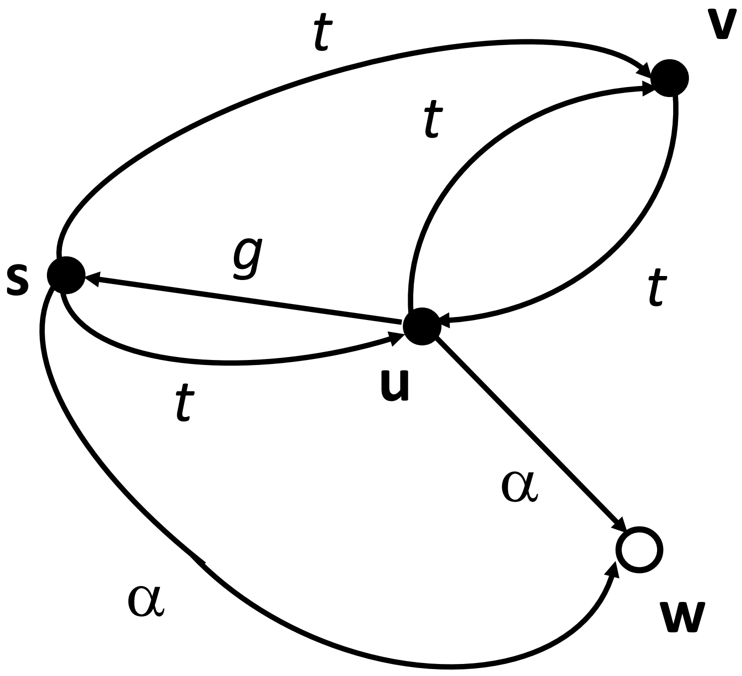
January 11, 2019

can•steal Predicate

Definition:

- *can•steal*(r, **x**, **y**, G_0) if, and only if, there is no edge from **x** to **y** labeled r in G_0, and the following hold simultaneously:
  - There is edge from **x** to **y** labeled r in G_n
  - There is a sequence of rule applications $\rho_1, \ldots, \rho_n$ such that $G_{i-1} \vdash G_i$ using ρ_i
  - For all vertices **v**, **w** in G_{i-1}, if there is an edge from **v** to **y** in G_0 labeled r, then ρ_i is ***not*** of the form "**v** grants (r to **y**) to **w**"

Example



can•steal(α, **s**, **w**, G_0):

1. **u** grants (*t* to **v**) to **s**
2. **s** takes (*t* to **u**) from **v**
3. **s** takes (α to **w**) from **u**

can•steal Theorem

- *can•steal*(r, **x**, **y**, G_0) if, and only if, the following hold simultaneously:
  a) There is no edge from **x** to **y** labeled r in G_0
  b) There exists a subject **x**′ such that **x**′ = **x** or **x**′ initially spans to **x**
  c) There exists a vertex **s** with an edge labeled α to **y** in G_0
  d) *can•share*(t, **x**′, **s**, G_0) holds

Outline of Proof

$\Rightarrow$: Assume conditions hold

- **x** subject
  - **x** gets t rights to **s**, then takes α to **y** from **s**
- **x** object
  - *can•share*(t, **x′**, **s**, G_0) holds
  - If **x′** has no α edge to **y** in G_0, **x′** takes (α to **y**) from **s** and grants it to **x**
  - If **x′** has a edge to **y** in G_0, **x′** creates surrogate **x″**, gives it (t to **s**) and (g to **x″**); then **x″** takes (α to **y**) and grants it to **x**

Outline of Proof

$\Leftarrow$: Assume *can•steal*(α, **x**, **y**, G_0) holds

- First two conditions immediate from definition of *can•steal*, *can•share*
- Third condition immediate from theorem of conditions for *can•share*
- Fourth condition: ρ minimal length sequence of rule applications deriving G_n from G_0; i smallest index such that $G_{i-1} \vdash G_i$ by rule ρ_i and adding α from some **p** to **y** in G_i
  - What is ρ_i?

Outline of Proof

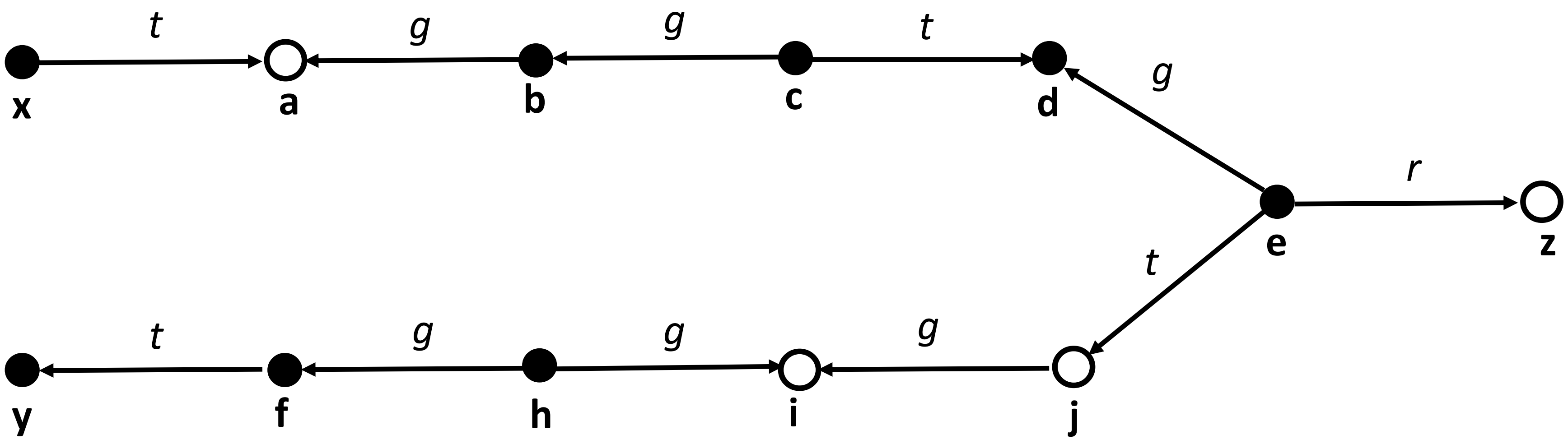
- Not remove or create rule
  - **y** exists already
- Not grant rule
  - G_i first graph in which edge labeled α to **y** is added, so by definition of *can•share*, cannot be grant
- take rule: so *can•share*(t, **p**, **s**, G_0) holds
  - So is subject **s′** such that **s′** = **s** or terminally spans to **s**
  - Sequence of islands with **x′** $\in I_1$ and **s′** $\in I_n$
- Derive witness to *can•share*(t, **x′**, **s**, G_0) that does not use "**s** grants (α to **y**) to" anyone

Conspiracy

- Minimum number of actors to generate a witness for

  can•share(α, **x**, **y**, G_0)
- Access set describes the "reach" of a subject
- Deletion set is set of vertices that cannot be involved in a transfer of rights
- Build *conspiracy graph* to capture how rights flow, and derive actors from it

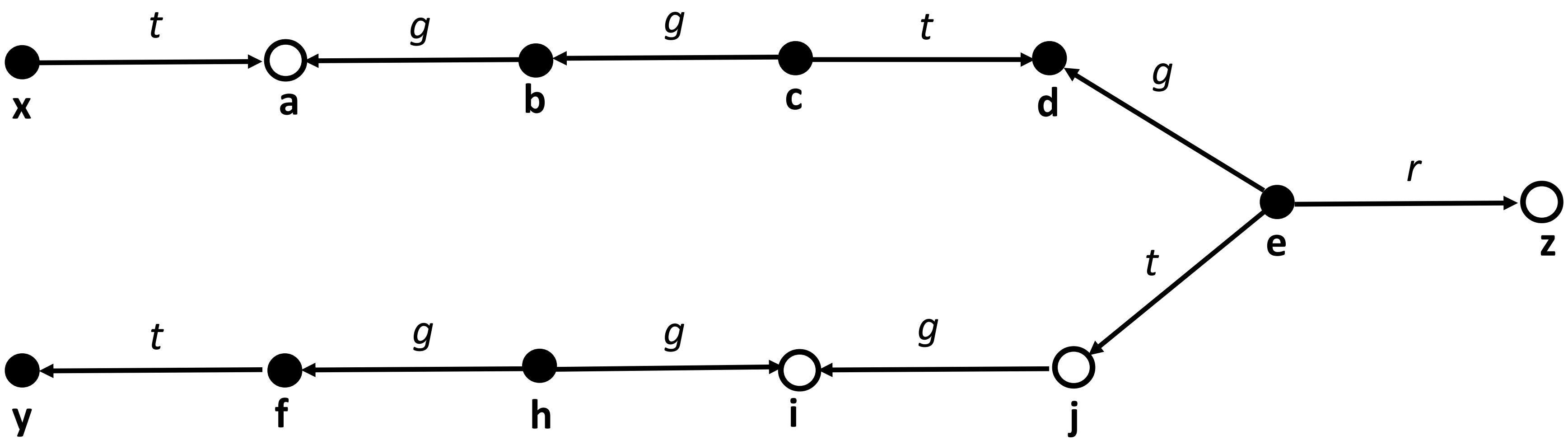
Example



Access Set

- *Access set A(**y**) with focus* **y**: set of vertices:
  - { **y** }
  - { **x** | **y** initially spans to **x** }
  - { **x′** | **y** terminally spans to **x** }
- Idea is that focus can give rights to, or acquire rights from, a vertex in this set

Example

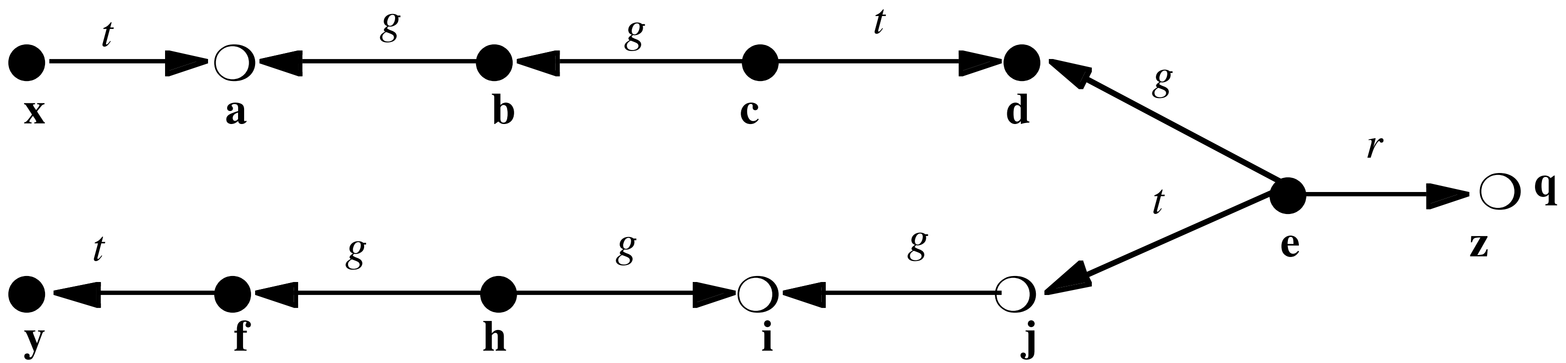


- A(**x**) = { **x**, **a** }
- A(**b**) = { **b**, **a** }
- A(**c**) = { **c**, **b**, **d** }
- A(**d**) = { **d** }
- A(**e**) = { **e**, **d**, **i**, **j** }
- A(**y**) = { **y** }
- A(**f**) = { **f**, **y** }
- A(**h**) = { **h**, **f**, **i** }

Deletion Set

- Deletion set $\delta(\mathbf{y}, \mathbf{y}')$: contains those vertices in $A(\mathbf{y}) \cap A(\mathbf{y}')$ such that:
  - **y** initially spans to **z** and **y′** terminally spans to **z**;
  - **y** terminally spans to **z** and **y′** initially spans to **z**;
  - $\mathbf{z} = \mathbf{y}$
  - $\mathbf{z} = \mathbf{y}'$
- Idea is that rights can be transferred between **y** and **y′** if this set non-empty

Example

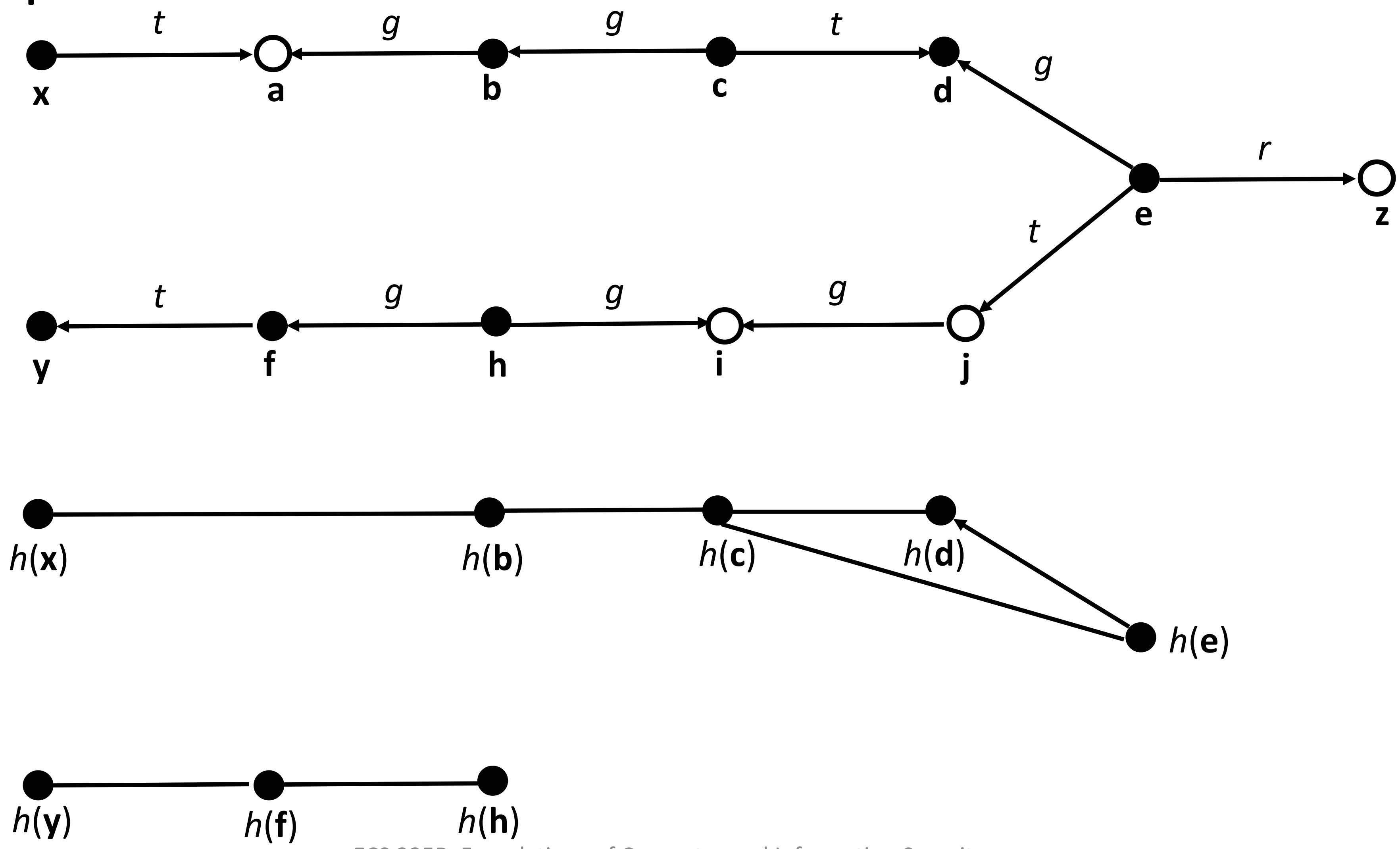


- δ(**x**, **b**) = { **a** }
- δ(**b**, **c**) = { **b** }
- δ(**c**, **d**) = { **d** }
- δ(**c**, **e**) = { **d** }
- δ(**d**, **e**) = { **d** }
- δ(**y**, **f**) = { **y** }
- δ(**h**, **f**) = { **f** }

Conspiracy Graph

- Abstracted graph H from G_0:
  - Each subject $\mathbf{x} \in G_0$ corresponds to a vertex $h(\mathbf{x}) \in H$
  - If $\delta(\mathbf{x}, \mathbf{y}) \neq \emptyset$, there is an edge between $h(\mathbf{x})$ and $h(\mathbf{y})$ in H
- Idea is that if $h(\mathbf{x})$, $h(\mathbf{y})$ are connected in H, then rights can be transferred between **x** and **y** in G_0

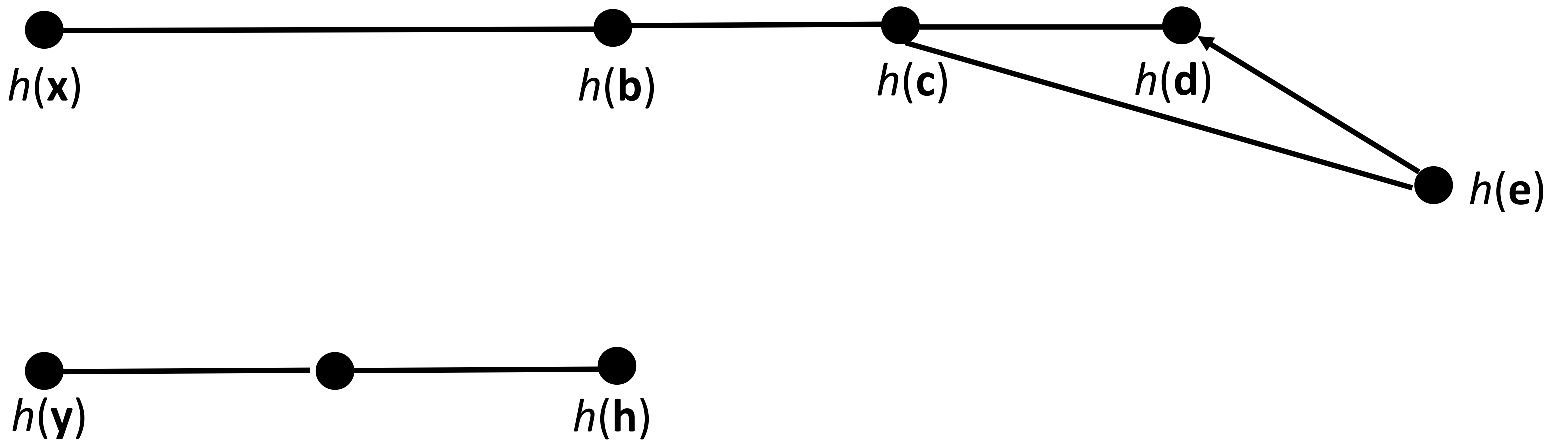
Example



Results

- $I(\mathbf{x})$: $h(\mathbf{x})$, all vertices $h(\mathbf{y})$ such that **y** initially spans to **x**
- $T(\mathbf{x})$: $h(\mathbf{x})$, all vertices $h(\mathbf{y})$ such that **y** terminally spans to **x**
- Theorem: *can•share*(α, **x**, **y**, G_0) iff there exists a path from some $h(\mathbf{p})$ in $I(\mathbf{x})$ to some $h(\mathbf{q})$ in $T(\mathbf{y})$
- Theorem: l vertices on shortest path between $h(\mathbf{p})$, $h(\mathbf{q})$ in above theorem; l conspirators necessary and sufficient to witness

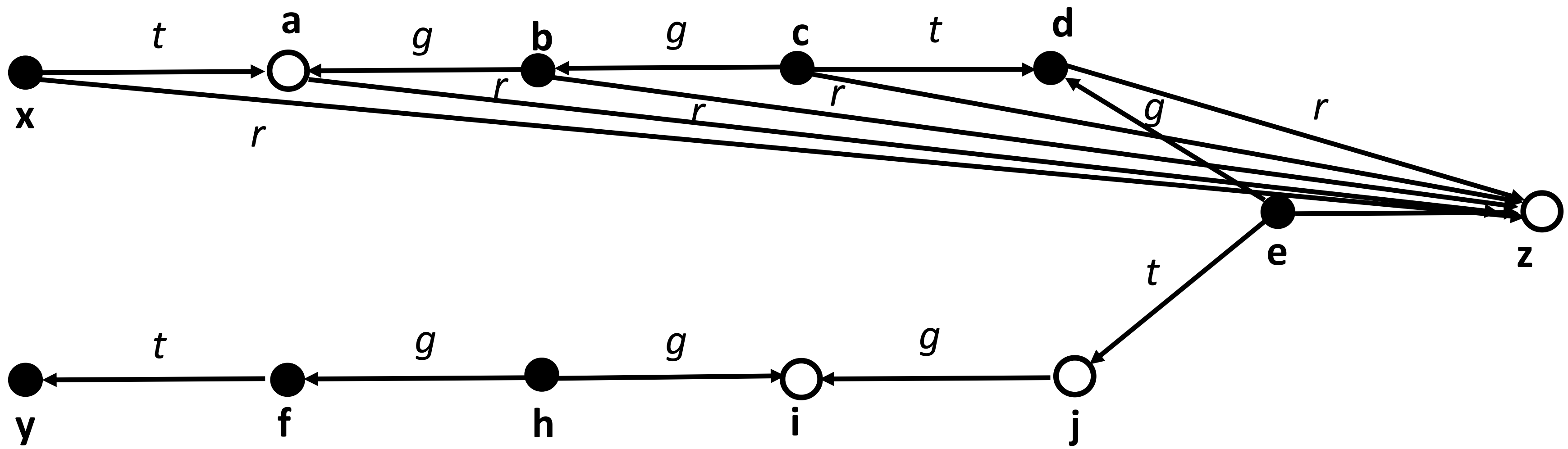
Example: Conspirators



- $I(\mathbf{x}) = \{\ h(\mathbf{x})\ \}$, $T(\mathbf{z}) = \{\ h(\mathbf{e})\ \}$
- Path between $h(\mathbf{x})$, $h(\mathbf{e})$ so $can\bullet share(r, \mathbf{x}, \mathbf{z}, G_0)$
- Shortest path between $h(\mathbf{x})$, $h(\mathbf{e})$ has 4 vertices

$\Rightarrow$ Conspirators are **e**, **c**, **b**, **x**

Example: Witness



1. **e** grants (*r* to **z**) to **d**
2. **c** takes (*r* to **z**) from **d**
3. **c** grants (*r* to **z**) to **b**
4. **b** grants (*r* to **z**) to **a**
5. **x** takes (*r* to **z**) from **a**

Key Question

- Characterize class of models for which safety is decidable
    - Existence: Take-Grant Protection Model is a member of such a class
    - Universality: In general, question undecidable, so for some models it is not decidable
- What is the dividing line?

Schematic Protection Model

- Type-based model
  - Protection type: entity label determining how control rights affect the entity
    - Set at creation and cannot be changed
  - Ticket: description of a single right over an entity
    - Entity has sets of tickets (called a *domain*)
    - Ticket is **X**/*r*, where **X** is entity and *r* right
  - Functions determine rights transfer
    - Link: are source, target "connected"?
    - Filter: is transfer of ticket authorized?

Link Predicate

- Idea: $link_i$(**X**, **Y**) if **X** can assert some control right over **Y**
- Conjunction of disjunction of:
  - $\mathbf{X}/z \in dom(\mathbf{X})$
  - $\mathbf{X}/z \in dom(\mathbf{Y})$
  - $\mathbf{Y}/z \in dom(\mathbf{X})$
  - $\mathbf{Y}/z \in dom(\mathbf{Y})$
  - **true**

Examples

- Take-Grant:

  $\mathit{link}(\mathbf{X}, \mathbf{Y}) = \mathbf{Y}/g \in \mathit{dom}(\mathbf{X}) \vee \mathbf{X}/t \in \mathit{dom}(\mathbf{Y})$
- Broadcast:

  $\mathit{link}(\mathbf{X}, \mathbf{Y}) = \mathbf{X}/b \in \mathit{dom}(\mathbf{X})$
- Pull:

  $\mathit{link}(\mathbf{X}, \mathbf{Y}) = \mathbf{Y}/p \in \mathit{dom}(\mathbf{Y})$

Filter Function

- Range is set of copyable tickets
  - Entity type, right
- Domain is subject pairs
- Copy a ticket **X**/*r*:*c* from *dom*(**Y**) to *dom*(**Z**)
  - $\mathbf{X}/rc \in dom(\mathbf{Y})$
  - $link_i(\mathbf{Y}, \mathbf{Z})$
  - $\tau(\mathbf{Y})/r{:}c \in f_i(\tau(\mathbf{Y}), \tau(\mathbf{Z}))$
- One filter function per link function

Example

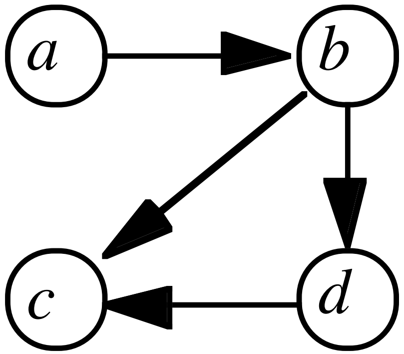
- $f(\tau(\mathbf{Y}), \tau(\mathbf{Z})) = T \times R$
  - Any ticket can be transferred (if other conditions met)
- $f(\tau(\mathbf{Y}), \tau(\mathbf{Z})) = T \times RI$
  - Only tickets with inert rights can be transferred (if other conditions met)
- $f(\tau(\mathbf{Y}), \tau(\mathbf{Z})) = \varnothing$
  - No tickets can be transferred

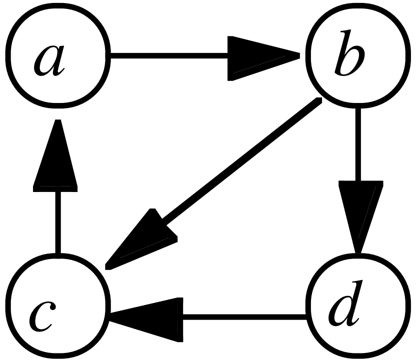
Example

- Take-Grant Protection Model
  - TS = { subjects }, TO = { objects }
  - $RC = \{ tc, gc \}$, $RI = \{ rc, wc \}$
  - $link(\mathbf{p}, \mathbf{q}) = \mathbf{p}/t \in dom(\mathbf{q}) \vee \mathbf{q}/g \in dom(\mathbf{p})$
  - $f(subject, subject) = \{ subject, object \} \times \{ tc, gc, rc, wc \}$

Create Operation

- Must handle type, tickets of new entity
- Relation *cc*(*a*, *b*) [*cc* for *can-create*]
  - Subject of type *a* can create entity of type *b*
- Rule of acyclic creates:





Types

- $cr(a, b)$: tickets created when subject of type a creates entity of type b [*cr* for *create-rule*]
- **B** object: $cr(a, b) \subseteq \{ b/r{:}c \in RI \}$
  - **A** gets **B**/$r{:}c$ iff $b/r{:}c \in cr(a, b)$
- **B** subject: $cr(a, b)$ has two subsets
  - $cr_P(a, b)$ added to **A**, $cr_C(a, b)$ added to **B**
  - **A** gets **B**/$r{:}c$ if $b/r{:}c \in cr_P(a, b)$
  - **B** gets **A**/$r{:}c$ if $a/r{:}c \in cr_C(a, b)$

Non-Distinct Types

$cr(a, a)$: who gets what?

- *self*/r:c are tickets for creator
- a/r:c tickets for created

$cr(a, a) = \{ a/r{:}c, \mathit{self}/r{:}c \mid r{:}c \in R\}$

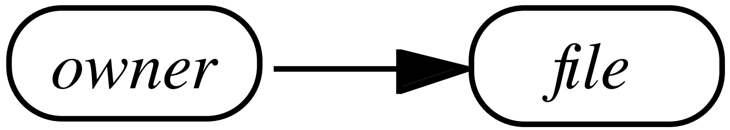
Attenuating Create Rule

$cr(a, b)$ attenuating if:

1. $cr_C(a, b) \subseteq cr_P(a, b)$ and
2. $a/r{:}c \in cr_P(a, b) \Rightarrow self/r{:}c \in cr_P(a, b)$

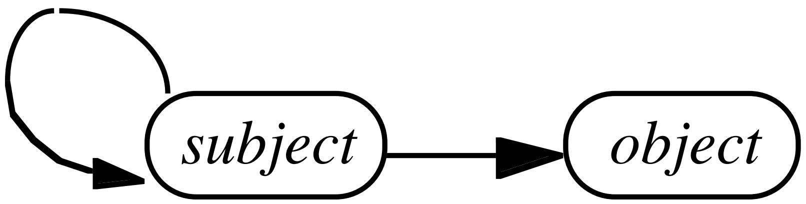
Example: Owner-Based Policy

- Users can create files, creator can give itself any inert rights over file
  - $cc = \{\ (\ user\ ,\ \ file\ )\ \}$
  - $cr(user, file) = \{\ file/r{:}c \mid r \in RI\ \}$
- Attenuating, as graph is acyclic, loop free



Example: Take-Grant

- Say subjects create subjects (type *s*), objects (type *o*), but get only inert rights over latter
  - $cc = \{ (s, s), (s, o) \}$
  - $cr_C(a, b) = \varnothing$
  - $cr_P(s, s) = \{s/tc, s/gc, s/rc, s/wc\}$
  - $cr_P(s, o) = \{s/rc, s/wc\}$
- Not attenuating, as no *self* tickets provided; *subject* creates *subject*



Safety Analysis

- Goal: identify types of policies with tractable safety analyses
- Approach: derive a state in which additional entries, rights do not affect the analysis; then analyze this state
  - Called a *maximal state*

Definitions

- System begins at initial state
- Authorized operation causes *legal transition*
- Sequence of legal transitions moves system into final state
  - This sequence is a *history*
  - Final state is *derivable* from history, initial state

More Definitions

- States represented by ^{h}
- Set of subjects SUB^h, entities ENT^h
- Link relation in context of state h is $link^h$
- Dom relation in context of state h is dom^h

$path^h$(X,Y)

- **X**, **Y** connected by one link or a sequence of links
- Formally, either of these hold:
  - for some i, $link_i^h(\mathbf{X}, \mathbf{Y})$; or
  - there is a sequence of subjects $\mathbf{X}_0, \dots, \mathbf{X}_n$ such that $link_i^h(\mathbf{X}, \mathbf{X}_0)$, $link_i^h(\mathbf{X}_n, \mathbf{Y})$, and for $k = 1, \dots, n$, $link_i^h(\mathbf{X}_{k-1}, \mathbf{X}_k)$
- If multiple such paths, refer to $path_j^h(\mathbf{X}, \mathbf{Y})$

Capacity $cap(path^h(X,Y))$

- Set of tickets that can flow over $path^h(\mathbf{X},\mathbf{Y})$
  - If $link_i^h(\mathbf{X},\mathbf{Y})$: set of tickets that can be copied over the link (i.e., $f_i(\tau(\mathbf{X}), \tau(\mathbf{Y}))$)
  - Otherwise, set of tickets that can be copied over *all* links in the sequence of links making up the $path^h(\mathbf{X},\mathbf{Y})$
- Note: all tickets (except those for the final link) *must* be copyable

Flow Function

- Idea: capture flow of tickets around a given state of the system
- Let there be *m* $path^h$s between subjects **X** and **Y** in state *h*. Then *flow function*

$$flow^h: SUB^h \times SUB^h \rightarrow 2^{T \times R}$$

is:

$$flow^h(\mathbf{X},\mathbf{Y}) = \cup_{i=1,\dots,m}\, cap(path_i^h(\mathbf{X},\mathbf{Y}))$$

Properties of Maximal State

- Maximizes flow between all pairs of subjects
  - State is called *
  - Ticket in *flow**(**X**,**Y**) means there exists a sequence of operations that can copy the ticket from **X** to **Y**
- Questions
  - Is maximal state unique?
  - Does every system have one?

Formal Definition

- Definition: $g \leq_0 h$ holds iff for all **X**, **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 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 $h \in H$, $h \leq_0 m$
- Outline of proof: induction
  - Basis: $H = \varnothing$; trivially true
  - Step: $|H'| = n + 1$, where $H' = G \cup \{h\}$. By IH, there is a $g \in G$ such that $x \leq_0 g$ for all $x \in G$.

Outline of Proof

- M interleaving histories of *g*, *h* which:
  - Preserves relative order of transitions in *g*, *h*
  - Omits second create operation if duplicated
- *M* ends up at state *m*
- If $path^g(\mathbf{X},\mathbf{Y})$ for **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 **X**, $\mathbf{Y} \in SUB^h$, $path^m(\mathbf{X},\mathbf{Y})$
  - So $h \leq_0 m$
- Hence *m* maximal state in *H′*

Answer to Second Question

- Theorem: every system has a maximal state *
- Outline of proof: *K* is set of derivable states containing exactly one state from each equivalence class of derivable states
  - Consider **X**, **Y** in SUB^0. Flow function's range is $2^{T \times R}$, so can take at most $2^{|T \times R|}$ values. As there are $|SUB^0|^2$ pairs of subjects in SUB^0, at most $2^{|T \times R|}\,|SUB^0|^2$ distinct equivalence classes; so *K* is finite
- Result follows from lemma

Safety Question

- In this model:

  Is it possible to have 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 or which can demand **X**/*rc* and τ(**X**)/*r*:*c* 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.